About the data

folder lung_image_sets contains three secondary subfolders: lung_aca subfolder with 5000 images of lung adenocarcinomas, lung_scc subfolder with 5000 images of lung squamous cell carcinomas, and lung_n subfolder with 5000 images of benign lung tissues.

Importing libraries

In [1]:
```
import os                       # for working with files
import numpy as np              # for numerical computationss
import pandas as pd             # for working with dataframes
import seaborn as sns
import torch                    # 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 ts
from  tensorflow import keras
import itertools
from sklearn.metrics import precision_score, accuracy_score, recall_score, confusion_ma
from mlxtend.plotting import plot_confusion_matrix

%matplotlib inline
```

Data loading and exploring

In [2]:
```
lung_dir = "../input/lung-and-colon-cancer-histopathological-images/lung_colon_image_se
lungs = os.listdir(lung_dir)
```

In [3]:
```
lungs
```

Out[3]:
```
['lung_aca', 'lung_scc', 'lung_n']
```

In [4]:
```
# 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(), column
print('Train data distribution :')
img_per_class_train
```

```
Train data distribution :
```

Out[4]:

	no. of images
lung_aca	5000
lung_scc	5000
lung_n	5000

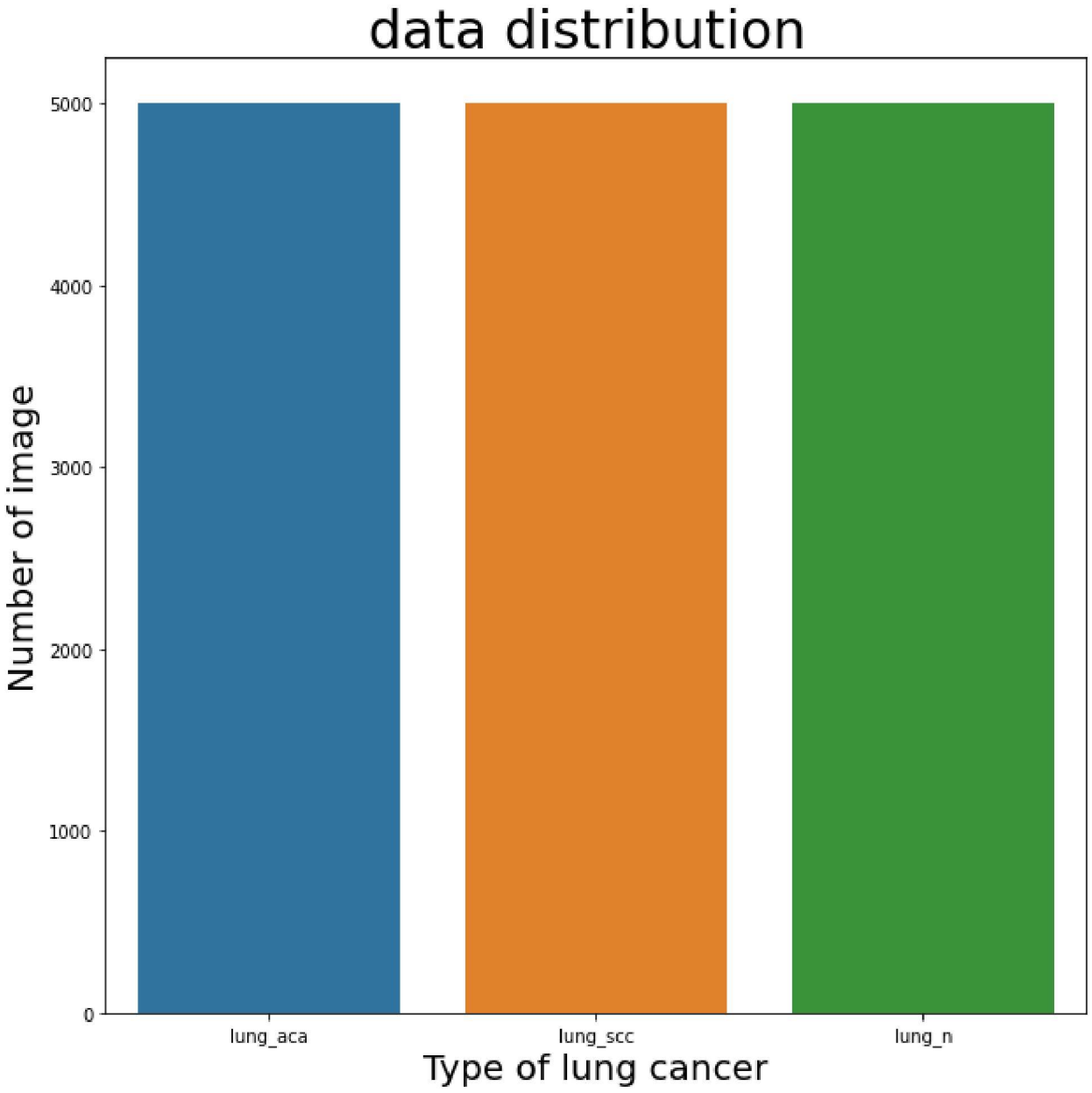
In [5]:

```
plt.figure(figsize=(10,10))
plt.title('data distribution ',fontsize=30)
plt.ylabel('Number of image',fontsize=20)
plt.xlabel('Type of lung cancer',fontsize=20)

keys = list(nums_train.keys())
vals = list(nums_train.values())
sns.barplot(x=keys, y=vals)
```

Out[5]:

```
<AxesSubplot:title={'center':'data distribution '}, xlabel='Type of lung cancer', ylabel
='Number of image'>
```



