

```
loaded_model = tf.keras.models.load_model(  
    'Model.h5',  
    custom_objects={'GlobalContextAttention': GlobalContextAttention}  
)  
print("Model loaded successfully.")  
except Exception as e:  
    print(f"Error loading the model: {e}")
```

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the Model loaded successfully.

```
[ ] import numpy as np  
    from sklearn.metrics import accuracy_score  
  
    # Get true labels from test generator  
    true_labels = test_gen.classes # Actual labels of test images  
  
    # Predict the classes  
    y_pred_probs = model.predict(test_gen, steps=test_steps)  
    y_pred = np.argmax(y_pred_probs, axis=1) # Convert probabilities to class indices  
  
    # Compute image-level accuracy  
    image_level_accuracy = accuracy_score(true_labels, y_pred)  
  
    print(f"Image Level Accuracy: {image_level_accuracy:.4f}")
```

50/50 14s 279ms/step
Image Level Accuracy: 0.9900

```
[ ] !kaggle datasets download -d andrewmvd/lung-and-colon-cancer-histopathological-images

Dataset URL: https://www.kaggle.com/datasets/andrewmvd/lung-and-colon-cancer-histopathological-images
License(s): CC-BY-SA-4.0
Downloading lung-and-colon-cancer-histopathological-images.zip to /content
100% 1.76G/1.76G [00:55<00:00, 35.4MB/s]
100% 1.76G/1.76G [00:55<00:00, 34.1MB/s]

[ ] import zipfile

with zipfile.ZipFile('lung-and-colon-cancer-histopathological-images.zip', 'r') as zip_ref:
    zip_ref.extractall('lung-and-colon-cancer-histopathological-images')

!ls lung-and-colon-cancer-histopathological-images

lung_colon_image_set

[ ] # import system libs
import os
import time
```

```
import time
import shutil
import pathlib
import itertools
from PIL import Image

# import data handling tools
import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# import Deep learning Libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras import regularizers

# Ignore Warnings
```



```
[ ] # Ignore Warnings
import warnings
warnings.filterwarnings("ignore")

print ('modules loaded')
```

modules loaded

```
# Generate data paths with labels
data_dir = '/content/lung-and-colon-cancer-histopathological-images/lung_colon_image_set'
filepaths = []
labels = []

folds = os.listdir(data_dir)
for fold in folds:
    foldpath = os.path.join(data_dir, fold)
    flist = os.listdir(foldpath)

    for f in flist:
        f_path = os.path.join(foldpath, f)
        filelist = os.listdir(f_path)

        for file in filelist:
            fpath = os.path.join(f_path, file)
            filepaths.append(fpath)
```

```
filepaths.append(fpath)

if f == 'colon_aca':
    labels.append('Colon Adenocarcinoma')

elif f == 'colon_n':
    labels.append('Colon Benign Tissue')

elif f == 'lung_aca':
    labels.append('Lung Adenocarcinoma')

elif f == 'lung_n':
    labels.append('Lung Benign Tissue')

elif f == 'lung_scc':
    labels.append('Lung Squamous Cell Carcinoma')

# Concatenate data paths with labels into one dataframe
Fseries = pd.Series(filepaths, name= 'filepaths')
Lseries = pd.Series(labels, name='labels')
df = pd.concat([Fseries, Lseries], axis= 1)
```

[] df

df

	filepaths	labels
0	/content/lung-and-colon-cancer-histopathologic...	Lung Adenocarcinoma
1	/content/lung-and-colon-cancer-histopathologic...	Lung Adenocarcinoma
2	/content/lung-and-colon-cancer-histopathologic...	Lung Adenocarcinoma
3	/content/lung-and-colon-cancer-histopathologic...	Lung Adenocarcinoma
4	/content/lung-and-colon-cancer-histopathologic...	Lung Adenocarcinoma
...
24995	/content/lung-and-colon-cancer-histopathologic...	Colon Benign Tissue
24996	/content/lung-and-colon-cancer-histopathologic...	Colon Benign Tissue
24997	/content/lung-and-colon-cancer-histopathologic...	Colon Benign Tissue
24998	/content/lung-and-colon-cancer-histopathologic...	Colon Benign Tissue
24999	/content/lung-and-colon-cancer-histopathologic...	Colon Benign Tissue

25000 rows x 2 columns


```
strat = df['labels']
train_df, dummy_df = train_test_split(df, train_size= 0.8, shuffle= True, random_state= 123, stratify= strat)

# valid and test dataframe
strat = dummy_df['labels']
valid_df, test_df = train_test_split(dummy_df, train_size= 0.5, shuffle= True, random_state= 123, stratify= strat)

# crobed image size
batch_size = 64
img_size = (224, 224)
channels = 3
img_shape = (img_size[0], img_size[1], channels)

tr_gen = ImageDataGenerator()
ts_gen = ImageDataGenerator()

train_gen = tr_gen.flow_from_dataframe( train_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                         color_mode= 'rgb', shuffle= True, batch_size= batch_size)

valid_gen = ts_gen.flow_from_dataframe( valid_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                         color_mode= 'rgb', shuffle= True, batch_size= batch_size)

test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                       color_mode= 'rgb', shuffle= False, batch_size= batch_size)
```

```
color_mode= 'rgb', shuffle= True, batch_size= batch_size)

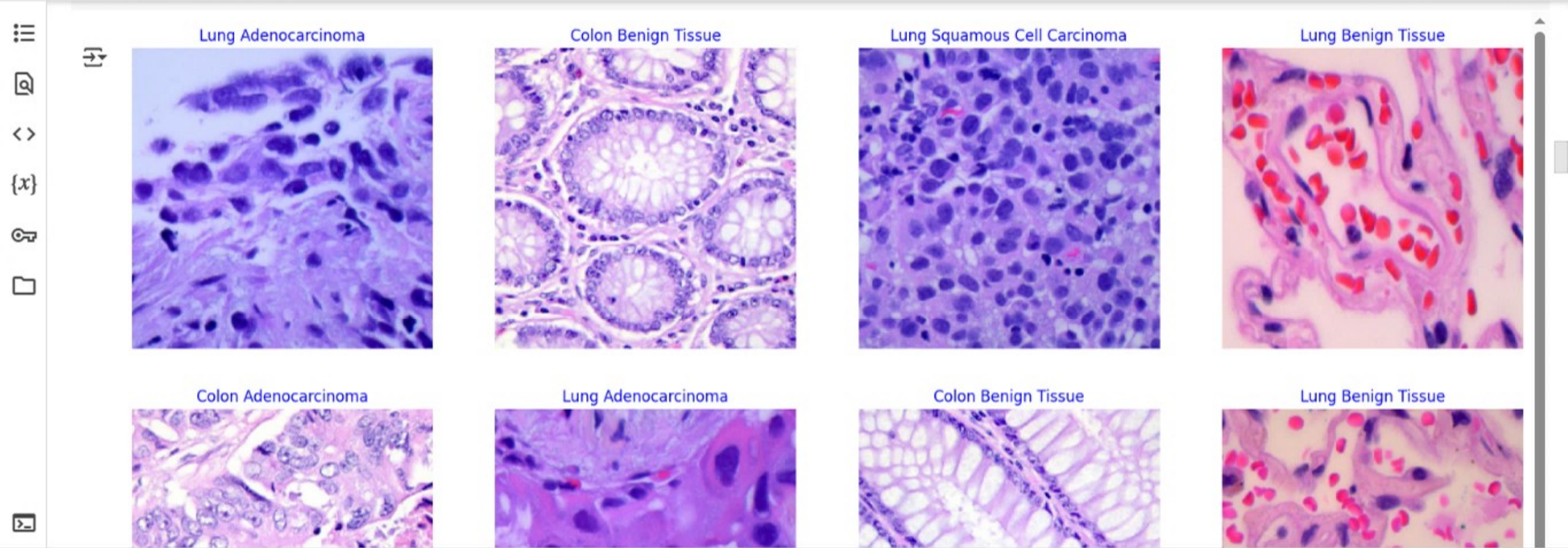
test_gen = ts_gen.flow_from_dataframe( test_df, x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical',
                                     color_mode= 'rgb', shuffle= False, batch_size= batch_size)
```

Found 20000 validated image filenames belonging to 5 classes.
Found 2500 validated image filenames belonging to 5 classes.
Found 2500 validated image filenames belonging to 5 classes.

```
g_dict = train_gen.class_indices
classes = list(g_dict.keys())
images, labels = next(train_gen)

plt.figure(figsize= (20, 20))

for i in range(16):
    plt.subplot(4, 4, i + 1)
    image = images[i] / 255
    plt.imshow(image)
    index = np.argmax(labels[i])
    class_name = classes[index]
    plt.title(class_name, color= 'blue', fontsize= 12)
    plt.axis('off')
plt.show()
```


```
import tensorflow as tf
from tensorflow.keras.layers import Layer, GlobalAveragePooling2D, Dense, Multiply, Reshape, Conv2D, MaxPooling2D, Flatten, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

# Global Context Attention Layer
class GlobalContextAttention(Layer):
    def __init__(self, channels, **kwargs):
        super(GlobalContextAttention, self).__init__(**kwargs)
        self.channels = channels
        self.dense1 = Dense(channels // 16, activation='relu') # Reduction
        self.dense2 = Dense(channels, activation='sigmoid') # Scaling

    def call(self, inputs):
        x = GlobalAveragePooling2D()(inputs)
        x = Reshape((1, 1, self.channels))(x)
        x = self.dense1(x)
        x = self.dense2(x)
        return Multiply()([inputs, x])

    def get_config(self):
        config = super().get_config()
        config.update({"channels": self.channels})
        return config
```

```
# This is CNN Model with Global Context Attention
model = Sequential([
    # Early Layers (No GC Attention)
    Conv2D(64, (3,3), padding="same", activation="relu", input_shape=img_shape),
    Conv2D(64, (3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3,3), padding="same", activation="relu"),
    Conv2D(128, (3,3), padding="same", activation="relu"),
    MaxPooling2D((2, 2)),

    # GC Attention only in deeper layers
    Conv2D(256, (3,3), padding="same", activation="relu"),
    Conv2D(256, (3,3), padding="same", activation="relu"),
    GlobalContextAttention(256), # GC applied here
    MaxPooling2D((2, 2)),

    Conv2D(512, (3,3), padding="same", activation="relu"),
    Conv2D(512, (3,3), padding="same", activation="relu"),
    GlobalContextAttention(512), # GC applied here
    MaxPooling2D((2, 2)),

    Conv2D(512, (3,3), padding="same", activation="relu"),
    Conv2D(512, (3,3), padding="same", activation="relu"),
```



```
[ ] GlobalContextAttention(512), # GC applied here
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(256, activation="relu"),
    Dropout(0.4), #dropout for regularization
    Dense(64, activation="relu"),
    Dense(5, activation="softmax")
])

model.compile(optimizer=Adam(learning_rate=0.0005), loss='categorical_crossentropy', metrics=['accuracy'])

# model summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_7 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d_2 (MaxPooling2D)	(None, 112, 112, 64)	0

<div>≡</div> <div>🔍</div> <div><></div> <div>{x}</div> <div>🔑</div> <div>📁</div> <div>🖨</div>	↕	conv2d_8 (Conv2D)	(None, 112, 112, 128)	73,856
		conv2d_9 (Conv2D)	(None, 112, 112, 128)	147,584
		max_pooling2d_3 (MaxPooling2D)	(None, 56, 56, 128)	0
		conv2d_10 (Conv2D)	(None, 56, 56, 256)	295,168
		conv2d_11 (Conv2D)	(None, 56, 56, 256)	590,080
		global_context_attention (GlobalContextAttention)	(None, 56, 56, 256)	8,464
		max_pooling2d_4 (MaxPooling2D)	(None, 28, 28, 256)	0
		conv2d_12 (Conv2D)	(None, 28, 28, 512)	1,180,160
		conv2d_13 (Conv2D)	(None, 28, 28, 512)	2,359,808
		global_context_attention_1 (GlobalContextAttention)	(None, 28, 28, 512)	33,312
		max_pooling2d_5 (MaxPooling2D)	(None, 14, 14, 512)	0
		conv2d_14 (Conv2D)	(None, 14, 14, 512)	2,359,808
		conv2d_15 (Conv2D)	(None, 14, 14, 512)	2,359,808

global_context_attention_2 (GlobalContextAttention)	(None, 14, 14, 512)	33,312
max_pooling2d_6 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense_6 (Dense)	(None, 256)	6,422,784
dropout (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 64)	16,448
dense_8 (Dense)	(None, 5)	325

Total params: 15,919,637 (60.73 MB)
Trainable params: 15,919,637 (60.73 MB)
Non-trainable params: 0 (0.00 B)

```
epochs = 20 # number of all epochs in training

history = model.fit(x= train_gen, epochs= epochs, verbose= 1, validation_data= test_gen,
                    validation_steps= None, shuffle= False)
```

Epoch 1/20

Epoch 1/20	313/313	362s	941ms/step	-	accuracy: 0.4007	-	loss: 1.2143	-	val_accuracy: 0.8596	-	val_loss: 0.3352
Epoch 2/20	313/313	231s	736ms/step	-	accuracy: 0.8581	-	loss: 0.3753	-	val_accuracy: 0.9180	-	val_loss: 0.2142
Epoch 3/20	313/313	230s	736ms/step	-	accuracy: 0.9079	-	loss: 0.2398	-	val_accuracy: 0.9572	-	val_loss: 0.1137
Epoch 4/20	313/313	230s	734ms/step	-	accuracy: 0.9409	-	loss: 0.1649	-	val_accuracy: 0.9576	-	val_loss: 0.1113
Epoch 5/20	313/313	231s	736ms/step	-	accuracy: 0.9514	-	loss: 0.1412	-	val_accuracy: 0.9760	-	val_loss: 0.0656
Epoch 6/20	313/313	262s	736ms/step	-	accuracy: 0.9650	-	loss: 0.0930	-	val_accuracy: 0.9332	-	val_loss: 0.1636
Epoch 7/20	313/313	230s	735ms/step	-	accuracy: 0.9678	-	loss: 0.0876	-	val_accuracy: 0.9804	-	val_loss: 0.0655
Epoch 8/20	313/313	230s	734ms/step	-	accuracy: 0.9749	-	loss: 0.0725	-	val_accuracy: 0.9708	-	val_loss: 0.0764
Epoch 9/20	313/313	263s	736ms/step	-	accuracy: 0.9788	-	loss: 0.0599	-	val_accuracy: 0.9768	-	val_loss: 0.0699
Epoch 10/20	313/313	229s	731ms/step	-	accuracy: 0.9838	-	loss: 0.0457	-	val_accuracy: 0.9696	-	val_loss: 0.0925
Epoch 11/20	313/313	229s	732ms/step	-	accuracy: 0.9862	-	loss: 0.0411	-	val_accuracy: 0.9880	-	val_loss: 0.0276
Epoch 12/20	313/313	230s	733ms/step	-	accuracy: 0.9828	-	loss: 0.0512	-	val_accuracy: 0.9908	-	val_loss: 0.0275
Epoch 13/20	313/313	230s	733ms/step	-	accuracy: 0.9894	-	loss: 0.0339	-	val_accuracy: 0.9896	-	val_loss: 0.0324
Epoch 14/20											

Copy of Untitled0.ipynb - Colab

colab.research.google.com/drive/1aZEFnOrLbcwHqskj8ES2ot1mf_z1plxs?usp=sharing#scrollTo=RHGKtpr8Ohk0

GmailYouTubeMapsSplit() String metho...

Copy of Untitled0.ipynb

FileEditViewInsertRuntimeToolsHelp

CommandsCodeText

Connect T4

Epoch 15/20
313/313 229s 732ms/step - accuracy: 0.9943 - loss: 0.0212 - val_accuracy: 0.9928 - val_loss: 0.0182
Epoch 16/20
313/313 229s 729ms/step - accuracy: 0.9938 - loss: 0.0216 - val_accuracy: 0.9928 - val_loss: 0.0212
Epoch 17/20
313/313 229s 731ms/step - accuracy: 0.9948 - loss: 0.0151 - val_accuracy: 0.9768 - val_loss: 0.0873
Epoch 18/20
313/313 262s 732ms/step - accuracy: 0.9882 - loss: 0.0418 - val_accuracy: 0.9760 - val_loss: 0.0594
Epoch 19/20
313/313 229s 732ms/step - accuracy: 0.9904 - loss: 0.0318 - val_accuracy: 0.9792 - val_loss: 0.0707
Epoch 20/20
313/313 262s 734ms/step - accuracy: 0.9916 - loss: 0.0307 - val_accuracy: 0.9900 - val_loss: 0.0316

Define needed variables
tr_acc = history.history['accuracy']
tr_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']
index_loss = np.argmin(val_loss)
val_lowest = val_loss[index_loss]
index_acc = np.argmax(val_acc)
acc_highest = val_acc[index_acc]
Epochs = [i+1 for i in range(len(tr_acc))]
loss_label = f'best epoch= {str(index_loss + 1)}'
acc_label = f'best epoch= {str(index_acc + 1)}'

Type here to search

25°C Rain

21:07
03-04-2025

Copy of Untitled0.ipynb - Colab

colab.research.google.com/drive/1aZEFnOrLbcwHqskj8ES2ot1mf_z1plxs?usp=sharing#scrollTo=RHGKtpr8Ohk0

GmailYouTubeMapsSplit() String metho...

Copy of Untitled0.ipynb

FileEditViewInsertRuntimeToolsHelp

ShareGemini

CommandsCodeText

Connect T4

Plot training history

plt.figure(figsize= (20, 8))

plt.style.use('fivethirtyeight')

plt.subplot(1, 2, 1)

plt.plot(Epochs, tr_loss, 'r', label= 'Training loss')

plt.plot(Epochs, val_loss, 'g', label= 'Validation loss')

plt.scatter(index_loss + 1, val_lowest, s= 150, c= 'blue', label= loss_label)

plt.title('Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(Epochs, tr_acc, 'r', label= 'Training Accuracy')

plt.plot(Epochs, val_acc, 'g', label= 'Validation Accuracy')

plt.scatter(index_acc + 1 , acc_highest, s= 150, c= 'blue', label= acc_label)

plt.title('Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.tight_layout

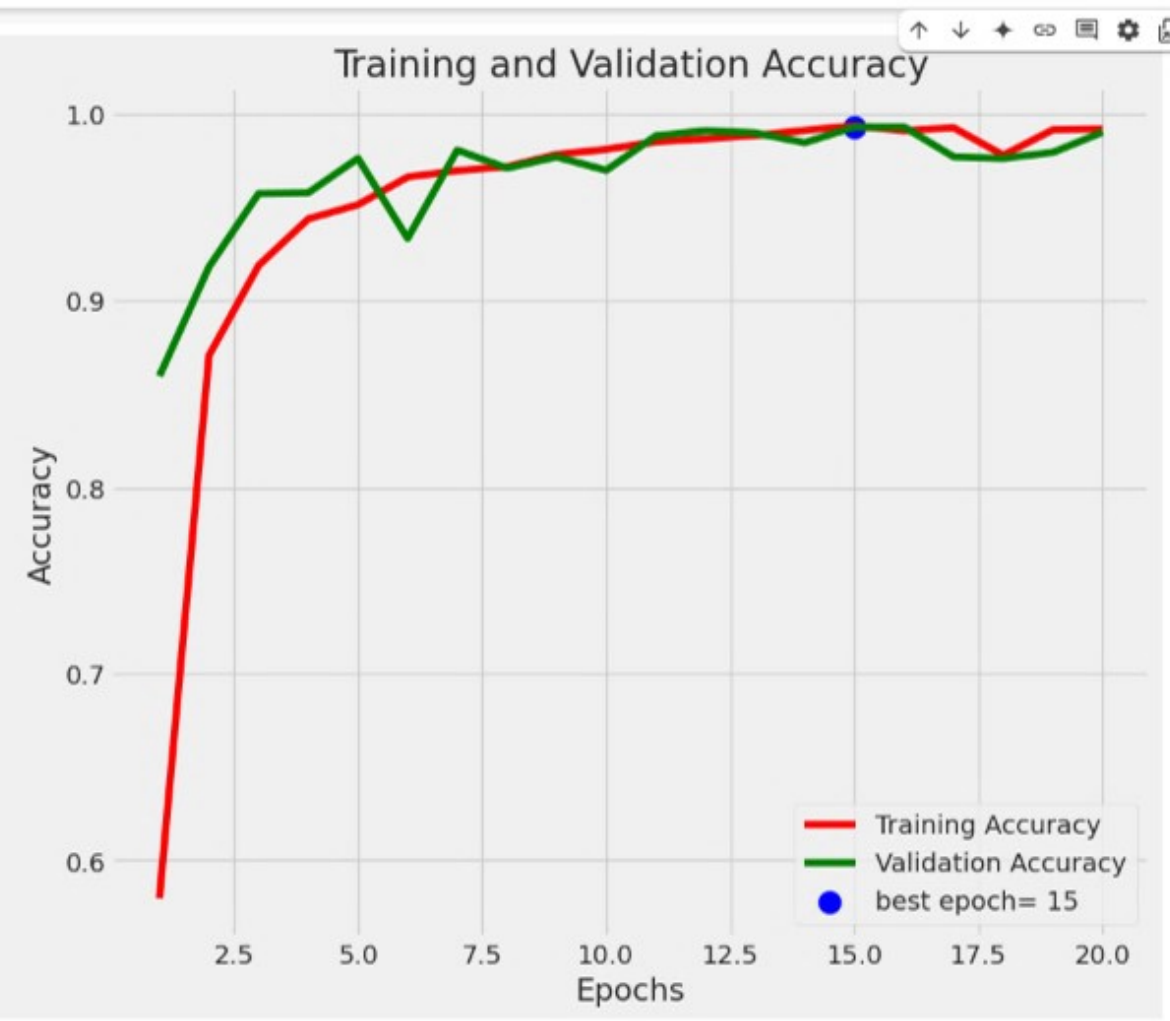
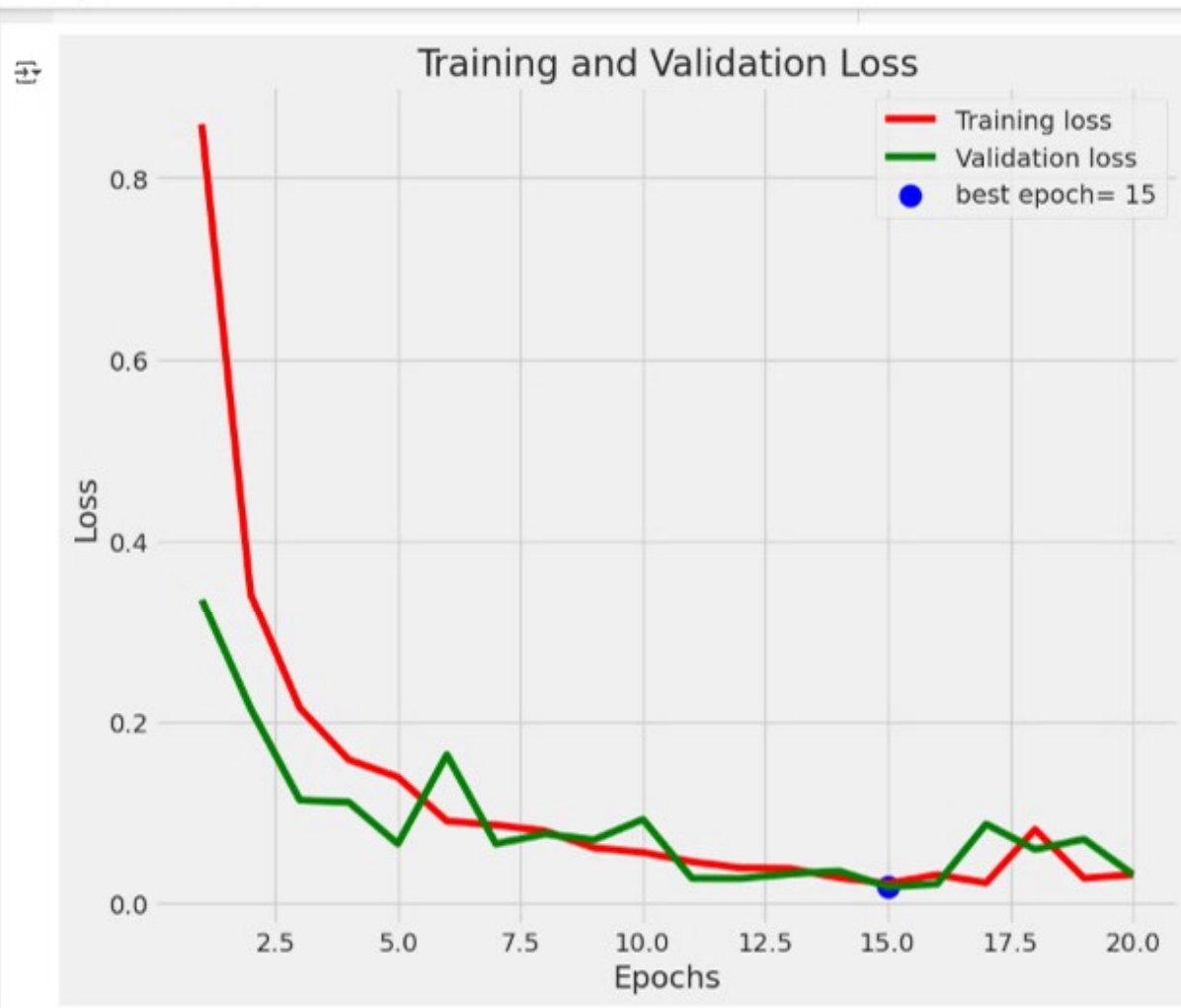
plt.show()

Type here to search

25°C Rain

21:07

03-04-2025



Copy of Untitled0.ipynb - Colab

colab.research.google.com/drive/1aZEFnOrLbcwHqskj8ES2ot1mf_z1plxs?usp=sharing#scrollTo=KETVNfE6Ordb

Gmail YouTube Maps Split() String metho...

Copy of Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

Connect T4

ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1) if ts_length%n == 0 and ts_length/n <= 80]))
test_steps = ts_length // test_batch_size

train_score = model.evaluate(train_gen, steps= test_steps, verbose= 1)
valid_score = model.evaluate(valid_gen, steps= test_steps, verbose= 1)
test_score = model.evaluate(test_gen, steps= test_steps, verbose= 1)

print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('-' * 20)
print("Valid Loss: ", valid_score[0])
print("Valid Accuracy: ", valid_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])

50/50 18s 365ms/step - accuracy: 0.9958 - loss: 0.0149
50/50 14s 279ms/step - accuracy: 0.9870 - loss: 0.0580
50/50 13s 265ms/step - accuracy: 0.9876 - loss: 0.0371
Train Loss: 0.010479768738150597
Train Accuracy: 0.9965624809265137

Valid Loss: 0.043072182685136795
Valid Accuracy: 0.9900000095367432

Test Loss: 0.031637050211429596
Test Accuracy: 0.9900000095367432

Type here to search

25°C Rain 21:09 03-04-2025

```
[ ] g_dict = test_gen.class_indices
    classes = list(g_dict.keys())

# Confusion matrix
cm = confusion_matrix(test_gen.classes, y_pred)

plt.figure(figsize= (10, 10))
plt.imshow(cm, interpolation= 'nearest', cmap= plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()

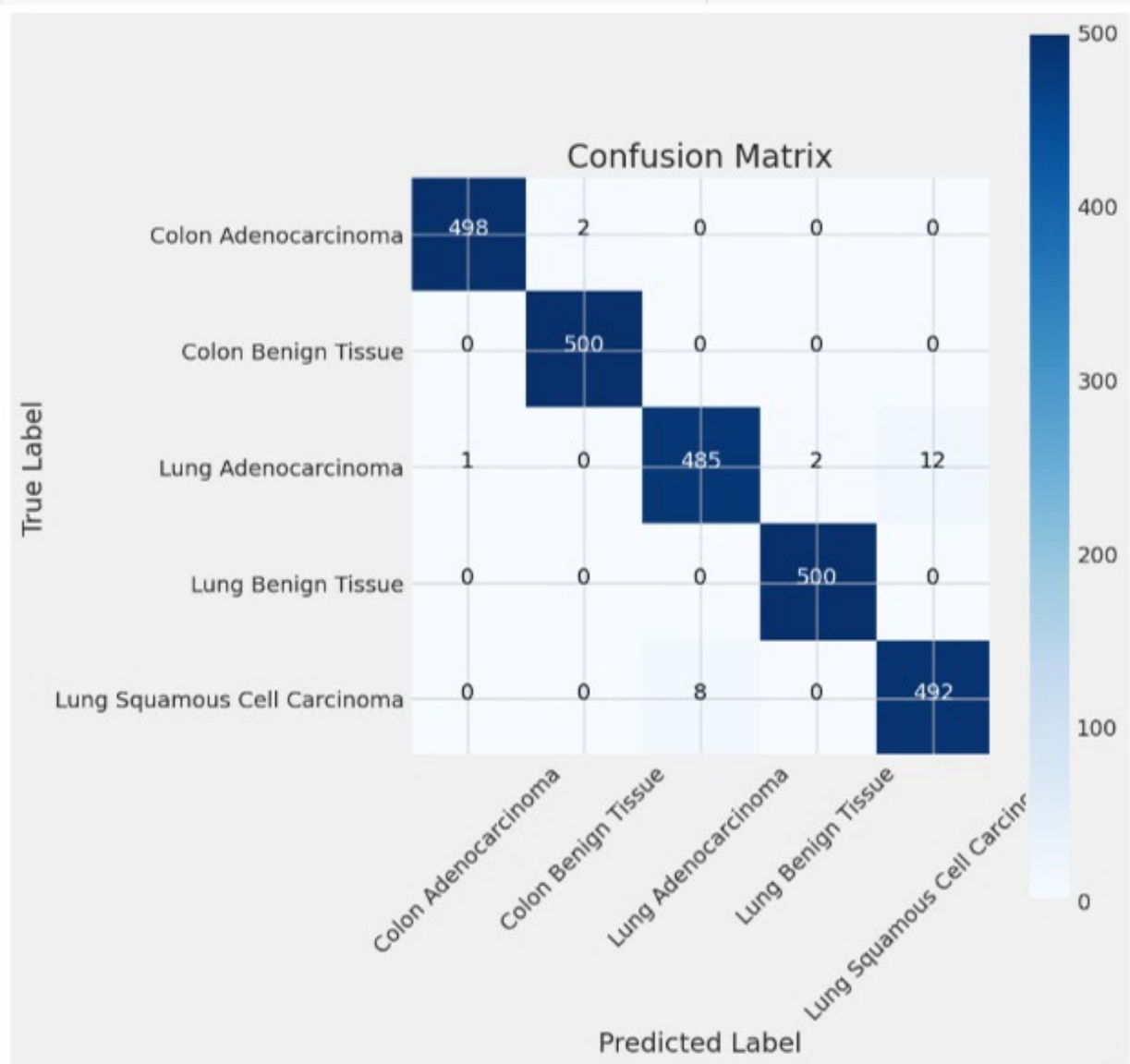
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation= 45)
plt.yticks(tick_marks, classes)

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')

plt.show()
```

500



```
# Classification report
print(classification_report(test_gen.classes, y_pred, target_names= classes))
```

	precision	recall	f1-score	support
Colon Adenocarcinoma	1.00	1.00	1.00	500
Colon Benign Tissue	1.00	1.00	1.00	500
Lung Adenocarcinoma	0.98	0.97	0.98	500
Lung Benign Tissue	1.00	1.00	1.00	500
Lung Squamous Cell Carcinoma	0.98	0.98	0.98	500
accuracy			0.99	2500
macro avg	0.99	0.99	0.99	2500
weighted avg	0.99	0.99	0.99	2500

```
[ ] model.save('Model.h5')
print("Model saved successfully.")
```

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using the SavedModel format instead.

Model saved successfully.

```
[ ] try:
    loaded_model = tf.keras.models.load_model(
        'Model.h5',
        custom_objects={'GlobalContextAttention': GlobalContextAttention}
    )
    print("Model loaded successfully.")
except Exception as e:
```