

TUGAS BESAR KULIAH SISTEM PENGUKURAN BERBASIS CITRA

Nama : Ronggur Mahendra Widya Putra

NIM : 13519008

In [1]:

```
# 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 [2]:

```

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=1234,
    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=1234,
    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 3088 files belonging to 24 classes.
 Using 2780 files for training.

Found 3088 files belonging to 24 classes.
 Using 308 files for validation.
 Found 312 files belonging to 24 classes.

List Class

In [3]:

```
# 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 [4]:

```
# 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 [5]:

```
# 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)
```

Makingsure Training using GPU

In [6]:

```
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
```

Creating CNN Model

Model 1

In [7]:

```

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

model_1 = models.Sequential()
model_1.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(input_size, input_size, 1)))
model_1.add(layers.MaxPooling2D((2, 2)))
model_1.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_1.add(layers.MaxPooling2D((2, 2)))
model_1.add(layers.Conv2D(64, (3, 3), activation='relu'))

model_1.add(tf.keras.layers.Flatten())
model_1.add(tf.keras.layers.Dense(120))
model_1.add(tf.keras.layers.Dense(24, activation = 'softmax'))

model_1.summary()

```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 120)	6021240
dense_1 (Dense)	(None, 24)	2904
=====		
Total params: 6,080,464		
Trainable params: 6,080,464		
Non-trainable params: 0		

In [8]:

```
# Optimizer
optimizer1 = 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
lossfn1 = tf.keras.losses.SparseCategoricalCrossentropy( from_logits=False, reduction=tf.ke

# Model Summary
model_1.compile(
    optimizer="rmsprop",
    loss="sparse_categorical_crossentropy",
    metrics=["sparse_categorical_accuracy"],
)
```

Model 2

In [9]:

```
# # 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_2 = tf.keras.models.Sequential()
model_2.add(tf.keras.layers.Conv2D(16, (filter_size,filter_size),
    input_shape= (input_size,input_size,3),
    activation = 'relu',
    padding='same'))
model_2.add(tf.keras.layers.Conv2D(16, (filter_size,filter_size),
    input_shape= (input_size,input_size,3),
    activation = 'relu',
    padding='same'))
model_2.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=1))
model_2.add(tf.keras.layers.Dropout(uniform(0, 1)))

model_2.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
    activation='relu',
    padding='valid'))
model_2.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
    activation='relu',
    padding='valid'))
model_2.add(tf.keras.layers.BatchNormalization())
model_2.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=2))
model_2.add(tf.keras.layers.Dropout(uniform(0, 1)))

model_2.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
    activation='relu',
    padding='valid'))
model_2.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
    activation='relu',
    padding='valid'))
model_2.add(tf.keras.layers.BatchNormalization())

model_2.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
    activation='relu',
    padding='valid'))
model_2.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),
    activation='relu',
    padding='valid'))
model_2.add(tf.keras.layers.BatchNormalization())
model_2.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=2))
model_2.add(tf.keras.layers.Dropout(uniform(0, 1)))

model_2.add(tf.keras.layers.Flatten())
model_2.add(tf.keras.layers.Dense(120, activation='relu'))
model_2.add(tf.keras.layers.Dense(120, activation='relu'))
model_2.add(tf.keras.layers.Dense(24,activation='softmax'))
```

In [10]:

```
METRICS = [ 'sparse_categorical_accuracy']
model_2.compile( optimizer= tf.keras.optimizers.Adam(lr=0.001),loss='sparse_categorical_crossentropy')
model_2.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
conv2d_3 (Conv2D)	(None, 128, 128, 16)	448
conv2d_4 (Conv2D)	(None, 128, 128, 16)	2320
max_pooling2d_2 (MaxPooling2D)	(None, 127, 127, 16)	0
dropout (Dropout)	(None, 127, 127, 16)	0
conv2d_5 (Conv2D)	(None, 125, 125, 32)	4640
conv2d_6 (Conv2D)	(None, 123, 123, 32)	9248
batch_normalization (Batch Normalization)	(None, 123, 123, 32)	128
max_pooling2d_3 (MaxPooling2D)	(None, 61, 61, 32)	0
dropout_1 (Dropout)	(None, 61, 61, 32)	0
conv2d_7 (Conv2D)	(None, 59, 59, 32)	9248
conv2d_8 (Conv2D)	(None, 57, 57, 32)	9248
batch_normalization_1 (Batch Normalization)	(None, 57, 57, 32)	128
conv2d_9 (Conv2D)	(None, 55, 55, 32)	9248
conv2d_10 (Conv2D)	(None, 53, 53, 32)	9248
batch_normalization_2 (Batch Normalization)	(None, 53, 53, 32)	128
max_pooling2d_4 (MaxPooling2D)	(None, 26, 26, 32)	0
dropout_2 (Dropout)	(None, 26, 26, 32)	0
flatten_1 (Flatten)	(None, 21632)	0
dense_2 (Dense)	(None, 120)	2595960
dense_3 (Dense)	(None, 120)	14520
dense_4 (Dense)	(None, 24)	2904

=====

Total params: 2,667,416

Trainable params: 2,667,224
Non-trainable params: 192

```
c:\Users\ZEPHYRUS GU502GU\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:110: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.  
super(Adam, self).__init__(name, **kwargs)
```

Model 3

In [11]:

```
#Model 3  
input_size = 128  
filter_size = 14  
num_filter = 8  
maxpool_size = 2  
batch_size = BATCH_SIZE  
epochs = 30  
  
model_3 = tf.keras.models.Sequential()  
model_3.add(tf.keras.layers.Conv2D(16, (filter_size,filter_size),  
                                   input_shape= (input_size,input_size,3),  
                                   activation = 'relu',  
                                   padding='same'))  
model_3.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),  
                                   activation='relu',  
                                   padding='valid'))  
model_3.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=1))  
  
model_3.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),  
                                   activation='relu',  
                                   padding='valid'))  
model_3.add(tf.keras.layers.Conv2D(32, (filter_size,filter_size),  
                                   activation='relu',  
                                   padding='valid'))  
model_3.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=2))  
model_3.add(tf.keras.layers.Dropout(uniform(0, 1)))  
  
model_3.add(tf.keras.layers.Conv2D(64, (filter_size,filter_size),  
                                   activation='relu',  
                                   padding='valid'))  
model_3.add(tf.keras.layers.Conv2D(64, (filter_size,filter_size),  
                                   activation='relu',  
                                   padding='valid'))  
model_3.add(tf.keras.layers.BatchNormalization())  
model_3.add(tf.keras.layers.MaxPooling2D(pool_size=(maxpool_size, maxpool_size),strides=2))  
model_3.add(tf.keras.layers.Dropout(uniform(0, 1)))  
  
model_3.add(tf.keras.layers.Flatten())  
model_3.add(tf.keras.layers.Dense(128, activation='relu'))  
model_3.add(tf.keras.layers.Dense(128, activation='relu'))  
model_3.add(tf.keras.layers.Dense(24,activation='softmax'))
```


In [12]:

```
METRICS = [ 'sparse_categorical_accuracy' ]

model_3.compile( optimizer= tf.keras.optimizers.Adam(lr=0.001), loss='sparse_categorical_crossentropy')

model_3.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
=====		
conv2d_11 (Conv2D)	(None, 128, 128, 16)	9424
conv2d_12 (Conv2D)	(None, 115, 115, 32)	100384
max_pooling2d_5 (MaxPooling2D)	(None, 114, 114, 32)	0
conv2d_13 (Conv2D)	(None, 101, 101, 32)	200736
conv2d_14 (Conv2D)	(None, 88, 88, 32)	200736
max_pooling2d_6 (MaxPooling2D)	(None, 44, 44, 32)	0
dropout_3 (Dropout)	(None, 44, 44, 32)	0
conv2d_15 (Conv2D)	(None, 31, 31, 64)	401472
conv2d_16 (Conv2D)	(None, 18, 18, 64)	802880
batch_normalization_3 (Batch Normalization)	(None, 18, 18, 64)	256
max_pooling2d_7 (MaxPooling2D)	(None, 9, 9, 64)	0
dropout_4 (Dropout)	(None, 9, 9, 64)	0
flatten_2 (Flatten)	(None, 5184)	0
dense_5 (Dense)	(None, 128)	663680
dense_6 (Dense)	(None, 128)	16512
dense_7 (Dense)	(None, 24)	3096

```
=====
Total params: 2,399,176
Trainable params: 2,399,048
Non-trainable params: 128
```

Training

In [13]:

```
# Training Model 1
history1 = model_1.fit(
    train_dataset,
    epochs=30,
    shuffle=True,
    validation_data = (validation_dataset)
)
```

Epoch 1/30

87/87 [=====] - 8s 51ms/step - loss: 165.8931 - sparse_categorical_accuracy: 0.4658 - val_loss: 0.5191 - val_sparse_categorical_accuracy: 0.8864

Epoch 2/30

87/87 [=====] - 4s 48ms/step - loss: 3.4740 - sparse_categorical_accuracy: 0.8590 - val_loss: 0.2727 - val_sparse_categorical_accuracy: 0.9318

Epoch 3/30

87/87 [=====] - 5s 49ms/step - loss: 21.3086 - sparse_categorical_accuracy: 0.8338 - val_loss: 0.4336 - val_sparse_categorical_accuracy: 0.9675

Epoch 4/30

87/87 [=====] - 5s 50ms/step - loss: 2.3258 - sparse_categorical_accuracy: 0.9421 - val_loss: 0.1463 - val_sparse_categorical_accuracy: 0.9870

Epoch 5/30

87/87 [=====] - 5s 50ms/step - loss: 10.2403 - sparse_categorical_accuracy: 0.9428 - val_loss: 0.4589 - val_sparse_categorical_accuracy: 0.9903

Epoch 6/30

87/87 [=====] - 5s 50ms/step - loss: 2.9069 - sparse_categorical_accuracy: 0.9723 - val_loss: 0.3268 - val_sparse_categorical_accuracy: 0.9968

Epoch 7/30

87/87 [=====] - 5s 54ms/step - loss: 14.2663 - sparse_categorical_accuracy: 0.9622 - val_loss: 0.8427 - val_sparse_categorical_accuracy: 0.9675

Epoch 8/30

87/87 [=====] - 7s 71ms/step - loss: 5.2379 - sparse_categorical_accuracy: 0.9781 - val_loss: 1.0001 - val_sparse_categorical_accuracy: 0.9903

Epoch 9/30

87/87 [=====] - 6s 64ms/step - loss: 1.6106 - sparse_categorical_accuracy: 0.9903 - val_loss: 0.2118 - val_sparse_categorical_accuracy: 0.9903

Epoch 10/30

87/87 [=====] - 6s 62ms/step - loss: 0.4617 - sparse_categorical_accuracy: 0.9932 - val_loss: 0.2118 - val_sparse_categorical_accuracy: 0.9968

Epoch 11/30

87/87 [=====] - 6s 64ms/step - loss: 1.9330 - sparse_categorical_accuracy: 0.9827 - val_loss: 7.3122e-04 - val_sparse_categorical_accuracy: 1.0000

Epoch 12/30

87/87 [=====] - 6s 62ms/step - loss: 1.5951 - sparse_categorical_accuracy: 0.9903 - val_loss: 0.4488 - val_sparse_categorical_accuracy: 0.9903

Epoch 13/30

```
87/87 [=====] - 6s 64ms/step - loss: 0.9665 - sparse_categorical_accuracy: 0.9903 - val_loss: 0.0205 - val_sparse_categorical_accuracy: 0.9935
Epoch 14/30
87/87 [=====] - 6s 64ms/step - loss: 0.1011 - sparse_categorical_accuracy: 0.9982 - val_loss: 0.4912 - val_sparse_categorical_accuracy: 0.9903
Epoch 15/30
87/87 [=====] - 7s 72ms/step - loss: 6.6320 - sparse_categorical_accuracy: 0.9795 - val_loss: 1.3248 - val_sparse_categorical_accuracy: 0.9903
Epoch 16/30
87/87 [=====] - 7s 72ms/step - loss: 1.2581 - sparse_categorical_accuracy: 0.9942 - val_loss: 0.1248 - val_sparse_categorical_accuracy: 0.9968
Epoch 17/30
87/87 [=====] - 7s 70ms/step - loss: 2.6191e-04 - sparse_categorical_accuracy: 0.9996 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 18/30
87/87 [=====] - 8s 81ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 19/30
87/87 [=====] - 8s 85ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 20/30
87/87 [=====] - 7s 77ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 21/30
87/87 [=====] - 7s 78ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 22/30
87/87 [=====] - 8s 80ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 23/30
87/87 [=====] - 7s 71ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 24/30
87/87 [=====] - 7s 70ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 25/30
87/87 [=====] - 6s 68ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 26/30
87/87 [=====] - 6s 68ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 27/30
87/87 [=====] - 6s 67ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 28/30
87/87 [=====] - 6s 69ms/step - loss: 0.0000e+00 -
```

```
sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 29/30
87/87 [=====] - 7s 77ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
Epoch 30/30
87/87 [=====] - 7s 75ms/step - loss: 0.0000e+00 - sparse_categorical_accuracy: 1.0000 - val_loss: 0.0515 - val_sparse_categorical_accuracy: 0.9968
```

In [14]:

```
# Training Model 2
history2 = model_2.fit(
    train_dataset,
    epochs=30,
    shuffle=True,
    validation_data = (validation_dataset)
)
```

Epoch 1/30

```
87/87 [=====] - 16s 139ms/step - loss: 2.5268 - sparse_categorical_accuracy: 0.3691 - val_loss: 0.8988 - val_sparse_categorical_accuracy: 0.9123
```

Epoch 2/30

```
87/87 [=====] - 10s 110ms/step - loss: 0.2476 - sparse_categorical_accuracy: 0.9342 - val_loss: 23.5757 - val_sparse_categorical_accuracy: 0.0779
```

Epoch 3/30

```
87/87 [=====] - 10s 108ms/step - loss: 0.0808 - sparse_categorical_accuracy: 0.9770 - val_loss: 23.5948 - val_sparse_categorical_accuracy: 0.0455
```

Epoch 4/30

```
87/87 [=====] - 10s 110ms/step - loss: 0.0762 - sparse_categorical_accuracy: 0.9773 - val_loss: 25.0027 - val_sparse_categorical_accuracy: 0.0357
```

Epoch 5/30

```
87/87 [=====] - 10s 111ms/step - loss: 0.0731 - sparse_categorical_accuracy: 0.9795 - val_loss: 21.1595 - val_sparse_categorical_accuracy: 0.0357
```

Epoch 6/30

```
87/87 [=====] - 11s 115ms/step - loss: 0.0672 - sparse_categorical_accuracy: 0.9809 - val_loss: 21.5560 - val_sparse_categorical_accuracy: 0.0357
```

Epoch 7/30

```
87/87 [=====] - 10s 111ms/step - loss: 0.0457 - sparse_categorical_accuracy: 0.9881 - val_loss: 17.4330 - val_sparse_categorical_accuracy: 0.0487
```

Epoch 8/30

```
87/87 [=====] - 10s 110ms/step - loss: 0.0443 - sparse_categorical_accuracy: 0.9867 - val_loss: 27.1873 - val_sparse_categorical_accuracy: 0.0357
```

Epoch 9/30

```
87/87 [=====] - 10s 111ms/step - loss: 0.0298 - sparse_categorical_accuracy: 0.9932 - val_loss: 28.6865 - val_sparse_categorical_accuracy: 0.0487
```

Epoch 10/30

```
87/87 [=====] - 10s 110ms/step - loss: 0.0499 - sparse_categorical_accuracy: 0.9842 - val_loss: 32.7295 - val_sparse_categorical_accuracy: 0.0487
```

Epoch 11/30

```
87/87 [=====] - 10s 111ms/step - loss: 0.0349 - sparse_categorical_accuracy: 0.9910 - val_loss: 28.8760 - val_sparse_categorical_accuracy: 0.0357
```

Epoch 12/30

```
87/87 [=====] - 10s 112ms/step - loss: 0.0231 - sparse_categorical_accuracy: 0.9939 - val_loss: 27.8960 - val_sparse_categorical_accuracy: 0.0487
```

Epoch 13/30

```
87/87 [=====] - 10s 112ms/step - loss: 0.0249 - sparse_categorical_accuracy: 0.9939 - val_loss: 27.8960 - val_sparse_categorical_accuracy: 0.0487
```

```
rse_categorical_accuracy: 0.9914 - val_loss: 32.0158 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 14/30
87/87 [=====] - 10s 111ms/step - loss: 0.0182 - spa
rse_categorical_accuracy: 0.9932 - val_loss: 36.6200 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 15/30
87/87 [=====] - 10s 114ms/step - loss: 0.0438 - spa
rse_categorical_accuracy: 0.9885 - val_loss: 26.5402 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 16/30
87/87 [=====] - 10s 112ms/step - loss: 0.0231 - spa
rse_categorical_accuracy: 0.9921 - val_loss: 26.7868 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 17/30
87/87 [=====] - 10s 110ms/step - loss: 0.0131 - spa
rse_categorical_accuracy: 0.9950 - val_loss: 31.2467 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 18/30
87/87 [=====] - 10s 113ms/step - loss: 0.0270 - spa
rse_categorical_accuracy: 0.9921 - val_loss: 28.0671 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 19/30
87/87 [=====] - 11s 114ms/step - loss: 0.0313 - spa
rse_categorical_accuracy: 0.9906 - val_loss: 31.5852 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 20/30
87/87 [=====] - 10s 112ms/step - loss: 0.0251 - spa
rse_categorical_accuracy: 0.9935 - val_loss: 32.7308 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 21/30
87/87 [=====] - 10s 114ms/step - loss: 0.0588 - spa
rse_categorical_accuracy: 0.9867 - val_loss: 39.3150 - val_sparse_categorical_
l_accuracy: 0.0487
Epoch 22/30
87/87 [=====] - 11s 114ms/step - loss: 0.0476 - spa
rse_categorical_accuracy: 0.9878 - val_loss: 30.4869 - val_sparse_categorical_
l_accuracy: 0.0422
Epoch 23/30
87/87 [=====] - 10s 114ms/step - loss: 0.0246 - spa
rse_categorical_accuracy: 0.9939 - val_loss: 37.6322 - val_sparse_categorical_
l_accuracy: 0.0357
Epoch 24/30
87/87 [=====] - 10s 110ms/step - loss: 0.0174 - spa
rse_categorical_accuracy: 0.9950 - val_loss: 41.3449 - val_sparse_categorical_
l_accuracy: 0.0357
Epoch 25/30
87/87 [=====] - 10s 109ms/step - loss: 0.0386 - spa
rse_categorical_accuracy: 0.9928 - val_loss: 41.1448 - val_sparse_categorical_
l_accuracy: 0.0357
Epoch 26/30
87/87 [=====] - 7s 82ms/step - loss: 0.0343 - spars
e_categorical_accuracy: 0.9917 - val_loss: 40.6883 - val_sparse_categorical_
accuracy: 0.0487
Epoch 27/30
87/87 [=====] - 8s 84ms/step - loss: 0.0172 - spars
e_categorical_accuracy: 0.9950 - val_loss: 45.6515 - val_sparse_categorical_
accuracy: 0.0649
Epoch 28/30
87/87 [=====] - 8s 84ms/step - loss: 0.0388 - spars
e_categorical_accuracy: 0.9881 - val_loss: 40.4782 - val_sparse_categorical_
```

accuracy: 0.0422

Epoch 29/30

87/87 [=====] - 10s 113ms/step - loss: 0.0378 - sparse_categorical_accuracy: 0.9910 - val_loss: 42.0297 - val_sparse_categorical_accuracy: 0.0487

Epoch 30/30

87/87 [=====] - 10s 109ms/step - loss: 0.0302 - sparse_categorical_accuracy: 0.9924 - val_loss: 34.0295 - val_sparse_categorical_accuracy: 0.0649

In [15]:

```
# Training Model 3
history3 = model_3.fit(
    train_dataset,
    epochs=30,
    shuffle=True,
    validation_data = (validation_dataset)
)
```

Epoch 1/30

```
87/87 [=====] - 34s 273ms/step - loss: 2.0481 - sparse_categorical_accuracy: 0.3917 - val_loss: 3.6334 - val_sparse_categorical_accuracy: 0.2468
```

Epoch 2/30

```
87/87 [=====] - 13s 137ms/step - loss: 0.4149 - sparse_categorical_accuracy: 0.8651 - val_loss: 38.4023 - val_sparse_categorical_accuracy: 0.2208
```

Epoch 3/30

```
87/87 [=====] - 13s 138ms/step - loss: 0.2008 - sparse_categorical_accuracy: 0.9450 - val_loss: 25.1020 - val_sparse_categorical_accuracy: 0.1461
```

Epoch 4/30

```
87/87 [=====] - 13s 139ms/step - loss: 0.1757 - sparse_categorical_accuracy: 0.9478 - val_loss: 2.5688 - val_sparse_categorical_accuracy: 0.6006
```

Epoch 5/30

```
87/87 [=====] - 13s 138ms/step - loss: 0.0963 - sparse_categorical_accuracy: 0.9737 - val_loss: 5.8490 - val_sparse_categorical_accuracy: 0.3604
```

Epoch 6/30

```
87/87 [=====] - 13s 139ms/step - loss: 0.1112 - sparse_categorical_accuracy: 0.9669 - val_loss: 0.2579 - val_sparse_categorical_accuracy: 0.9318
```

Epoch 7/30

```
87/87 [=====] - 13s 138ms/step - loss: 0.1061 - sparse_categorical_accuracy: 0.9683 - val_loss: 9.2605 - val_sparse_categorical_accuracy: 0.2597
```

Epoch 8/30

```
87/87 [=====] - 12s 136ms/step - loss: 0.0749 - sparse_categorical_accuracy: 0.9777 - val_loss: 0.8603 - val_sparse_categorical_accuracy: 0.7175
```

Epoch 9/30

```
87/87 [=====] - 13s 141ms/step - loss: 0.0574 - sparse_categorical_accuracy: 0.9853 - val_loss: 0.6230 - val_sparse_categorical_accuracy: 0.8377
```

Epoch 10/30

```
87/87 [=====] - 13s 140ms/step - loss: 0.0516 - sparse_categorical_accuracy: 0.9842 - val_loss: 8.5928 - val_sparse_categorical_accuracy: 0.3409
```

Epoch 11/30

```
87/87 [=====] - 13s 139ms/step - loss: 0.0507 - sparse_categorical_accuracy: 0.9849 - val_loss: 14.6717 - val_sparse_categorical_accuracy: 0.3149
```

Epoch 12/30

```
87/87 [=====] - 13s 138ms/step - loss: 0.0579 - sparse_categorical_accuracy: 0.9845 - val_loss: 14.6992 - val_sparse_categorical_accuracy: 0.2175
```

Epoch 13/30

```
87/87 [=====] - 13s 141ms/step - loss: 0.0322 - sparse_categorical_accuracy: 0.9845 - val_loss: 14.6992 - val_sparse_categorical_accuracy: 0.2175
```



```
rse_categorical_accuracy: 0.9899 - val_loss: 2.1610 - val_sparse_categorical_accuracy: 0.6656
Epoch 14/30
87/87 [=====] - 12s 135ms/step - loss: 0.0225 - sparse_categorical_accuracy: 0.9921 - val_loss: 2.1411 - val_sparse_categorical_accuracy: 0.7013
Epoch 15/30
87/87 [=====] - 12s 135ms/step - loss: 0.0467 - sparse_categorical_accuracy: 0.9863 - val_loss: 14.7012 - val_sparse_categorical_accuracy: 0.2857
Epoch 16/30
87/87 [=====] - 12s 135ms/step - loss: 0.0461 - sparse_categorical_accuracy: 0.9853 - val_loss: 3.9097 - val_sparse_categorical_accuracy: 0.4318
Epoch 17/30
87/87 [=====] - 13s 141ms/step - loss: 0.0326 - sparse_categorical_accuracy: 0.9888 - val_loss: 1.6834 - val_sparse_categorical_accuracy: 0.6656
Epoch 18/30
87/87 [=====] - 13s 141ms/step - loss: 0.0275 - sparse_categorical_accuracy: 0.9924 - val_loss: 6.0157 - val_sparse_categorical_accuracy: 0.3279
Epoch 19/30
87/87 [=====] - 13s 142ms/step - loss: 0.0272 - sparse_categorical_accuracy: 0.9928 - val_loss: 23.4782 - val_sparse_categorical_accuracy: 0.2890
Epoch 20/30
87/87 [=====] - 13s 137ms/step - loss: 0.0462 - sparse_categorical_accuracy: 0.9845 - val_loss: 0.0415 - val_sparse_categorical_accuracy: 0.9870
Epoch 21/30
87/87 [=====] - 13s 140ms/step - loss: 0.0323 - sparse_categorical_accuracy: 0.9899 - val_loss: 15.0468 - val_sparse_categorical_accuracy: 0.1916
Epoch 22/30
87/87 [=====] - 13s 140ms/step - loss: 0.0200 - sparse_categorical_accuracy: 0.9939 - val_loss: 20.5498 - val_sparse_categorical_accuracy: 0.3084
Epoch 23/30
87/87 [=====] - 13s 146ms/step - loss: 0.0256 - sparse_categorical_accuracy: 0.9942 - val_loss: 5.4647 - val_sparse_categorical_accuracy: 0.4416
Epoch 24/30
87/87 [=====] - 10s 113ms/step - loss: 0.0430 - sparse_categorical_accuracy: 0.9860 - val_loss: 9.5315 - val_sparse_categorical_accuracy: 0.3929
Epoch 25/30
87/87 [=====] - 10s 106ms/step - loss: 0.0416 - sparse_categorical_accuracy: 0.9871 - val_loss: 9.7630 - val_sparse_categorical_accuracy: 0.3377
Epoch 26/30
87/87 [=====] - 10s 113ms/step - loss: 0.0204 - sparse_categorical_accuracy: 0.9932 - val_loss: 28.4930 - val_sparse_categorical_accuracy: 0.1299
Epoch 27/30
87/87 [=====] - 13s 137ms/step - loss: 0.0144 - sparse_categorical_accuracy: 0.9950 - val_loss: 10.6105 - val_sparse_categorical_accuracy: 0.3831
Epoch 28/30
87/87 [=====] - 13s 138ms/step - loss: 0.0156 - sparse_categorical_accuracy: 0.9950 - val_loss: 26.9795 - val_sparse_categorical_accuracy: 0.3831
```

```
l_accuracy: 0.2013
Epoch 29/30
87/87 [=====] - 12s 137ms/step - loss: 0.0317 - sparse_categorical_accuracy: 0.9924 - val_loss: 4.2126 - val_sparse_categorical_accuracy: 0.4675
Epoch 30/30
87/87 [=====] - 13s 138ms/step - loss: 0.0125 - sparse_categorical_accuracy: 0.9942 - val_loss: 0.0127 - val_sparse_categorical_accuracy: 0.9968
```

Save Model

In [29]:

```
modelFileDirectoryName1 = 'saved_model/Model_1'
modelFileDirectoryName2 = 'saved_model/Model_2'
modelFileDirectoryName3 = 'saved_model/Model_3'
model_1.save(modelFileDirectoryName1)
model_2.save(modelFileDirectoryName2)
model_3.save(modelFileDirectoryName3)
```

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 3 of 3). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/Model_1\assets

INFO:tensorflow:Assets written to: saved_model/Model_1\assets

WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 8). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/Model_2\assets

INFO:tensorflow:Assets written to: saved_model/Model_2\assets

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

INFO:tensorflow:Assets written to: saved_model/Model_3\assets

INFO:tensorflow:Assets written to: saved_model/Model_3\assets

Model Evaluation and History

In [36]:

```
# Model 1
loss, acc = model_1.evaluate(test_dataset)
print("Test accuracy: {:.2f}%".format(100 * acc))
```

```
10/10 [=====] - 1s 36ms/step - loss: 5.3646 - sparse_categorical_accuracy: 0.9583
Test accuracy: 95.83%
```

In [40]:

```
# Model 2
loss, acc = model_2.evaluate(test_dataset)
print("Test accuracy: {:.5.2f}%".format(100 * acc))
```

10/10 [=====] - 1s 18ms/step - loss: 33.6375 - spar
se_categorical_accuracy: 0.0609
Test accuracy: 6.09%

In [39]:

```
# Model 3
loss, acc = model_3.evaluate(test_dataset)
print("Test accuracy: {:.5.2f}%".format(100 * acc))
```

10/10 [=====] - 1s 26ms/step - loss: 0.0051 - spars
e_categorical_accuracy: 1.0000
Test accuracy: 100.00%

In []: