TUGAS BESAR KULIAH SISTEM PENGUKURAN BERBASIS CITRA

Nama : Ronggur Mahendra Widya Putra

NIM : 13519008

In [48]:

```
# Import Library
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import cv2
import os
import numpy as np
import scipy
from skimage import color, data, restoration
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from random import uniform
from tensorflow.keras.layers import BatchNormalization
```

Loading Data dari folder ./Data/ClassifiedData

In [21]:

```
ClassifiedDataDir = "./Data/ClassifiedData/Train"
BATCH_SIZE = 32
IMG_SIZE = (128, 128)
train_dataset = tf.keras.utils.image_dataset_from_directory(
    directory = ClassifiedDataDir,
        labels='inferred',
        label_mode='int',
        class_names=None,
        color_mode='rgb',
        batch_size=BATCH_SIZE,
        image_size=IMG_SIZE,
        shuffle=True,
        seed=1337,
        subset="training",
        validation_split=0.1,
        interpolation='bilinear',
        follow_links=False,
        crop_to_aspect_ratio=False,
        # rescale = 1./255,
    )
validation_dataset = tf.keras.utils.image_dataset_from_directory(
    directory = ClassifiedDataDir,
        labels='inferred',
        label_mode='int',
        class_names=None,
        color_mode='rgb',
        batch_size=BATCH_SIZE,
        image_size=IMG_SIZE,
        shuffle=True,
        seed=1337,
        subset="validation",
        validation_split=0.1,
        interpolation='bilinear',
        follow_links=False,
        crop_to_aspect_ratio=False,
        # rescale = 1./255,
    )


test_dataset = tf.keras.utils.image_dataset_from_directory(
    directory = "./Data/ClassifiedData/Test",
        labels='inferred',
        label_mode='int',
        class_names=None,
        color_mode='rgb',
        batch_size=BATCH_SIZE,
        image_size=IMG_SIZE,
        shuffle=True,
        seed=None,
        interpolation='bilinear',
        follow_links=False,
        crop_to_aspect_ratio=False,
        # rescale = 1./255,
    )
```

```
Found 3525 files belonging to 24 classes.
Using 3173 files for training.
```

```
Found 3525 files belonging to 24 classes.
Using 352 files for validation.
Found 312 files belonging to 24 classes.
```

List Class

In [22]:

```
# for element in dataset.as_numpy_iterator():
#     print(element)
class_names = train_dataset.class_names
print(class_names)
print(len(class_names))
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y']
24
```

In [23]:

```
# print('Number of training batches: %d' % tf.data.experimental.cardinality(train_dataset).
# print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation_da
```

Optimization buffer untuk Dataset

In [24]:

```
# DataSet
AUTOTUNE = tf.data.AUTOTUNE
train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

Model 1

In [47]:

```
# # # Model 1 Initialize
# input_size = 128
# filter_size = 14
# num_filter = 8
# maxpool_size = 2
# batch_size = BATCH_SIZE
# epochs = 30

# model1 = tf.keras.models.Sequential([
#     tf.keras.layers.InputLayer(input_shape=(input_size, input_size, 3)),
#     tf.keras.layers.Reshape((256, 256 * 3)),
#     tf.keras.layers.Bidirectional(tf.keras.layers.LSTM( 256, return_sequences=True, retur
#     tf.keras.layers.Bidirectional(tf.keras.layers.LSTM( 256 )),
#     tf.keras.layers.Dense( 256 ),
#     tf.keras.layers.Dropout(.2),
#     tf.keras.layers.Dense( 256 ),
# ])

# # model = models.Sequential()
# # model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(input_size, input_s
# # model.add(layers.MaxPooling2D((2, 2)))
# # model.add(layers.Conv2D(64, (3, 3), activation='relu'))
# # model.add(layers.MaxPooling2D((2, 2)))
# # model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# model1.add(tf.keras.layers.Flatten())
# # model.add(tf.keras.layers.Dense(24))
# model1.add(tf.keras.layers.Dense(24, activation = 'softmax'))

# # model.add(layers.Flatten())
# # model.add(layers.Dense(64, activation='relu'))
# # model.add(layers.Dense(24))

# model1.summary()
```

In [26]:

```
# # Optimizer
# optimizer = tf.keras.optimizers.Nadam(
#     learning_rate=0.00001, beta_1=0.9, beta_2=0.999, epsilon=1e-07,
#     name='Nadam'
# ) # 0.00001

# # Loss Fn
# lossfn = tf.keras.losses.SparseCategoricalCrossentropy( from_logits=False, reduction=tf.k

# # Model Summary
# model.compile(
#     optimizer="rmsprop",
#     loss="sparse_categorical_crossentropy",
#     metrics=["sparse_categorical_accuracy"],
# )
```

Creating CNN Model

In [27]:

```
# Model 2

input_size = 128
filter_size = 3
num_filter = 8
maxpool_size = 2
batch_size = BATCH_SIZE
epochs = 30

steps_per_epoch = 24720/batch_size

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(16, (filter_size,filter_size),
                 input_shape= (input_size,input_size,3),
                 activation ='relu',
                 padding='same'))
model.add(tf.keras.layers.Conv2D(16, (filter_size,filter_size),
                 input_shape= (input_size,input_size,3),
                 activation ='relu',
                 padding='same'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=1))
model.add(tf.keras.layers.Dropout(uniform(0, 1)))

model.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
                 activation='relu',
                 padding='valid'))
model.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
                 activation='relu',
                 padding='valid'))
model.add(BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=2))
model.add(tf.keras.layers.Dropout(uniform(0, 1)))

model.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
                 activation='relu',
                 padding='valid'))
model.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
                 activation='relu',
                 padding='valid'))
model.add(BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=2))
model.add(tf.keras.layers.Dropout(uniform(0, 1)))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(120, activation='relu'))
model.add(tf.keras.layers.Dense(24,activation='softmax'))
```

In [28]:

```
METRICS = [ 'accuracy']#, 'precision','recall']
model.compile( optimizer= tf.keras.optimizers.Adam(lr=0.001),loss='sparse_categorical_cross

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 128, 128, 16)      448

 conv2d_7 (Conv2D)           (None, 128, 128, 16)      2320

 max_pooling2d_3 (MaxPooling  (None, 127, 127, 16)     0
 2D)

 dropout_3 (Dropout)         (None, 127, 127, 16)      0

 conv2d_8 (Conv2D)           (None, 125, 125, 32)      4640

 conv2d_9 (Conv2D)           (None, 123, 123, 32)      9248

 batch_normalization_2 (Batc  (None, 123, 123, 32)     128
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 61, 61, 32)       0
 2D)

 dropout_4 (Dropout)         (None, 61, 61, 32)        0

 conv2d_10 (Conv2D)          (None, 59, 59, 32)        9248

 conv2d_11 (Conv2D)          (None, 57, 57, 32)        9248

 batch_normalization_3 (Batc  (None, 57, 57, 32)       128
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 28, 28, 32)       0
 2D)

 dropout_5 (Dropout)         (None, 28, 28, 32)        0

 flatten_1 (Flatten)         (None, 25088)             0

 dense_2 (Dense)             (None, 120)               3010680

 dense_3 (Dense)             (None, 24)                2904

=================================================================
Total params: 3,048,992
Trainable params: 3,048,864
Non-trainable params: 128
_________________________________________________________________
```

Makingsure Training using GPU

In [29]:

```
print(tf.test.is_built_with_cuda())
print(tf.config.list_physical_devices('GPU'))
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
True
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
Num GPUs Available:  1
```

Training

In [30]:

```
# Training
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data = (validation_dataset)
    )
```

```
Epoch 1/30
100/100 [==============================] - 8s 69ms/step - loss: 3.7018 - acc
uracy: 0.1210 - val_loss: 19.0571 - val_accuracy: 0.0511
Epoch 2/30
100/100 [==============================] - 7s 70ms/step - loss: 2.1337 - acc
uracy: 0.3325 - val_loss: 14.2802 - val_accuracy: 0.1477
Epoch 3/30
100/100 [==============================] - 8s 72ms/step - loss: 1.4926 - acc
uracy: 0.5317 - val_loss: 2.2757 - val_accuracy: 0.4830
Epoch 4/30
100/100 [==============================] - 8s 73ms/step - loss: 0.8629 - acc
uracy: 0.7182 - val_loss: 3.0339 - val_accuracy: 0.5227
Epoch 5/30
100/100 [==============================] - 8s 73ms/step - loss: 0.5989 - acc
uracy: 0.7945 - val_loss: 12.5601 - val_accuracy: 0.1733
Epoch 6/30
100/100 [==============================] - 8s 75ms/step - loss: 0.4056 - acc
uracy: 0.8544 - val_loss: 9.9583 - val_accuracy: 0.2670
Epoch 7/30
100/100 [==============================] - 8s 77ms/step - loss: 0.3230 - acc
uracy: 0.8951 - val_loss: 8.1203 - val_accuracy: 0.3011
Epoch 8/30
100/100 [==============================] - 8s 80ms/step - loss: 0.2675 - acc
uracy: 0.9171 - val_loss: 0.8034 - val_accuracy: 0.7500
Epoch 9/30
100/100 [==============================] - 8s 78ms/step - loss: 0.2010 - acc
uracy: 0.9354 - val_loss: 0.7667 - val_accuracy: 0.7983
Epoch 10/30
100/100 [==============================] - 8s 75ms/step - loss: 0.1977 - acc
uracy: 0.9392 - val_loss: 1.3063 - val_accuracy: 0.6705
Epoch 11/30
100/100 [==============================] - 8s 75ms/step - loss: 0.2042 - acc
uracy: 0.9382 - val_loss: 1.2486 - val_accuracy: 0.7273
Epoch 12/30
100/100 [==============================] - 8s 75ms/step - loss: 0.1601 - acc
uracy: 0.9499 - val_loss: 0.5626 - val_accuracy: 0.8324
Epoch 13/30
100/100 [==============================] - 8s 75ms/step - loss: 0.1331 - acc
uracy: 0.9581 - val_loss: 0.9256 - val_accuracy: 0.7358
Epoch 14/30
100/100 [==============================] - 8s 75ms/step - loss: 0.1202 - acc
uracy: 0.9634 - val_loss: 1.4013 - val_accuracy: 0.6420
Epoch 15/30
100/100 [==============================] - 8s 76ms/step - loss: 0.1233 - acc
uracy: 0.9638 - val_loss: 0.7337 - val_accuracy: 0.8295
Epoch 16/30
100/100 [==============================] - 8s 76ms/step - loss: 0.1327 - acc
uracy: 0.9587 - val_loss: 1.1404 - val_accuracy: 0.7784
Epoch 17/30
100/100 [==============================] - 8s 77ms/step - loss: 0.1318 - acc
uracy: 0.9628 - val_loss: 0.8612 - val_accuracy: 0.8040
```

```
Epoch 18/30
100/100 [==============================] - 8s 76ms/step - loss: 0.1076 - acc
uracy: 0.9660 - val_loss: 2.7944 - val_accuracy: 0.6449
Epoch 19/30
100/100 [==============================] - 8s 76ms/step - loss: 0.0864 - acc
uracy: 0.9723 - val_loss: 2.8514 - val_accuracy: 0.6449
Epoch 20/30
100/100 [==============================] - 8s 76ms/step - loss: 0.0746 - acc
uracy: 0.9729 - val_loss: 0.1848 - val_accuracy: 0.9574
Epoch 21/30
100/100 [==============================] - 8s 78ms/step - loss: 0.1082 - acc
uracy: 0.9675 - val_loss: 4.1118 - val_accuracy: 0.4517
Epoch 22/30
100/100 [==============================] - 8s 78ms/step - loss: 0.0945 - acc
uracy: 0.9697 - val_loss: 2.9749 - val_accuracy: 0.5597
Epoch 23/30
100/100 [==============================] - 8s 75ms/step - loss: 0.0646 - acc
uracy: 0.9801 - val_loss: 6.3521 - val_accuracy: 0.3523
Epoch 24/30
100/100 [==============================] - 8s 77ms/step - loss: 0.0713 - acc
uracy: 0.9748 - val_loss: 11.6890 - val_accuracy: 0.2358
Epoch 25/30
100/100 [==============================] - 8s 81ms/step - loss: 0.0651 - acc
uracy: 0.9776 - val_loss: 6.2285 - val_accuracy: 0.3494
Epoch 26/30
100/100 [==============================] - 8s 79ms/step - loss: 0.0851 - acc
uracy: 0.9760 - val_loss: 1.5400 - val_accuracy: 0.7585
Epoch 27/30
100/100 [==============================] - 8s 79ms/step - loss: 0.0496 - acc
uracy: 0.9849 - val_loss: 1.5441 - val_accuracy: 0.7614
Epoch 28/30
100/100 [==============================] - 8s 79ms/step - loss: 0.0703 - acc
uracy: 0.9798 - val_loss: 8.9021 - val_accuracy: 0.2841
Epoch 29/30
100/100 [==============================] - 8s 80ms/step - loss: 0.0542 - acc
uracy: 0.9833 - val_loss: 7.8091 - val_accuracy: 0.3068
Epoch 30/30
100/100 [==============================] - 8s 77ms/step - loss: 0.0571 - acc
uracy: 0.9817 - val_loss: 1.2181 - val_accuracy: 0.8011
```

In [31]:

```
loss, acc = model.evaluate(test_dataset)
print(loss, acc)
```

```
10/10 [==============================] - 1s 15ms/step - loss: 0.6346 - accur
acy: 0.8910
0.6345966458320618 0.8910256624221802
```

Saving model to load later

In [49]:

```
model.save('saved_model/FinalModel')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_co
nvolution_op, _jit_compiled_convolution_op while saving (showing 5 of 6). Th
ese functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/FinalModel\assets

INFO:tensorflow:Assets written to: saved_model/FinalModel\assets
```

Model's Performance

In [45]:

```
TrainingAccuracy = history.history['accuracy'][len(history.history['accuracy'])-1]
print("Train accuracy: {:5.2f}%".format(100 * TrainingAccuracy))
print("Test accuracy: {:5.2f}%".format(100 * acc))
```

```
Train accuracy: 98.17%
Test accuracy: 89.10%
```

Training History

In [35]:

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

# test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

Out[35]:

```
<matplotlib.legend.Legend at 0x224ea73adc0>
```