

Importing libraries

In [1]:

```
import os                                     # for working with files
import numpy as np                            # for numerical computations
import pandas as pd                           # for working with dataframes
import seaborn as sns                         # Pytorch module
import matplotlib.pyplot as plt # for plotting informations on graph and images using t
import torch.nn as nn                          # for creating neural networks
from torch.utils.data import DataLoader # for dataloaders
from PIL import Image                         # for checking images
import torch.nn.functional as F # for functions for calculating loss
import torchvision.transforms as transforms    # for transforming images into tensors
from torchvision.utils import make_grid        # for data checking
from torchvision.datasets import ImageFolder   # for working with classes and images
# from torchsummary import summary             # for getting the summary of our model
import tensorflow as tf
from tensorflow import keras
import itertools
from sklearn.metrics import precision_score, accuracy_score, recall_score, confusion_ma

%matplotlib inline

import cv2
from tensorflow.keras.preprocessing import image

from glob import glob
import shutil

from mlxtend.plotting import plot_confusion_matrix
```

Data loading and exploring

In [2]:

```
# shutil.rmtree('/kaggle/working/Lung_aca')
# shutil.rmtree('/kaggle/working/Lung_scc')
# shutil.rmtree('/kaggle/working/Lung_n')
```

In [3]:

```
os.makedirs('/kaggle/working/lung_aca')
os.makedirs('/kaggle/working/lung_scc')
os.makedirs('/kaggle/working/lung_n')
```

In [4]:

```
lung_dir1 = "/kaggle/working/"
lungss1 = os.listdir(lung_dir1)
lungss1
```

Out[4]:

```
['lung_aca',
 '.virtual_documents',
 'lung_scc',
 '__notebook_source__.ipynb',
 'lung_n']
```

```
In [5]: if '__notebook_source__.ipynb' in lungs1:  
    os.remove('/kaggle/working/__notebook_source__.ipynb')  
  
if '.virtual_documents' in lungs1:  
    shutil.rmtree('/kaggle/working/.virtual_documents')
```

```
In [6]: lung_dir2 = "/kaggle/working/"  
lung_dir2  
lung_dir2
```

```
Out[6]: ['lung_aca', 'lung_scc', 'lung_n']
```

```
In [7]: folders = glob('/kaggle/input/lung-and-colon-cancer-histopathological-images/lung_colon_im  
print('New Paths: ', folders)
```

```
New Paths: ['/kaggle/input/lung-and-colon-cancer-histopathological-images/lung_colon_im  
age_set/lung_image_sets/lung_aca', '/kaggle/input/lung-and-colon-cancer-histopathologica  
l-images/lung_colon_image_set/lung_image_sets/lung_scc', '/kaggle/input/lung-and-colon-c  
ancer-histopathological-images/lung_colon_image_set/lung_image_sets/lung_n']
```

```
In [8]: for i in folders:  
    IMAGE_FILES = glob(i + '/*.jpeg')  
    train_imgs = IMAGE_FILES  
    num_show = len(IMAGE_FILES)  
    columns = 5  
  
    plt.figure(figsize=(25,12))  
    for idx, train_img in enumerate(train_imgs):  
        if idx >= num_show:  
            break  
  
        t = train_img.split('/')  
        # Original:  
        bgrimg = cv2.imread(train_img)  
        # Denoising:  
        dst_img = cv2.fastNlMeansDenoisingColored(bgrimg, None, 10, 10, 7, 15)  
  
        cv2.imwrite('/kaggle/working/' + t[6] + '/' + t[7], dst_img)
```

```
<Figure size 1800x864 with 0 Axes>  
<Figure size 1800x864 with 0 Axes>  
<Figure size 1800x864 with 0 Axes>
```

Show some example for lung cancer

```
In [9]: lung_dir = "/kaggle/working/"  
lung_dir  
lung_dir
```

```
Out[9]: ['lung_aca', 'lung_scc', 'lung_n']
```

```
In [10]: # Number of images for each disease  
nums_train = {}
```

```

nums_val = {}
for lung in lungs:
    nums_train[lung] = len(os.listdir(lung_dir + '/' + lung))
img_per_class_train = pd.DataFrame(nums_train.values(), index=nums_train.keys(), columns=['Train data distribution :'])
img_per_class_train

```

Train data distribution :

Out[10]:

	no. of images
lung_aca	5000
lung_scc	5000
lung_n	5000

In [11]:

```

# Function to show image
train = ImageFolder(lung_dir, transform=transforms.ToTensor())
def show_image(image, label):
    print("Label :" + train.classes[label] + "(" + str(label) + ")")
    return image.permute(1, 2, 0)

```

Lung_aca

In [12]:

```

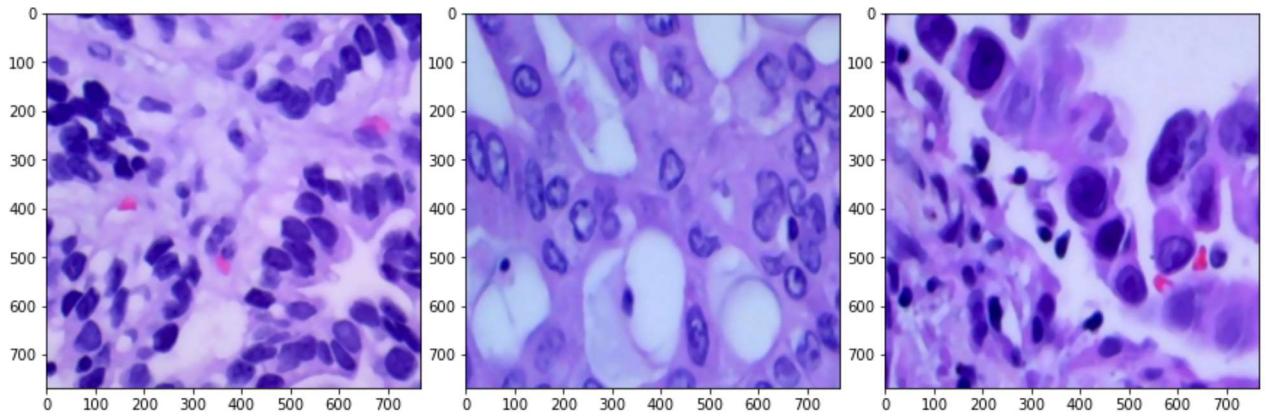
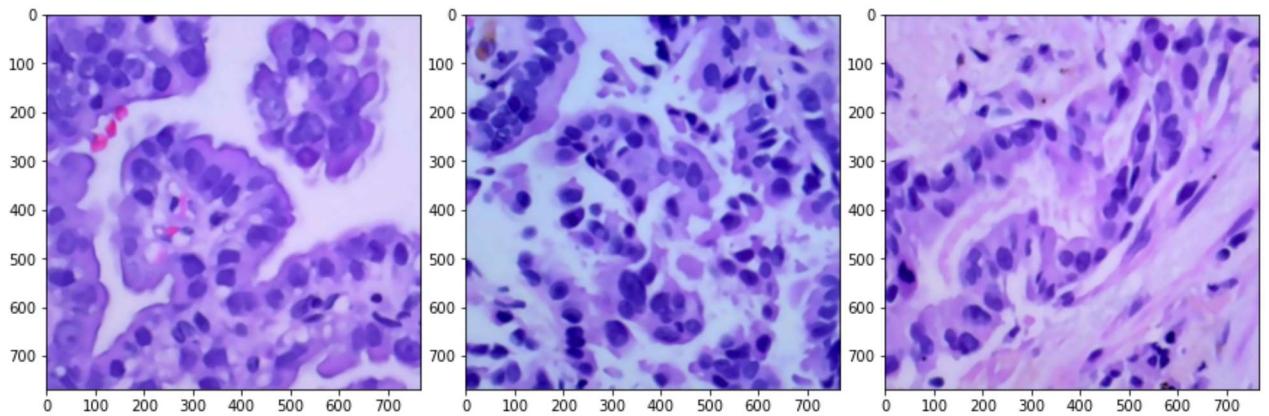
fig, axs = plt.subplots(2, 3, figsize=(12,10))
fig.tight_layout(pad=0)
axs[0,0].imshow(show_image(*train[1]))
axs[0,1].imshow(show_image(*train[1100]))
axs[1, 0].imshow(show_image(*train[2010]))
axs[1,1].imshow(show_image(*train[3500]))
axs[0,2].imshow(show_image(*train[4120]))
axs[1,2].imshow(show_image(*train[4860]))

```

Label :lung_aca(0)
Label :lung_aca(0)
Label :lung_aca(0)
Label :lung_aca(0)
Label :lung_aca(0)
Label :lung_aca(0)

Out[12]:

<matplotlib.image.AxesImage at 0x7f473d5cf2d0>

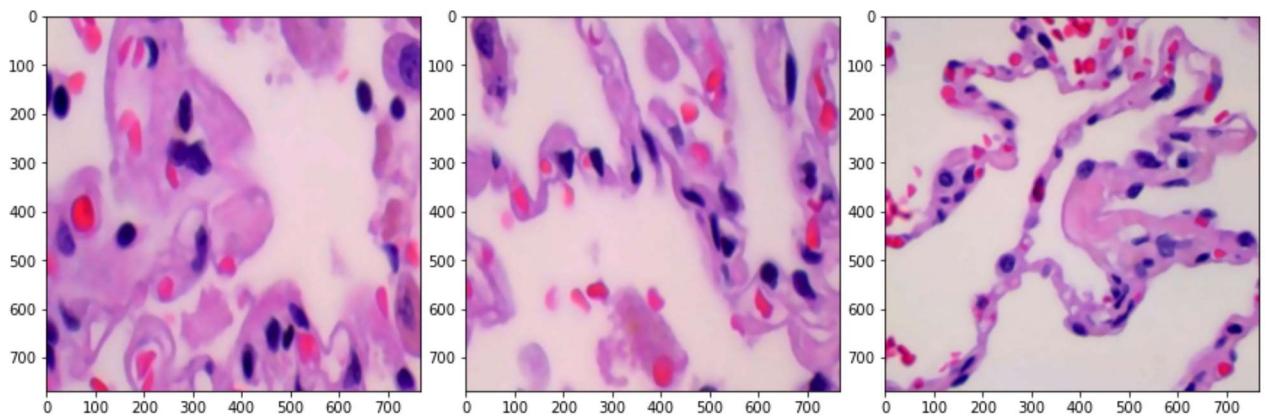
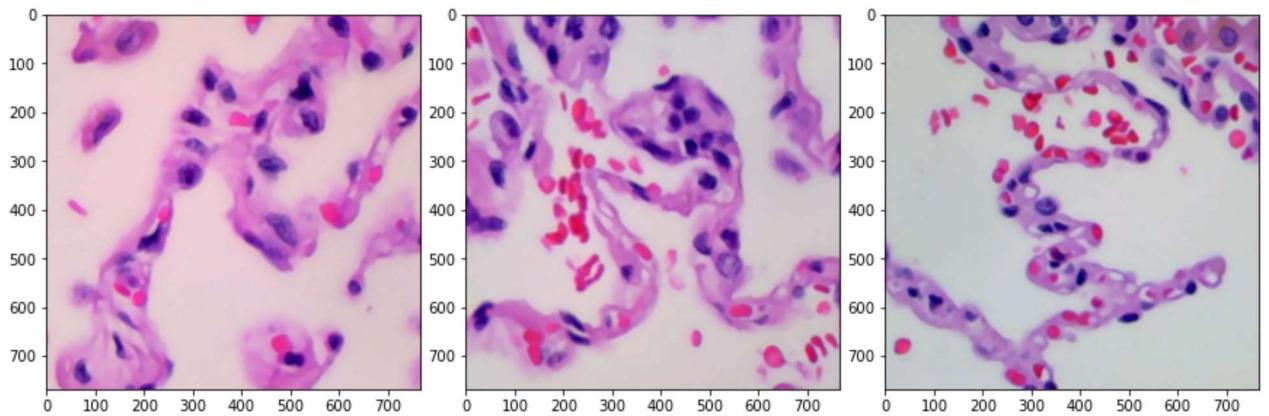


Lung_n

```
In [13]: fig, axs = plt.subplots(2, 3, figsize=(12,10))
fig.tight_layout(pad=0)
axs[0,0].imshow(show_image(*train[5010]))
axs[0,1].imshow(show_image(*train[6050]))
axs[1, 0].imshow(show_image(*train[7000]))
axs[1,1].imshow(show_image(*train[7500]))
axs[0,2].imshow(show_image(*train[8000]))
axs[1,2].imshow(show_image(*train[8620]))
```

```
Label :lung_n(1)
<matplotlib.image.AxesImage at 0x7f473c2252d0>
```

```
Out[13]:
```

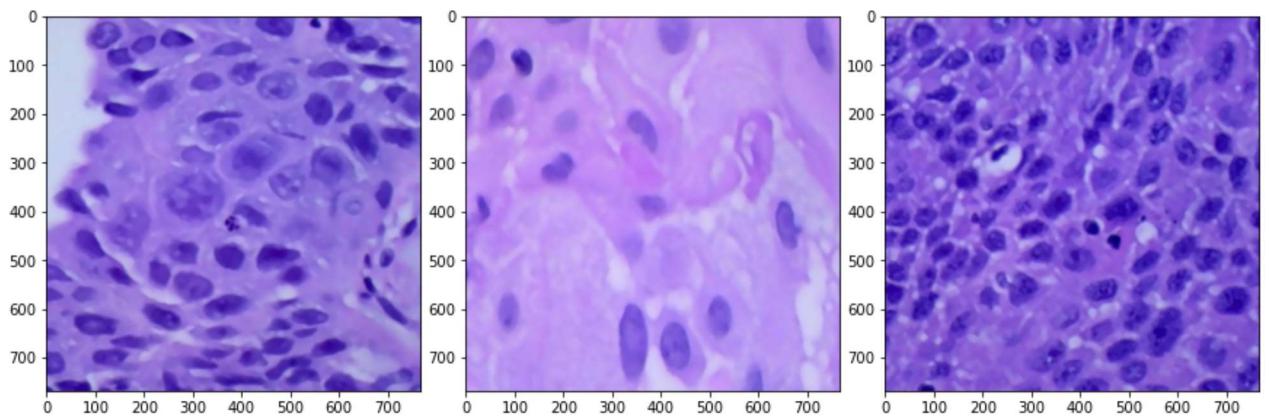
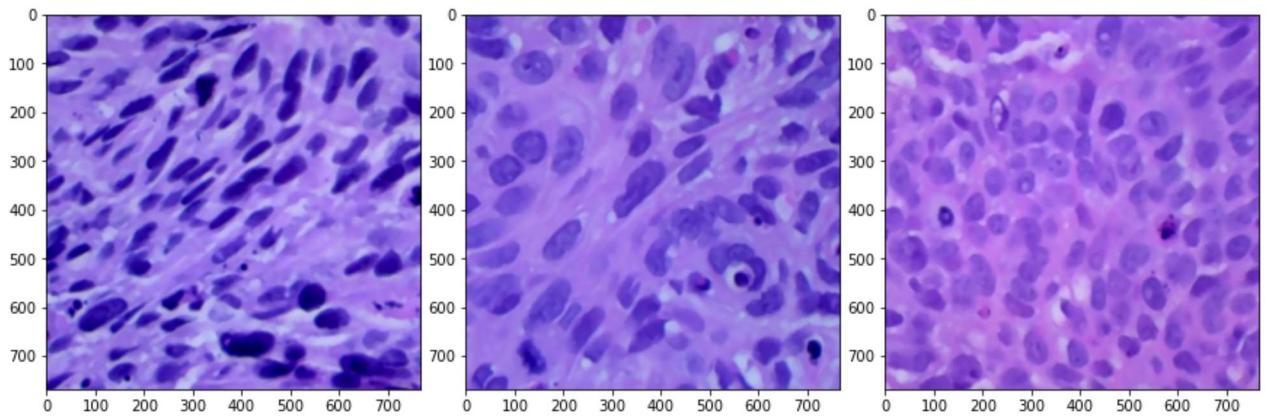


Lung_scc

```
In [14]: fig, axs = plt.subplots(2, 3, figsize=(12,10))
fig.tight_layout(pad=0)
axs[0,0].imshow(show_image(*train[11001]))
axs[0,1].imshow(show_image(*train[12000]))
axs[1, 0].imshow(show_image(*train[13050]))
axs[1,1].imshow(show_image(*train[14000]))
axs[0,2].imshow(show_image(*train[14200]))
axs[1,2].imshow(show_image(*train[14800]))
```

```
Label :lung_scc(2)
Label :lung_scc(2)
Label :lung_scc(2)
Label :lung_scc(2)
Label :lung_scc(2)
Label :lung_scc(2)
```

```
Out[14]: <matplotlib.image.AxesImage at 0x7f4737733290>
```



Modeling

In [15]:

```
train_gen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                                       rotation_range = 20 ,
                                                       horizontal_flip = True ,
                                                       validation_split = 0.2
)
valid_gen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255,validation_spl
train_data = train_gen.flow_from_directory(lung_dir, subset='training', target_size=(22
                                               class_mode='categorical', shuffle=True)

val_data = valid_gen.flow_from_directory(lung_dir, subset='validation', target_size=(22
                                               class_mode='categorical', shuffle=False)
```

Found 12000 images belonging to 3 classes.

Found 3000 images belonging to 3 classes.

In [16]:

```
unique, counts = np.unique(val_data.classes, return_counts=True)
dict(zip(unique, counts))
```

Out[16]: {0: 1000, 1: 1000, 2: 1000}

model_1

```
In [17]: model_1 = keras.models.Sequential()
model_1.add(keras.layers.Conv2D(8, 3, activation='relu', input_shape=(224, 224, 3)))

model_1.add(keras.layers.Dropout(0.05))
model_1.add(keras.layers.MaxPooling2D())

model_1.add(keras.layers.Conv2D(16, 3, activation='relu', input_shape=(224, 224, 3)))

model_1.add(keras.layers.Dropout(0.1))
model_1.add(keras.layers.MaxPooling2D())

model_1.add(keras.layers.Conv2D(32, 3, activation='relu', input_shape=(224, 224, 3)))

model_1.add(keras.layers.Dropout(0.1))
model_1.add(keras.layers.MaxPooling2D())

model_1.add(keras.layers.Conv2D(64, 3, activation='relu'))
model_1.add(keras.layers.Dropout(0.15))
model_1.add(keras.layers.MaxPooling2D())

model_1.add(keras.layers.Conv2D(128, 3, activation='relu'))
model_1.add(keras.layers.Dropout(0.2))
model_1.add(keras.layers.MaxPooling2D())

model_1.add(keras.layers.Flatten())
model_1.add(keras.layers.Dense(256, activation='relu'))
model_1.add(keras.layers.Dense(3, activation='softmax'))

model_1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model_1.summary()
```

2023-02-04 01:02:34.827754: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:34.829045: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:34.829793: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:34.830697: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 AVX512F FMA To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-02-04 01:02:34.830951: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:34.831655: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:34.832287: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:39.770211: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2023-02-04 01:02:39.771054: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node read from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero

```

one NUMA node, so returning NUMA node zero
2023-02-04 01:02:39.771762: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2023-02-04 01:02:39.772444: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Cre
ated device /job:localhost/replica:0/task:0/device:GPU:0 with 15043 MB memory: -> devic
e: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
Model: "sequential"

```

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 222, 222, 8)	224
dropout (Dropout)	(None, 222, 222, 8)	0
max_pooling2d (MaxPooling2D)	(None, 111, 111, 8)	0
conv2d_1 (Conv2D)	(None, 109, 109, 16)	1168
dropout_1 (Dropout)	(None, 109, 109, 16)	0
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 16)	0
conv2d_2 (Conv2D)	(None, 52, 52, 32)	4640
dropout_2 (Dropout)	(None, 52, 52, 32)	0
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 32)	0
conv2d_3 (Conv2D)	(None, 24, 24, 64)	18496
dropout_3 (Dropout)	(None, 24, 24, 64)	0
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 64)	0
conv2d_4 (Conv2D)	(None, 10, 10, 128)	73856
dropout_4 (Dropout)	(None, 10, 10, 128)	0
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 128)	0
flatten (Flatten)	(None, 3200)	0
dense (Dense)	(None, 256)	819456
dense_1 (Dense)	(None, 3)	771
<hr/>		
Total params:	918,611	
Trainable params:	918,611	
Non-trainable params:	0	

In [18]:

```

history = model_1.fit(train_data,
                      validation_data=val_data,
                      epochs = 5)

```

```

2023-02-04 01:02:41.742682: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:1
85] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/5

```

```

2023-02-04 01:02:44.669520: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cu
DNN version 8005
188/188 [=====] - 262s 1s/step - loss: 0.3862 - accuracy: 0.828
6 - val_loss: 0.3469 - val_accuracy: 0.8620
Epoch 2/5
188/188 [=====] - 254s 1s/step - loss: 0.2327 - accuracy: 0.904
6 - val_loss: 0.2183 - val_accuracy: 0.9333
Epoch 3/5
188/188 [=====] - 255s 1s/step - loss: 0.1913 - accuracy: 0.922
7 - val_loss: 0.2036 - val_accuracy: 0.9133
Epoch 4/5
188/188 [=====] - 257s 1s/step - loss: 0.1573 - accuracy: 0.937
4 - val_loss: 0.1955 - val_accuracy: 0.9130
Epoch 5/5
188/188 [=====] - 252s 1s/step - loss: 0.1342 - accuracy: 0.948
7 - val_loss: 0.2399 - val_accuracy: 0.8917

```

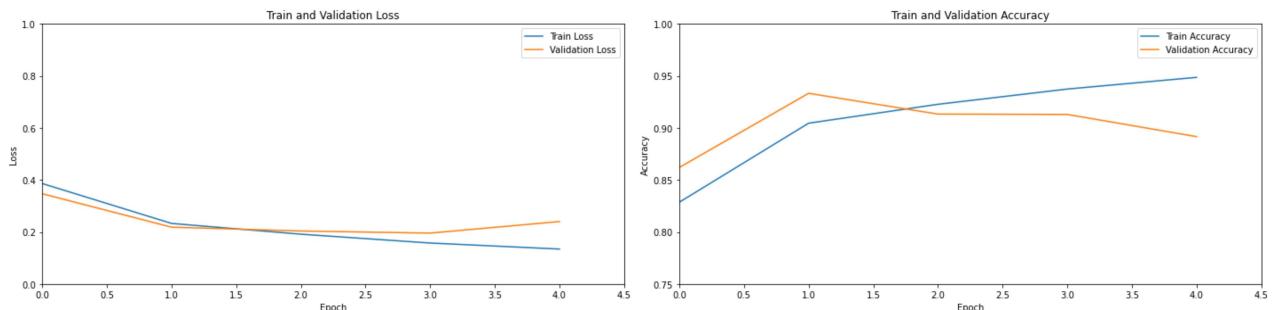
In [19]:

```

plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
plt.title("Train and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.plot(history.history['loss'], label="Train Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.xlim(0, 4.5)
plt.ylim(0.0,1.0)
plt.legend()

plt.subplot(1,2,2)
plt.title("Train and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.plot(history.history['accuracy'], label="Train Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.xlim(0, 4.5)
plt.ylim(0.75,1.0)
plt.legend()
plt.tight_layout()

```



In [20]:

```

Y_pred = model_1.predict(val_data)
y_pred = np.argmax(Y_pred, axis=1)

print(classification_report(val_data.classes, y_pred))

```

	precision	recall	f1-score	support
0	0.99	0.69	0.81	1000
1	1.00	1.00	1.00	1000

2	0.76	0.99	0.86	1000
accuracy			0.89	3000
macro avg	0.91	0.89	0.89	3000
weighted avg	0.91	0.89	0.89	3000

In [21]:

```
# calculating and plotting the confusion matrix
cm1 = confusion_matrix(val_data.classes, y_pred)
plot_confusion_matrix(conf_mat=cm1, show_absolute=True,
                      show_normed=True,
                      colorbar=True)
plt.show()
```