Show some example for lung cancer

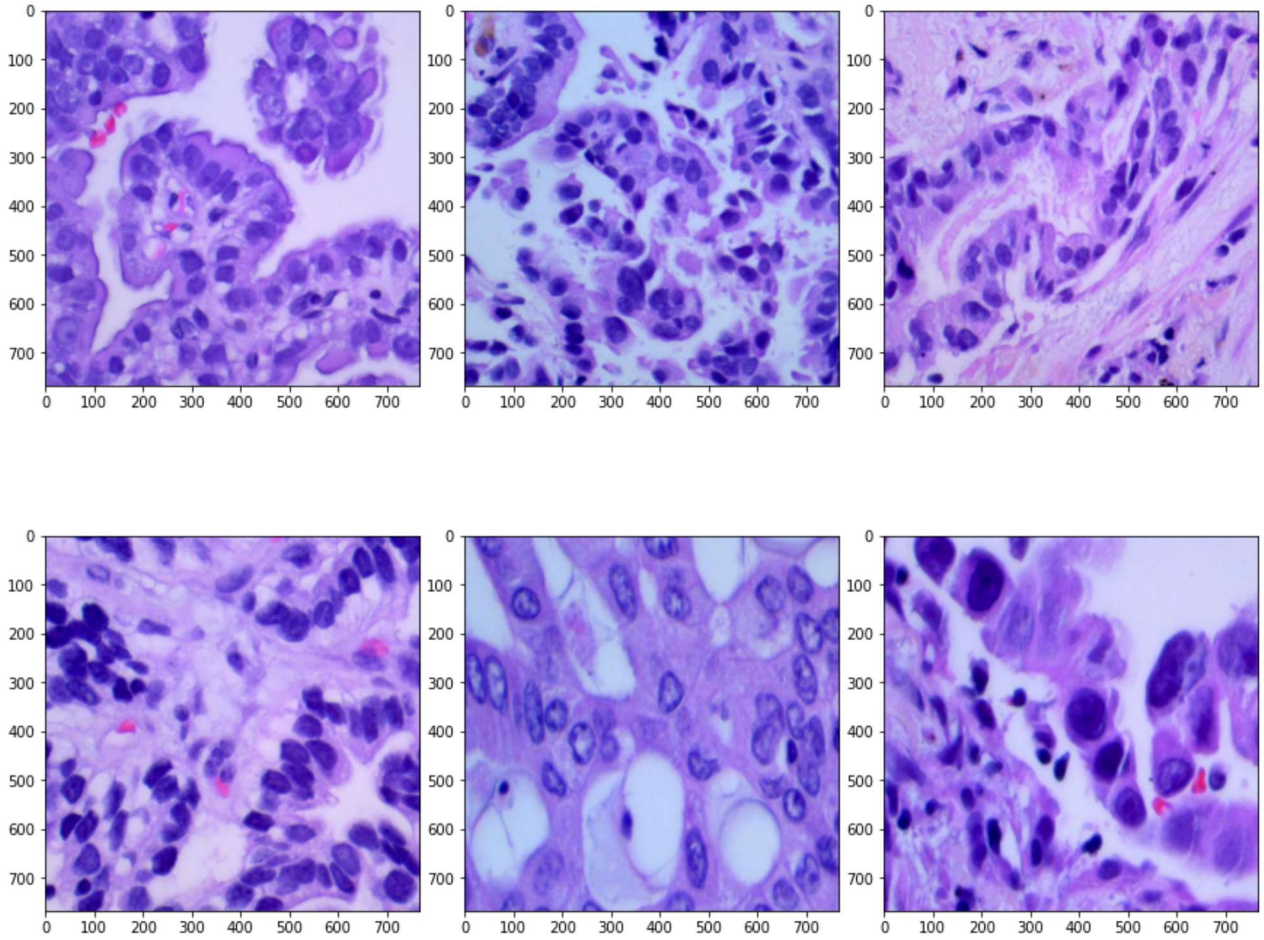
In [6]:
```
# Function to show image
train = ImageFolder(lung_dir, transform=transforms.ToTensor())
def show_image(image, label):
    print("Label :" + train.classes[label] + "(" + str(label) + ")")
    return image.permute(1, 2, 0)
```

Lung_aca

In [7]:
```
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[7]:
```
<matplotlib.image.AxesImage at 0x7f593704b190>
```

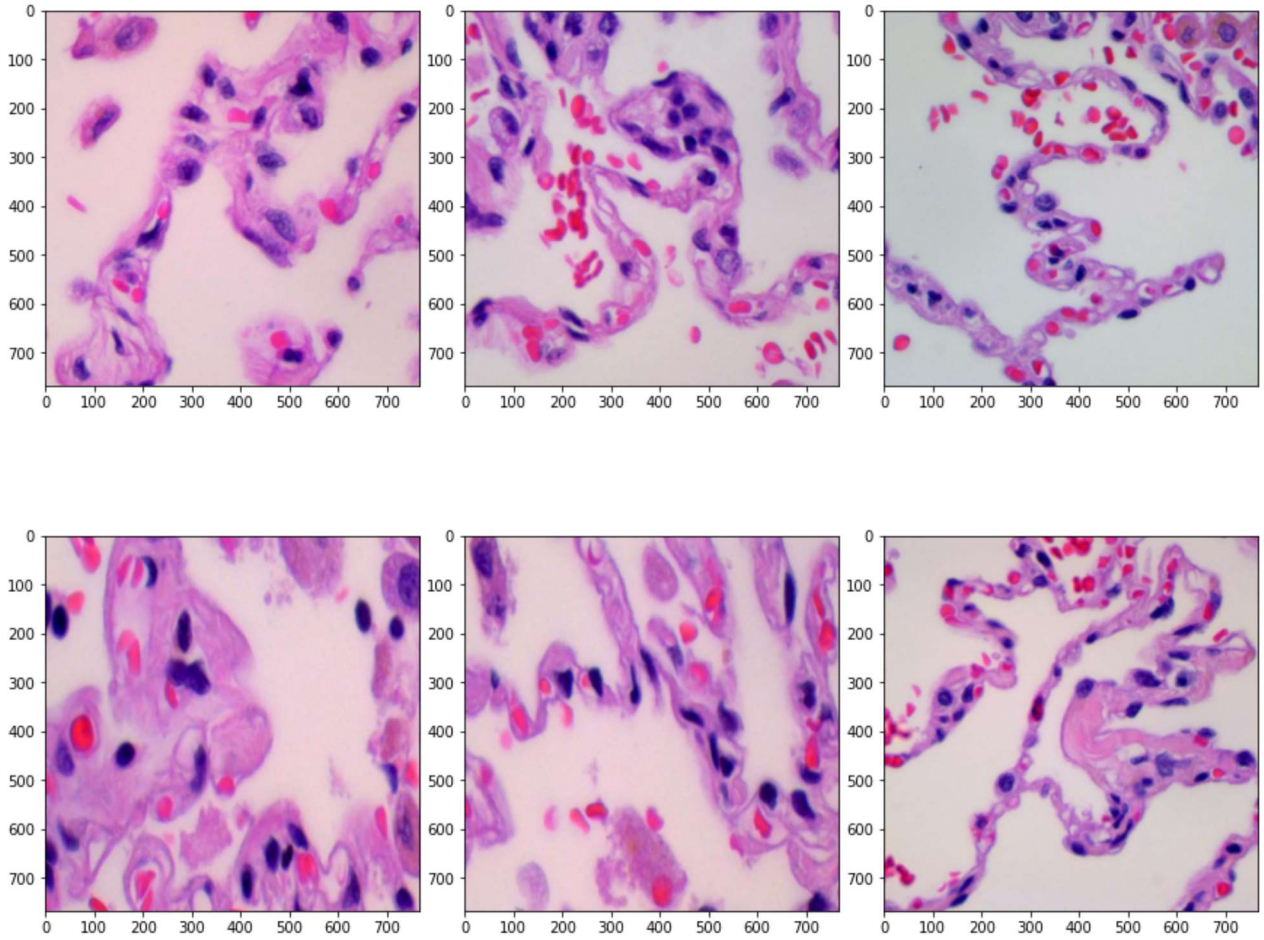


Lung_n

In [8]:
```
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)
Label :lung_n(1)
Label :lung_n(1)
Label :lung_n(1)
Label :lung_n(1)
Label :lung_n(1)
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x7f593550d190>
```

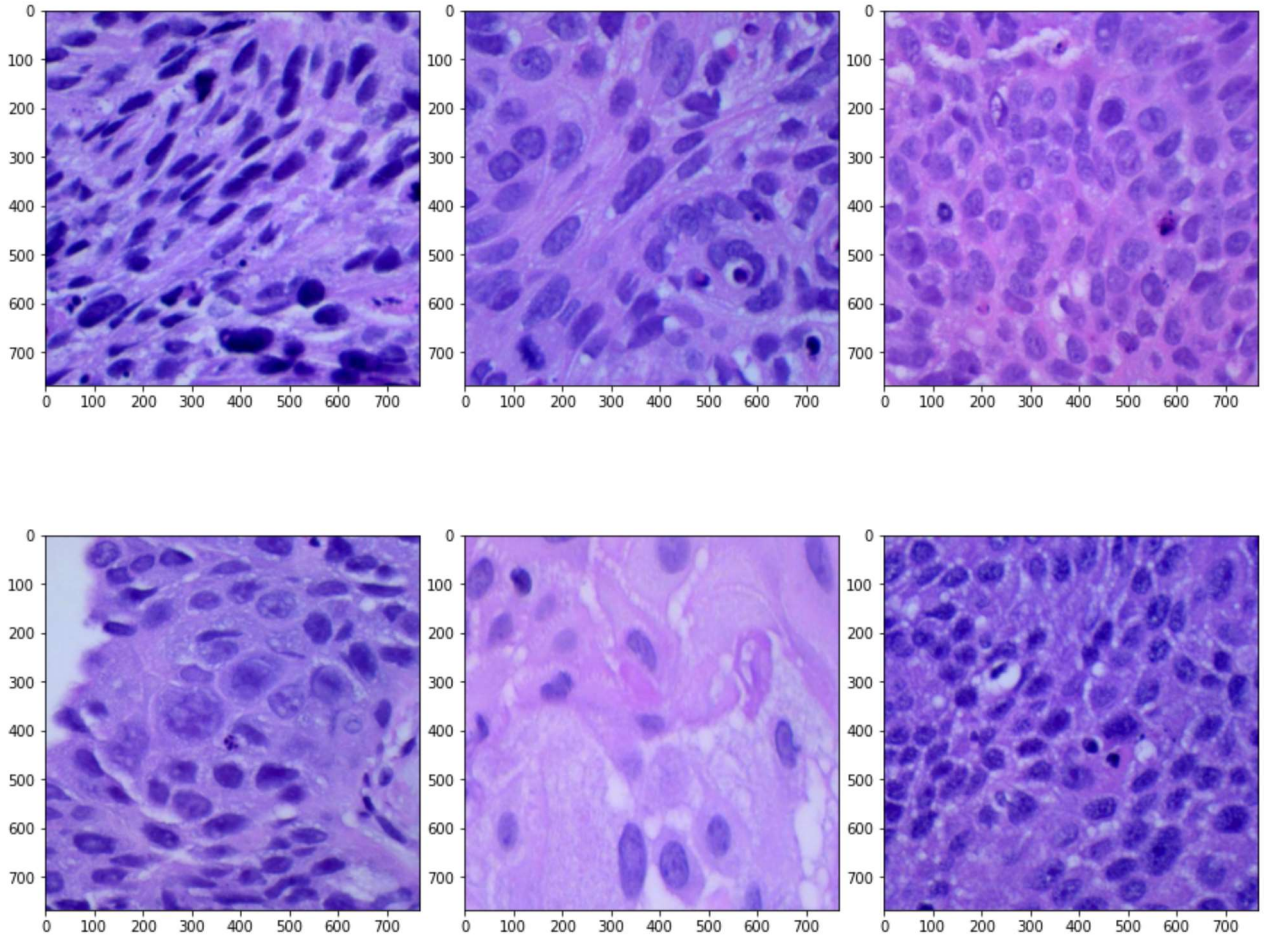


Lung_scc

In [9]:
```
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[9]:
```
<matplotlib.image.AxesImage at 0x7f59352f2050>
```



Modeling

In [10]:
```
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 [11]:
```
import numpy
unique, counts = numpy.unique(val_data.classes, return_counts=True)

dict(zip(unique, counts))
```

Out[11]:
```
{0: 1000, 1: 1000, 2: 1000}
```

model_1

In [12]:

```
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 10:46:44.960722: 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 10:46:44.962143: 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 10:46:44.963976: 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 10:46:44.965218: 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 10:46:44.966443: 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 10:46:44.967804: 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 10:46:44.970577: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Te
nsorFlow 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 fla
gs.
Model: "sequential"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
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 (MaxPooling2 (None, 54, 54, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 52, 52, 32)        4640
_________________________________________________________________
dropout_2 (Dropout)          (None, 52, 52, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
dropout_3 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 128)       73856
_________________________________________________________________
dropout_4 (Dropout)          (None, 10, 10, 128)       0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 3200)              0
_________________________________________________________________
dense (Dense)                (None, 256)               819456
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 771
=================================================================
Total params: 918,611
Trainable params: 918,611
Non-trainable params: 0
_________________________________________________________________
2023-02-04 10:46:45.216682: 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 10:46:45.217541: 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 10:46:45.218294: 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 10:46:45.219015: 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 10:46:45.219953: 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 10:46:45.220855: 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
```

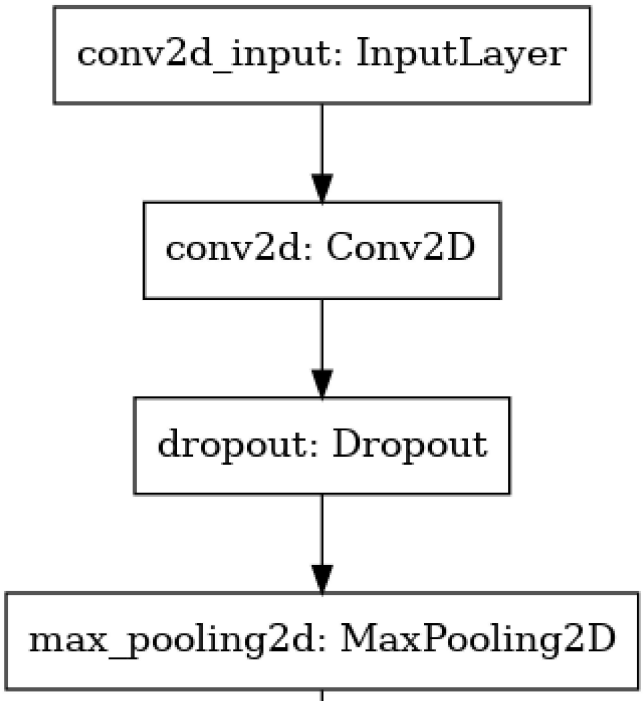
```
2023-02-04 10:46:49.461901: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
successful NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero
2023-02-04 10:46:49.463188: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937]
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 10:46:49.464269: 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 10:46:49.465270: 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 10:46:49.466263: 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 10:46:49.467226: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Cre
ated device /job:localhost/replica:0/task:0/device:GPU:0 with 13349 MB memory:  -> devic
e: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5
2023-02-04 10:46:49.467706: 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 10:46:49.468717: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Cre
ated device /job:localhost/replica:0/task:0/device:GPU:1 with 13349 MB memory:  -> devic
e: 1, name: Tesla T4, pci bus id: 0000:00:05.0, compute capability: 7.5
```

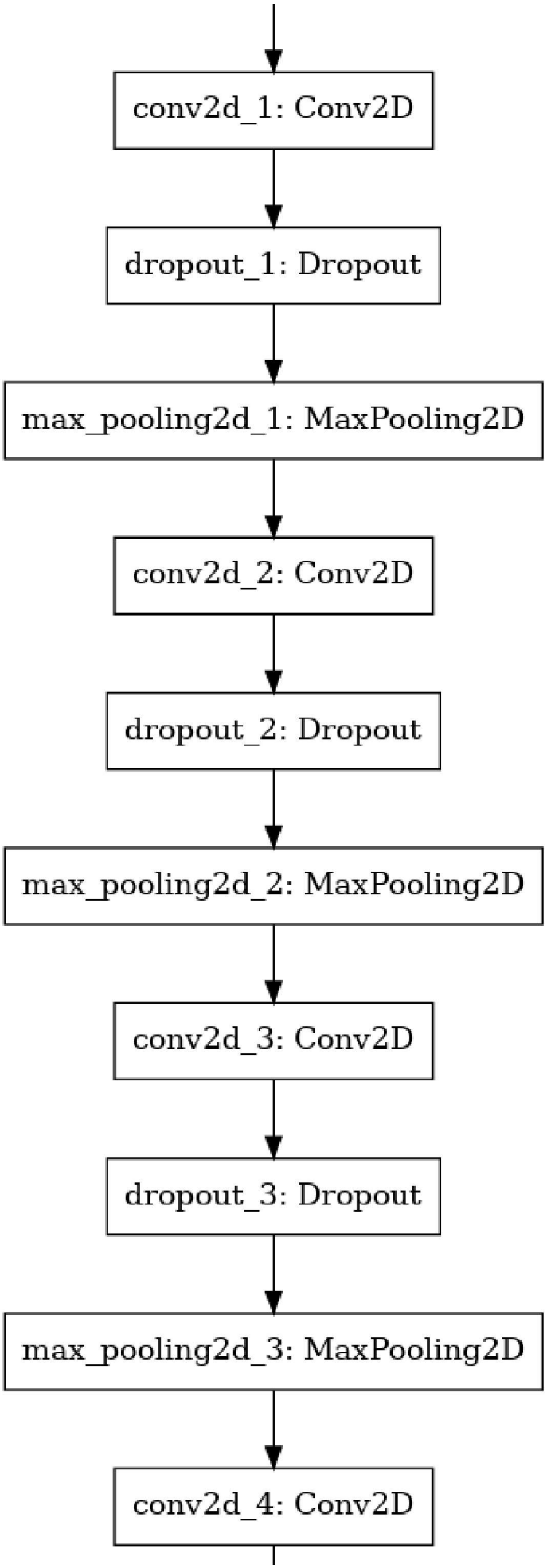
In [13]:

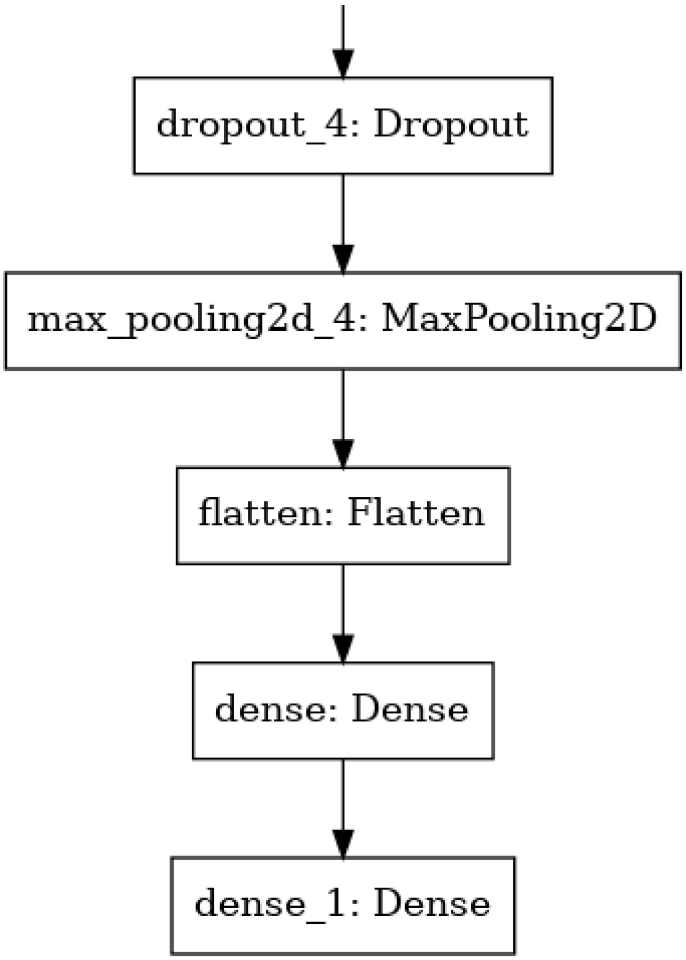
```
keras.utils.plot_model(
    model_1,
    to_file="model.png",
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir="TB",
    expand_nested=False,
    dpi=96,
    layer_range=None,

)
```

Out[13]:




conv2d_1: Conv2D
dropout_1: Dropout
max_pooling2d_1: MaxPooling2D
conv2d_2: Conv2D
dropout_2: Dropout
max_pooling2d_2: MaxPooling2D
conv2d_3: Conv2D
dropout_3: Dropout
max_pooling2d_3: MaxPooling2D
conv2d_4: Conv2D



In [14]:

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

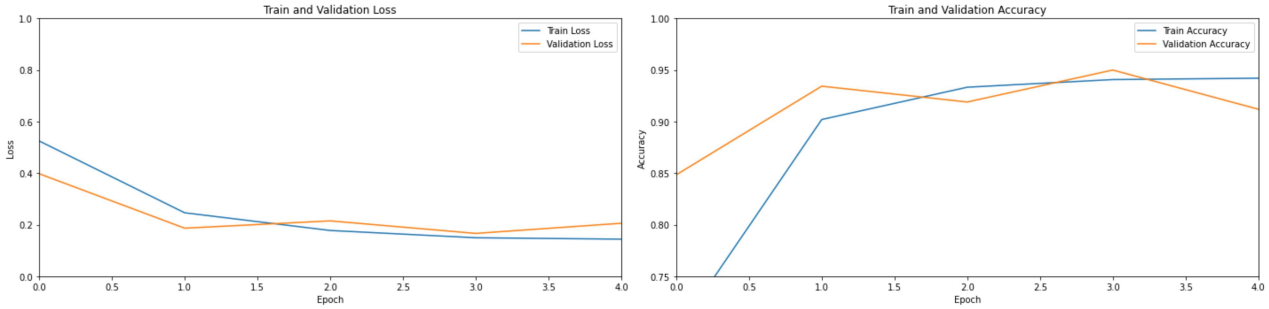
```
2023-02-04 10:46:51.525589: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:1
85] None of the MLIR Optimization Passes are enabled (registered 2)
Epoch 1/5
2023-02-04 10:46:53.614742: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded cu
DNN version 8005
188/188 [==============================] - 264s 1s/step - loss: 0.5249 - accuracy: 0.695
1 - val_loss: 0.3978 - val_accuracy: 0.8483
Epoch 2/5
188/188 [==============================] - 262s 1s/step - loss: 0.2462 - accuracy: 0.902
0 - val_loss: 0.1863 - val_accuracy: 0.9343
Epoch 3/5
188/188 [==============================] - 262s 1s/step - loss: 0.1775 - accuracy: 0.933
3 - val_loss: 0.2147 - val_accuracy: 0.9190
Epoch 4/5
188/188 [==============================] - 262s 1s/step - loss: 0.1492 - accuracy: 0.940
8 - val_loss: 0.1661 - val_accuracy: 0.9500
Epoch 5/5
188/188 [==============================] - 261s 1s/step - loss: 0.1438 - accuracy: 0.942
1 - val_loss: 0.2055 - val_accuracy: 0.9120
```

In [15]:

```
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)
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)
plt.ylim(0.75,1.0)
plt.legend()
plt.tight_layout()
```



In [16]:

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

print(classification_report(val_data.classes, y_pred))
```

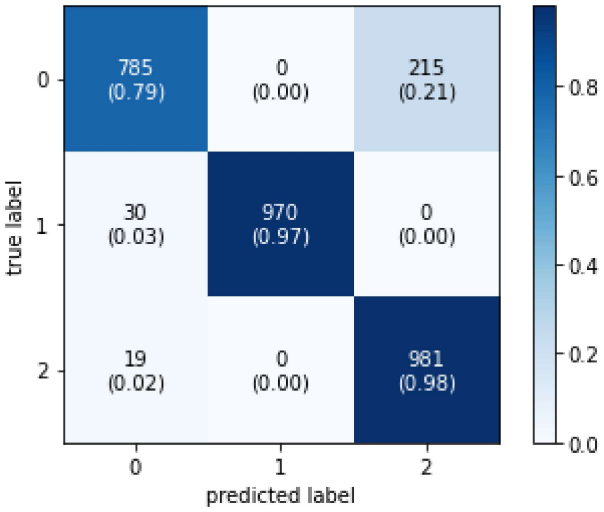
```
              precision    recall  f1-score   support

           0       0.94      0.79      0.86      1000
           1       1.00      0.97      0.98      1000
           2       0.82      0.98      0.89      1000

    accuracy                           0.91      3000
   macro avg       0.92      0.91      0.91      3000
weighted avg       0.92      0.91      0.91      3000
```

In [17]:

```
# 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()
```



Model_2

In [18]:

```
model_2 = keras.models.Sequential()

model_2.add(keras.layers.Conv2D(32, 3, activation='relu', input_shape=(224, 224, 3)))

model_2.add(keras.layers.Dropout(0.1))
model_2.add(keras.layers.MaxPooling2D())

model_2.add(keras.layers.Conv2D(64, 3, activation='relu'))
model_2.add(keras.layers.Dropout(0.2))
model_2.add(keras.layers.MaxPooling2D())

model_2.add(keras.layers.Flatten())
model_2.add(keras.layers.Dense(128, activation='relu'))
model_2.add(keras.layers.Dense(3, activation='softmax'))

model_2.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accuracy'])
model_2.summary()
```

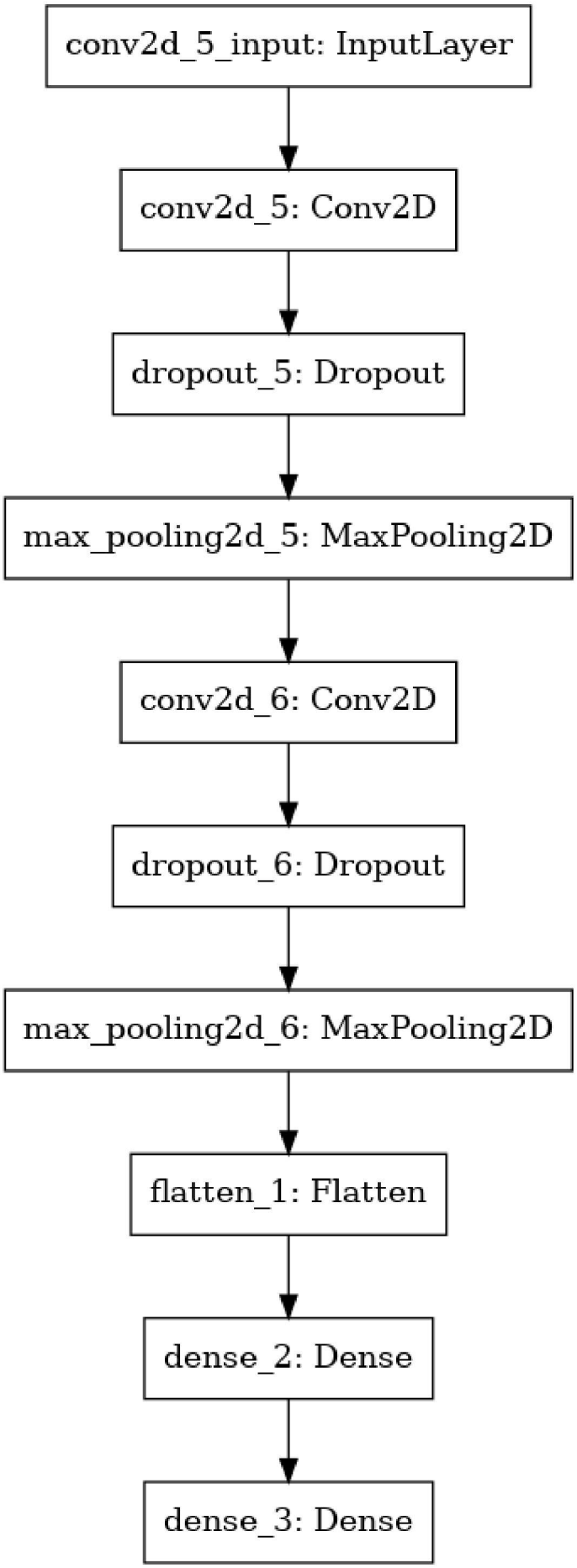
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
dropout_5 (Dropout)          (None, 222, 222, 32)      0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
dropout_6 (Dropout)          (None, 109, 109, 64)      0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 186624)            0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               23888000
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
```

```
=================================================================
Total params: 23,907,779
Trainable params: 23,907,779
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
keras.utils.plot_model(
    model_2,
    to_file="model.png",
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir="TB",
    expand_nested=False,
    dpi=96,
    layer_range=None,
    #show_layer_activations=False,
)
```

Out[19]:


conv2d_5_input: InputLayer
conv2d_5: Conv2D
dropout_5: Dropout
max_pooling2d_5: MaxPooling2D
conv2d_6: Conv2D
dropout_6: Dropout
max_pooling2d_6: MaxPooling2D
flatten_1: Flatten
dense_2: Dense
dense_3: Dense

In [20]:

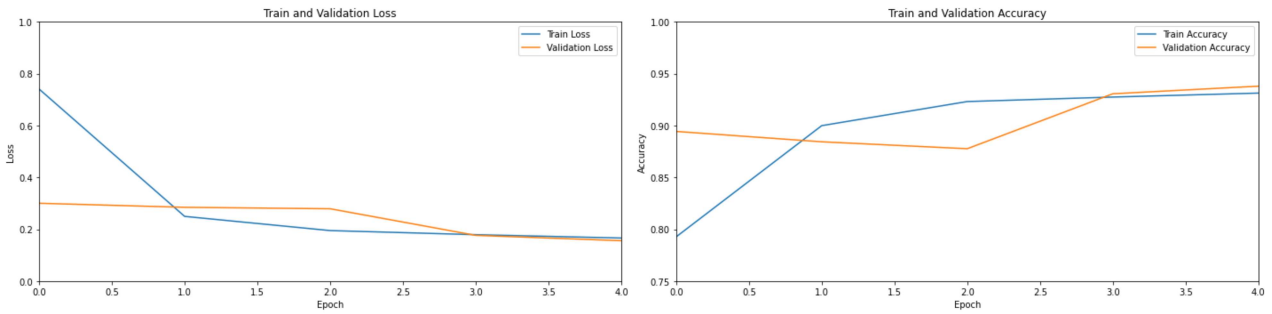
```
history = model_2.fit(train_data,
          validation_data=val_data,
          epochs = 5)
```

```
Epoch 1/5
188/188 [==============================] - 265s 1s/step - loss: 0.7409 - accuracy: 0.792
7 - val_loss: 0.3001 - val_accuracy: 0.8943
Epoch 2/5
188/188 [==============================] - 259s 1s/step - loss: 0.2495 - accuracy: 0.899
9 - val_loss: 0.2845 - val_accuracy: 0.8843
Epoch 3/5
188/188 [==============================] - 261s 1s/step - loss: 0.1947 - accuracy: 0.923
2 - val_loss: 0.2789 - val_accuracy: 0.8777
Epoch 4/5
188/188 [==============================] - 262s 1s/step - loss: 0.1789 - accuracy: 0.927
5 - val_loss: 0.1763 - val_accuracy: 0.9307
Epoch 5/5
188/188 [==============================] - 258s 1s/step - loss: 0.1660 - accuracy: 0.931
3 - val_loss: 0.1556 - val_accuracy: 0.9380
```

In [21]:

```
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)
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)
plt.ylim(0.75,1.0)
plt.legend()
plt.tight_layout()
```



In [22]:

```
Y_pred = model_2.predict(val_data)
y_pred = np.argmax(Y_pred, axis=1)

print(classification_report(val_data.classes, y_pred))
```

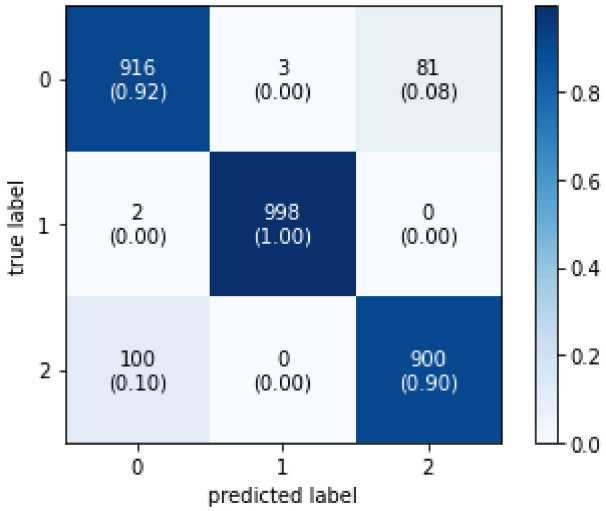
```
              precision    recall  f1-score   support
```

```
              0       0.90      0.92      0.91      1000
              1       1.00      1.00      1.00      1000
              2       0.92      0.90      0.91      1000

       accuracy                           0.94      3000
      macro avg       0.94      0.94      0.94      3000
   weighted avg       0.94      0.94      0.94      3000
```

In [23]:

```
# 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()
```



Model_3

In [24]:

```
model_3 = keras.models.Sequential()



model_3.add(keras.layers.Conv2D(64, 3, activation='relu', input_shape=(224, 224, 3)))
model_3.add(keras.layers.MaxPooling2D())

model_3.add(keras.layers.Flatten())
model_3.add(keras.layers.Dense(128, activation='relu'))
model_3.add(keras.layers.Dense(3, activation='softmax'))

model_3.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accuracy'])
model_3.summary()
```

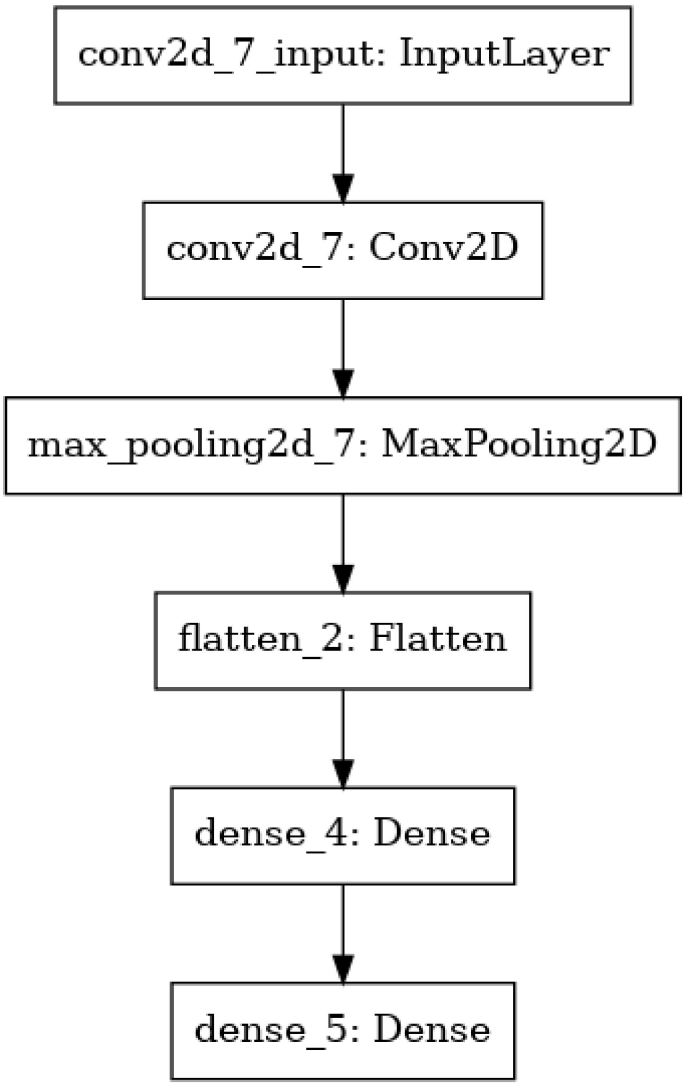
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 222, 222, 64)      1792
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 111, 111, 64)      0
```

```
flatten_2 (Flatten)          (None, 788544)            0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               100933760
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 387
=================================================================
Total params: 100,935,939
Trainable params: 100,935,939
Non-trainable params: 0
_________________________________________________________________
```

In [25]:

```
keras.utils.plot_model(
    model_3,
    to_file="model.png",
    show_shapes=False,
    show_dtype=False,
    show_layer_names=True,
    rankdir="TB",
    expand_nested=False,
    dpi=96,
    layer_range=None,
    #show_layer_activations=False,
)
```

Out[25]:

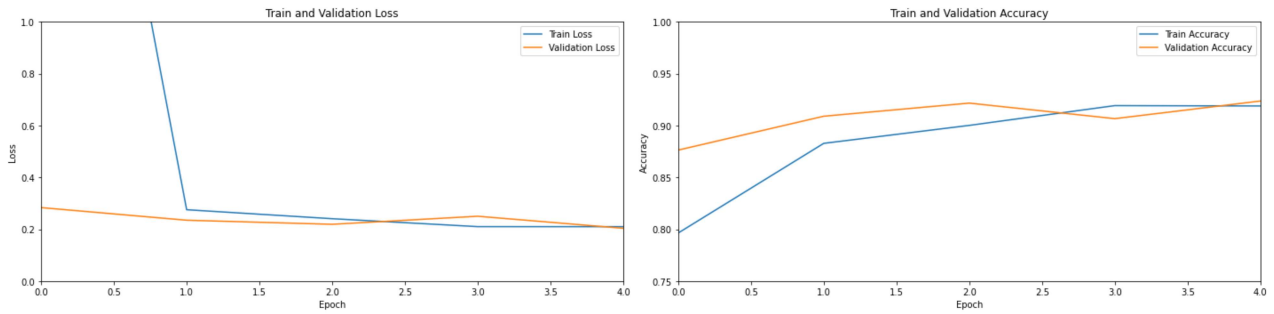


In [26]:
```
history = model_3.fit(train_data,
          validation_data=val_data,
          epochs = 5)
```

```
Epoch 1/5
188/188 [==============================] - 263s 1s/step - loss: 3.2486 - accuracy: 0.796
3 - val_loss: 0.2834 - val_accuracy: 0.8763
Epoch 2/5
188/188 [==============================] - 257s 1s/step - loss: 0.2752 - accuracy: 0.882
8 - val_loss: 0.2347 - val_accuracy: 0.9090
Epoch 3/5
188/188 [==============================] - 255s 1s/step - loss: 0.2406 - accuracy: 0.900
2 - val_loss: 0.2190 - val_accuracy: 0.9217
Epoch 4/5
188/188 [==============================] - 286s 2s/step - loss: 0.2100 - accuracy: 0.919
3 - val_loss: 0.2500 - val_accuracy: 0.9067
Epoch 5/5
188/188 [==============================] - 266s 1s/step - loss: 0.2098 - accuracy: 0.918
9 - val_loss: 0.2036 - val_accuracy: 0.9237
```

In [27]:
```
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)
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)
plt.ylim(0.75,1.0)
plt.legend()
plt.tight_layout()
```



In [28]:
```
Y_pred = model_3.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       0.86      0.91      0.89      1000
              1       1.00      0.97      0.99      1000
              2       0.91      0.88      0.90      1000

       accuracy                           0.92      3000
      macro avg       0.93      0.92      0.92      3000
   weighted avg       0.93      0.92      0.92      3000
```

In [29]:

```
# 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()
```