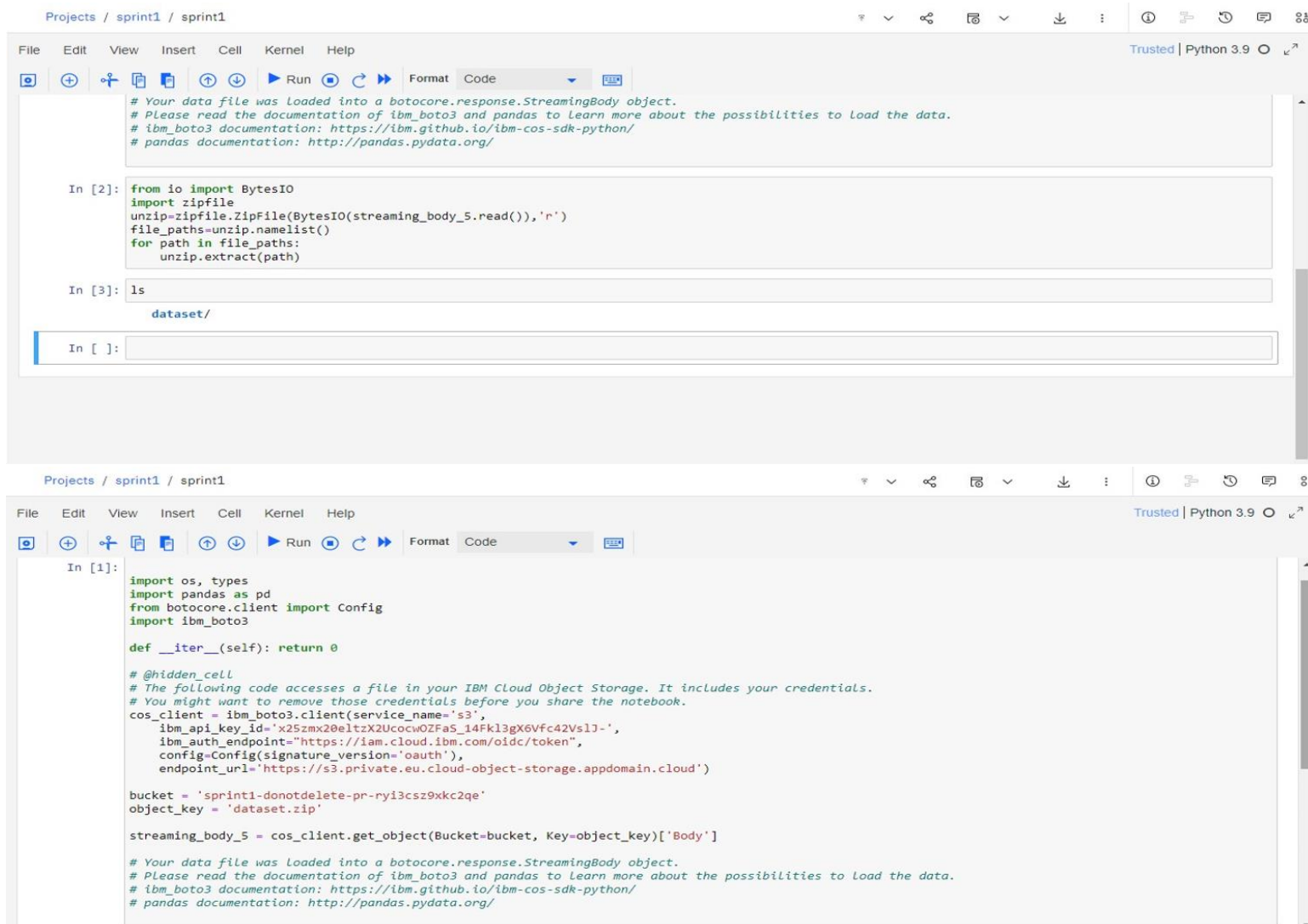


Project Development Phase Sprint - 3

Date	13 November 2022
Team ID	PNT2022TMID03497
Project Name	A Gesture - Based Tool for Sterile Browsing of Radiology Images
Marks	4 Marks

IBM Watson Studio:

Sprint 1:



The screenshot displays the IBM Watson Studio interface for a Jupyter notebook. The top navigation bar shows 'Projects / sprint1 / sprint1'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and formatting. The notebook contains two code cells. The first cell, labeled 'In [2]:', contains a comment about loading data into a boto3 response object and a code snippet that imports BytesIO and zipfile, then extracts a zip file from a streaming body. The second cell, labeled 'In [3]:', contains a simple 'ls' command. The output of the second cell shows the directory structure: 'dataset/'. The bottom screenshot shows the first cell, 'In [1]:', which contains a more complex code snippet. This snippet imports os, types, pandas, boto3, and Config. It defines a class with an __iter__ method and a @hidden_cell decorator. The code then initializes a boto3 client for IBM Cloud Object Storage, sets up credentials, and retrieves a specific object from a bucket named 'sprint1-donotdelete-pr-ryi3csz9xkc2qe' with the key 'dataset.zip'. It then loads the object into a streaming body and provides a comment about loading data into a boto3 response object.

```
Projects / sprint1 / sprint1

File Edit View Insert Cell Kernel Help Trusted | Python 3.9

In [2]: # Your data file was loaded into a boto3.response.StreamingBody object.
import zipfile
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_5.read()), 'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)

In [3]: ls

dataset/

In [ ]:
```

```
Projects / sprint1 / sprint1

File Edit View Insert Cell Kernel Help Trusted | Python 3.9

In [1]: import os, types
import pandas as pd
from boto3.client import Config
import ibm_boto3

def __iter__(self): return 0

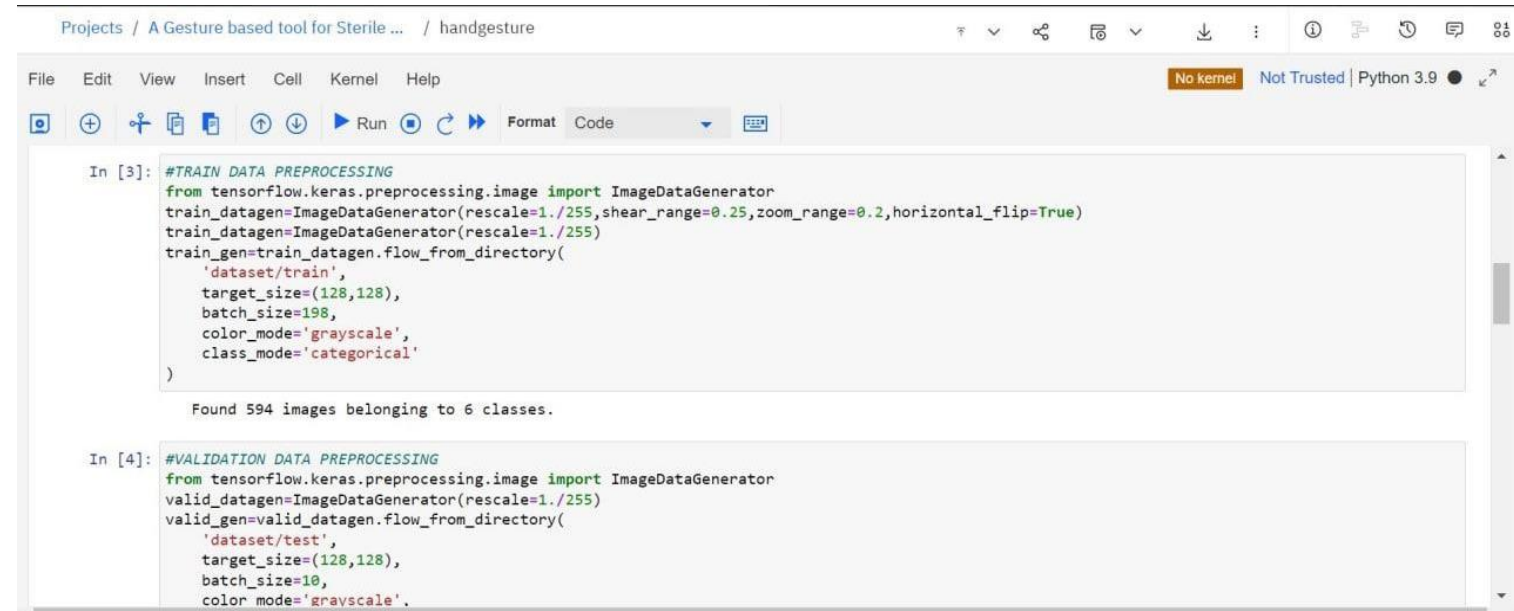
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='x25zmx20eltzX2Uocw0ZFaS_14Fk13gX6Vfc42VslJ-',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

bucket = 'sprint1-donotdelete-pr-ryi3csz9xkc2qe'
object_key = 'dataset.zip'

streaming_body_5 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a boto3.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

Sprint 2:



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and formatting. The notebook is titled "Projects / A Gesture based tool for Sterile ... / handgesture". The code is written in Python 3.9. The first cell (In [3]) contains the following code:

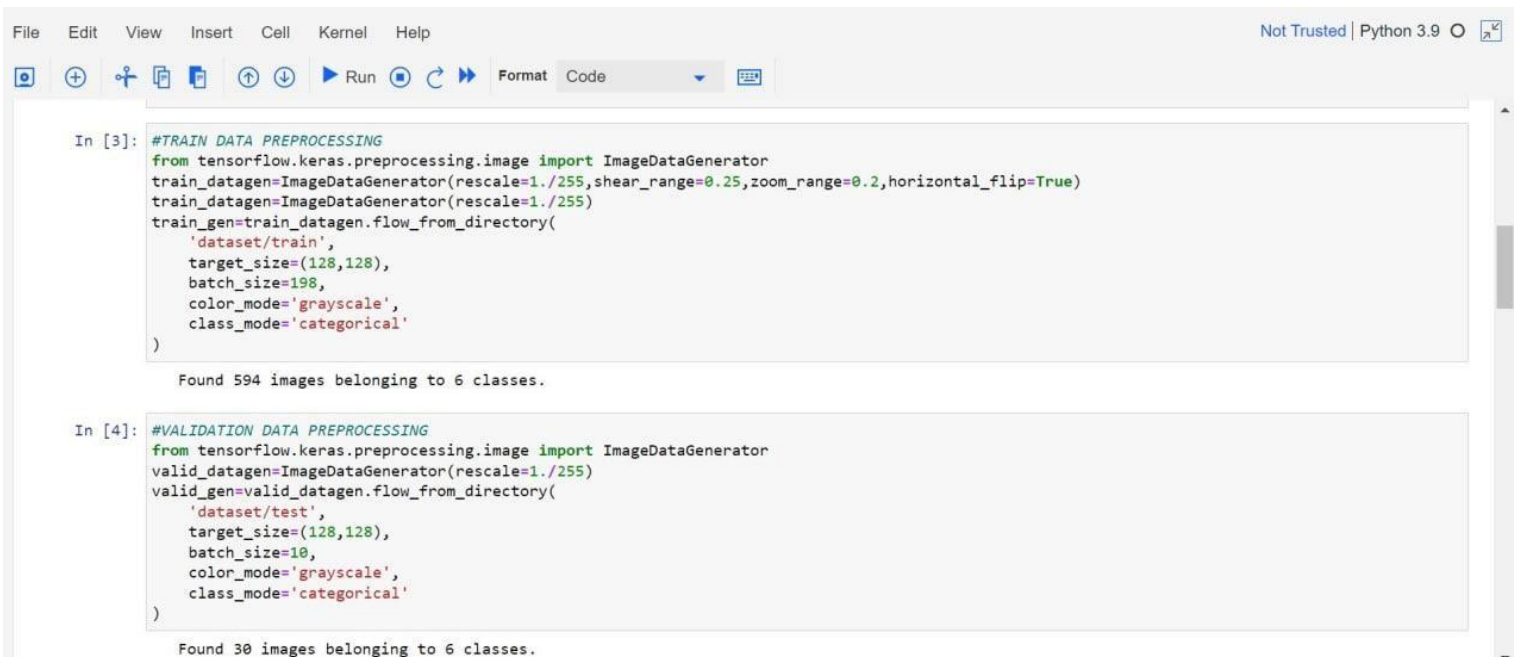
```
#TRAIN DATA PREPROCESSING
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.25,zoom_range=0.2,horizontal_flip=True)
train_datagen=ImageDataGenerator(rescale=1./255)
train_gen=train_datagen.flow_from_directory(
    'dataset/train',
    target_size=(128,128),
    batch_size=198,
    color_mode='grayscale',
    class_mode='categorical'
)
```

The output of the first cell is:

```
Found 594 images belonging to 6 classes.
```

The second cell (In [4]) contains the following code:

```
#VALIDATION DATA PREPROCESSING
from tensorflow.keras.preprocessing.image import ImageDataGenerator
valid_datagen=ImageDataGenerator(rescale=1./255)
valid_gen=valid_datagen.flow_from_directory(
    'dataset/test',
    target_size=(128,128),
    batch_size=10,
    color_mode='grayscale',
    class_mode='categorical'
)
```



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and formatting. The notebook is titled "Projects / A Gesture based tool for Sterile ... / handgesture". The code is written in Python 3.9. The first cell (In [3]) contains the following code:

```
#TRAIN DATA PREPROCESSING
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.25,zoom_range=0.2,horizontal_flip=True)
train_datagen=ImageDataGenerator(rescale=1./255)
train_gen=train_datagen.flow_from_directory(
    'dataset/train',
    target_size=(128,128),
    batch_size=198,
    color_mode='grayscale',
    class_mode='categorical'
)
```

The output of the first cell is:

```
Found 594 images belonging to 6 classes.
```

The second cell (In [4]) contains the following code:

```
#VALIDATION DATA PREPROCESSING
from tensorflow.keras.preprocessing.image import ImageDataGenerator
valid_datagen=ImageDataGenerator(rescale=1./255)
valid_gen=valid_datagen.flow_from_directory(
    'dataset/test',
    target_size=(128,128),
    batch_size=10,
    color_mode='grayscale',
    class_mode='categorical'
)
```

The output of the second cell is:

```
Found 30 images belonging to 6 classes.
```

```
File Edit View Insert Cell Kernel Help Not Trusted | Python 3.9
[+] [-] [Run] [Format] [Code] [Exit fullscreen]

In [5]: #CONVOLUTION NEURAL NETWORK MODEL
import tensorflow as tf
print(tf.__version__)
model=tf.keras.Sequential([
    tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(128,128,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(16,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.Dense(6,activation='softmax')
])

2.7.2

In [6]: #LOSS FUNCTION AND OPTIMIZER
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['Accuracy'])

In [7]: model.summary()

Model: "sequential"
Layer (type) Output Shape Param #
-----
conv2d (Conv2D) (None, 126, 126, 16) 160
```

```
File Edit View Insert Cell Kernel Help Not Trusted | Python 3.9
[+] [-] [Run] [Format] [Code] [Exit fullscreen]

Model: "sequential"
Layer (type) Output Shape Param #
-----
conv2d (Conv2D) (None, 126, 126, 16) 160
max_pooling2d (MaxPooling2D) (None, 63, 63, 16) 0
conv2d_1 (Conv2D) (None, 61, 61, 32) 4640
max_pooling2d_1 (MaxPooling2D) (None, 30, 30, 32) 0
conv2d_2 (Conv2D) (None, 28, 28, 16) 4624
max_pooling2d_2 (MaxPooling2D) (None, 14, 14, 16) 0
flatten (Flatten) (None, 3136) 0
dense (Dense) (None, 512) 1606144
dense_1 (Dense) (None, 6) 3078
=====
Total params: 1,618,646
Trainable params: 1,618,646
```

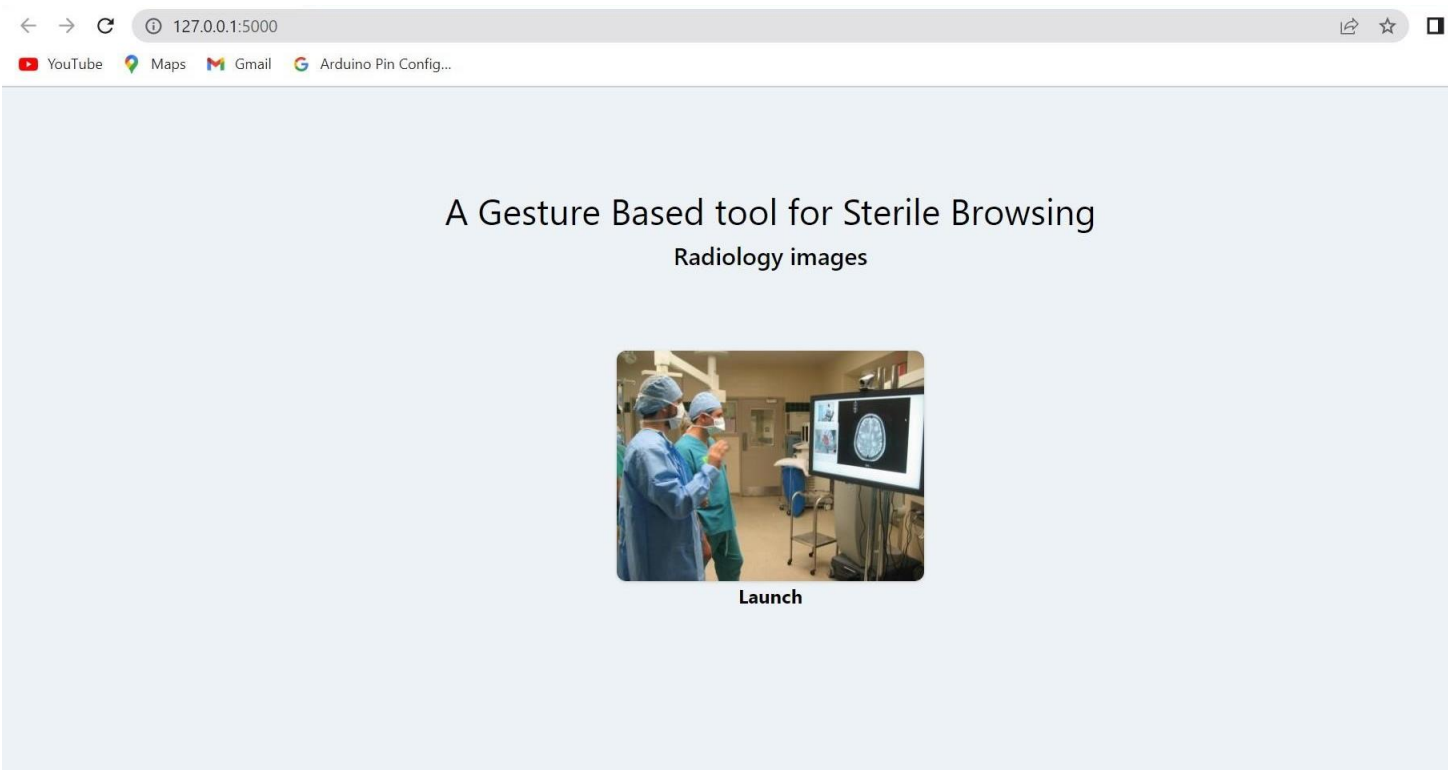
```
In [8]: #TRAINING THE MODEL
trainmodel=model.fit(
    train_gen,
    steps_per_epoch=3,
    epochs=20,
    validation_data=valid_gen,
    validation_steps=3,
)
```

```
Epoch 1/20
3/3 [=====] - 6s 2s/step - loss: 1.7762 - Accuracy: 0.2003 - val_loss: 1.6875 - val_Accuracy: 0.2667
Epoch 2/20
3/3 [=====] - 6s 2s/step - loss: 1.6189 - Accuracy: 0.4327 - val_loss: 1.4747 - val_Accuracy: 0.4000
Epoch 3/20
3/3 [=====] - 6s 2s/step - loss: 1.3585 - Accuracy: 0.6515 - val_loss: 1.2633 - val_Accuracy: 0.5667
Epoch 4/20
3/3 [=====] - 6s 2s/step - loss: 1.0766 - Accuracy: 0.6515 - val_loss: 0.9468 - val_Accuracy: 0.8333
Epoch 5/20
3/3 [=====] - 6s 2s/step - loss: 0.8148 - Accuracy: 0.7273 - val_loss: 0.9268 - val_Accuracy: 0.7000
Epoch 6/20
3/3 [=====] - 6s 2s/step - loss: 0.6414 - Accuracy: 0.7694 - val_loss: 0.6724 - val_Accuracy: 0.7667
Epoch 7/20
3/3 [=====] - 6s 2s/step - loss: 0.4723 - Accuracy: 0.8519 - val_loss: 0.6585 - val_Accuracy: 0.6667
Epoch 8/20
3/3 [=====] - 6s 2s/step - loss: 0.3796 - Accuracy: 0.8636 - val_loss: 0.5784 - val_Accuracy: 0.8667
Epoch 9/20
3/3 [=====] - 6s 2s/step - loss: 0.2983 - Accuracy: 0.9226 - val_loss: 0.7214 - val_Accuracy: 0.7333
```

```
Epoch 9/20
3/3 [=====] - 6s 2s/step - loss: 0.2983 - Accuracy: 0.9226 - val_loss: 0.7214 - val_Accuracy: 0.7333
Epoch 10/20
3/3 [=====] - 6s 2s/step - loss: 0.2561 - Accuracy: 0.9242 - val_loss: 0.5249 - val_Accuracy: 0.8667
Epoch 11/20
3/3 [=====] - 6s 2s/step - loss: 0.2160 - Accuracy: 0.9276 - val_loss: 0.6461 - val_Accuracy: 0.7667
Epoch 12/20
3/3 [=====] - 6s 2s/step - loss: 0.1949 - Accuracy: 0.9293 - val_loss: 0.7166 - val_Accuracy: 0.7667
Epoch 13/20
3/3 [=====] - 6s 2s/step - loss: 0.1745 - Accuracy: 0.9461 - val_loss: 0.6339 - val_Accuracy: 0.8333
Epoch 14/20
3/3 [=====] - 6s 2s/step - loss: 0.1319 - Accuracy: 0.9613 - val_loss: 0.5087 - val_Accuracy: 0.9000
Epoch 15/20
3/3 [=====] - 6s 2s/step - loss: 0.1055 - Accuracy: 0.9663 - val_loss: 0.7516 - val_Accuracy: 0.8000
Epoch 16/20
3/3 [=====] - 6s 2s/step - loss: 0.0914 - Accuracy: 0.9731 - val_loss: 0.5083 - val_Accuracy: 0.9000
Epoch 17/20
3/3 [=====] - 6s 2s/step - loss: 0.0810 - Accuracy: 0.9764 - val_loss: 0.7712 - val_Accuracy: 0.8333
Epoch 18/20
3/3 [=====] - 6s 2s/step - loss: 0.0508 - Accuracy: 0.9899 - val_loss: 0.5598 - val_Accuracy: 0.9000
```

Sprint 3:

Home Page:



Main page:

