cnn-model

April 1, 2024

[1]:
```
import tensorflow.keras
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.models import load_model, Model
from tensorflow.keras.layers import Dense,Conv2D, Flatten, MaxPool2D, Dropout
```

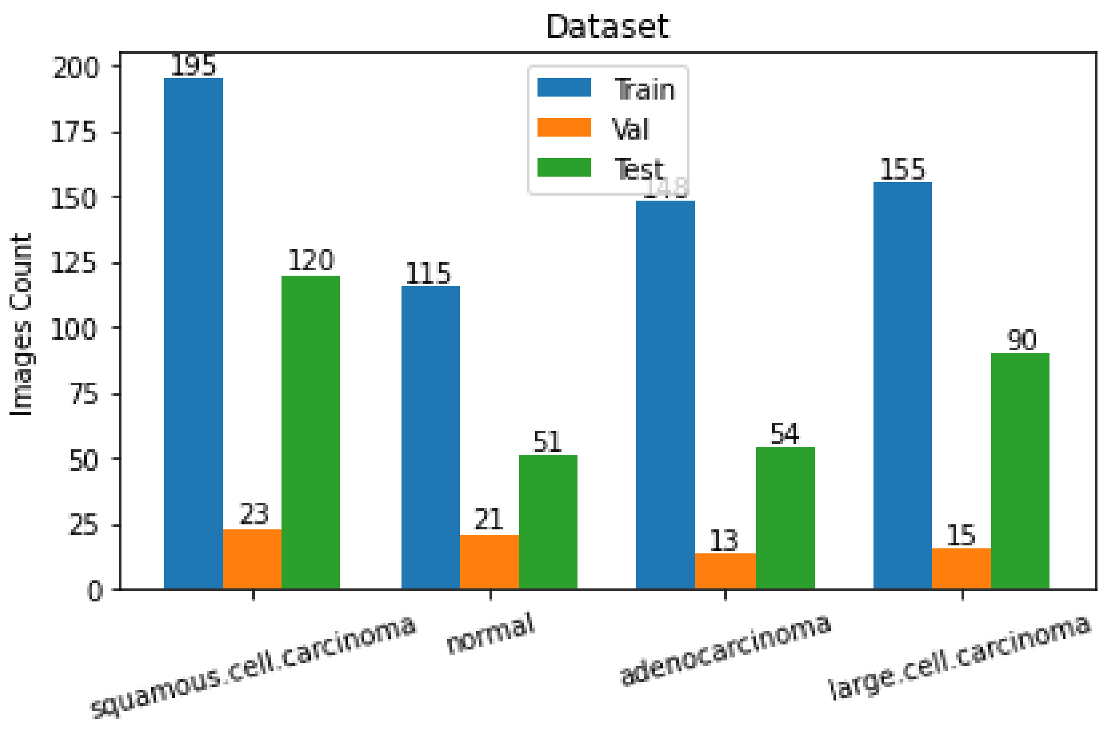
[2]:
```
import os
import cv2
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
train_path = (r"D:\M.sc BDA\Projects\Lung Cancer Detection using␣
 ↪CNN\Data\train")
val_path = (r"D:\M.sc BDA\Projects\Lung Cancer Detection using CNN\Data\valid")
test_path = (r"D:\M.sc BDA\Projects\Lung Cancer Detection using CNN\Data\test")
```

[4]:
```
def GetDatasetSize(path):
    num_of_image = {}
    for folder in os.listdir(path):
        # Counting the Number of Files in the Folder
        num_of_image[folder] = len(os.listdir(os.path.join(path, folder)));
    return num_of_image;

train_set = GetDatasetSize(train_path)
val_set = GetDatasetSize(val_path)
test_set = GetDatasetSize(test_path)
print(train_set,"\n\n",val_set,"\n\n",test_set)
```

```
{'adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib': 195,
'large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa': 115, 'normal': 148,
'squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa': 155}

 {'adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib': 23,
'large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa': 21, 'normal': 13,
'squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa': 15}

 {'adenocarcinoma': 120, 'large.cell.carcinoma': 51, 'normal': 54,
'squamous.cell.carcinoma': 90}
```

[5]:
```
labels = ['squamous.cell.carcinoma', 'normal', 'adenocarcinoma', 'large.cell.
 ↪carcinoma']
train_list = list(train_set.values())
val_list = list(val_set.values())
test_list = list(test_set.values())

x = np.arange(len(labels))  # the label locations
width = 0.25  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width, train_list, width, label='Train')
rects2 = ax.bar(x, val_list, width, label='Val')
rects3 = ax.bar(x + width, test_list, width, label='Test')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Images Count')
ax.set_title('Dataset')
ax.set_xticks(x, labels)
plt.xticks(rotation=15)
ax.legend()

ax.bar_label(rects1)
ax.bar_label(rects2)
ax.bar_label(rects3)

fig.tight_layout()

plt.show()
```



[6]:
```
train_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                  horizontal_flip = True,
                                  fill_mode = 'nearest',
                                  zoom_range=0.2,
                                  shear_range = 0.2,
                                  width_shift_range=0.2,
                                  height_shift_range=0.2,
                                  rotation_range=0.4)

train_data = train_datagen.flow_from_directory(train_path,
                                                   batch_size = 5,
                                                   target_size = (350,350),
                                                   class_mode = 'categorical')
```

```
Found 613 images belonging to 4 classes.
```

[7]:
```
train_data.class_indices
```

[7]:
```
{'adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib': 0,
 'large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa': 1,
 'normal': 2,
 'squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa': 3}
```

```
[8]: val_datagen = ImageDataGenerator(rescale = 1.0/255.0)
     val_data = val_datagen.flow_from_directory(val_path,
                                                batch_size = 5,
                                                target_size = (350,350),
                                                class_mode = 'categorical')
```

```
Found 72 images belonging to 4 classes.
```

```
[9]: val_data.class_indices
```

```
[9]: {'adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib': 0,
      'large.cell.carcinoma_left.hilum_T2_N2_M0_IIIa': 1,
      'normal': 2,
      'squamous.cell.carcinoma_left.hilum_T1_N2_M0_IIIa': 3}
```

```
[10]: test_datagen = ImageDataGenerator(rescale = 1.0/255.0)
      test_data = test_datagen.flow_from_directory(test_path,
                                                   batch_size = 5,
                                                   target_size = (350,350),
                                                   class_mode = 'categorical')
```

```
Found 315 images belonging to 4 classes.
```

```
[11]: test_data.class_indices
```

```
[11]: {'adenocarcinoma': 0,
       'large.cell.carcinoma': 1,
       'normal': 2,
       'squamous.cell.carcinoma': 3}
```

1 CNN MODEL

```
[12]: model = Sequential()

      # Convolutional Layer with input shape (350,350,3)
      model.add(Conv2D(filters=32, kernel_size= (3,3), activation= 'relu',␣
       ↪input_shape=(350,350,3)) )

      model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu' ))
      model.add(MaxPool2D(pool_size=(2,2)))

      model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu' ))
      model.add(MaxPool2D(pool_size=(2,2)))

      model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu' ))
      model.add(MaxPool2D(pool_size=(2,2)))
```

```python
model.add(Dropout(rate=0.25))

model.add(Flatten())
model.add(Dense(units=64, activation='relu'))
model.add(Dropout(rate=0.25))
model.add(Dense(units=4, activation='sigmoid'))

model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy']  )

model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 348, 348, 32)      896

 conv2d_1 (Conv2D)           (None, 346, 346, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 173, 173, 32)     0
 )

 conv2d_2 (Conv2D)           (None, 171, 171, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 85, 85, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 83, 83, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 41, 41, 128)      0
 2D)

 dropout (Dropout)           (None, 41, 41, 128)       0

 flatten (Flatten)           (None, 215168)            0

 dense (Dense)               (None, 64)                13770816

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 13,873,572
Trainable params: 13,873,572
Non-trainable params: 0
```

```
-----------------------------------------------------------------
```

[14]:
```python
# Adding Model check point Callback

mc = ModelCheckpoint(
    filepath="./ct_cnn_best_model.hdf5",
    monitor= 'val_accuracy',
    verbose= 1,
    save_best_only= True,
    mode = 'auto'
    );

call_back = [mc];
```

[15]:
```python
# Fitting the Model
cnn = model.fit(
    train_data,
    steps_per_epoch = train_data.samples//train_data.batch_size,
    epochs = 20,
    validation_data = val_data,
    validation_steps = val_data.samples//val_data.batch_size,
    callbacks = call_back
    )
```

```
Epoch 1/20
122/122 [==============================] - ETA: 0s - loss: 1.4438 - accuracy:
0.3454
Epoch 1: val_accuracy improved from -inf to 0.44286, saving model to
.\ct_cnn_best_model.hdf5
122/122 [==============================] - 137s 1s/step - loss: 1.4438 -
accuracy: 0.3454 - val_loss: 1.2634 - val_accuracy: 0.4429
Epoch 2/20
122/122 [==============================] - ETA: 0s - loss: 1.1116 - accuracy:
0.5066
Epoch 2: val_accuracy improved from 0.44286 to 0.50000, saving model to
.\ct_cnn_best_model.hdf5
122/122 [==============================] - 133s 1s/step - loss: 1.1116 -
accuracy: 0.5066 - val_loss: 0.9843 - val_accuracy: 0.5000
Epoch 3/20
122/122 [==============================] - ETA: 0s - loss: 1.0784 - accuracy:
0.5049
Epoch 3: val_accuracy did not improve from 0.50000
122/122 [==============================] - 120s 981ms/step - loss: 1.0784 -
accuracy: 0.5049 - val_loss: 1.0759 - val_accuracy: 0.4857
Epoch 4/20
122/122 [==============================] - ETA: 0s - loss: 1.0328 - accuracy:
0.5049
Epoch 4: val_accuracy did not improve from 0.50000
```

```
122/122 [==============================] - 116s 948ms/step - loss: 1.0328 -
accuracy: 0.5049 - val_loss: 1.1088 - val_accuracy: 0.4714
Epoch 5/20
122/122 [==============================] - ETA: 0s - loss: 1.0177 - accuracy:
0.5214
Epoch 5: val_accuracy did not improve from 0.50000
122/122 [==============================] - 128s 1s/step - loss: 1.0177 -
accuracy: 0.5214 - val_loss: 1.0287 - val_accuracy: 0.4714
Epoch 6/20
122/122 [==============================] - ETA: 0s - loss: 0.9819 - accuracy:
0.5411
Epoch 6: val_accuracy did not improve from 0.50000
122/122 [==============================] - 129s 1s/step - loss: 0.9819 -
accuracy: 0.5411 - val_loss: 1.0218 - val_accuracy: 0.4857
Epoch 7/20
122/122 [==============================] - ETA: 0s - loss: 0.9291 - accuracy:
0.5543
Epoch 7: val_accuracy improved from 0.50000 to 0.51429, saving model to
.\ct_cnn_best_model.hdf5
122/122 [==============================] - 121s 989ms/step - loss: 0.9291 -
accuracy: 0.5543 - val_loss: 0.9807 - val_accuracy: 0.5143
Epoch 8/20
122/122 [==============================] - ETA: 0s - loss: 0.8973 - accuracy:
0.5691
Epoch 8: val_accuracy improved from 0.51429 to 0.54286, saving model to
.\ct_cnn_best_model.hdf5
122/122 [==============================] - 125s 1s/step - loss: 0.8973 -
accuracy: 0.5691 - val_loss: 0.9185 - val_accuracy: 0.5429
Epoch 9/20
122/122 [==============================] - ETA: 0s - loss: 0.8883 - accuracy:
0.5773
Epoch 9: val_accuracy did not improve from 0.54286
122/122 [==============================] - 134s 1s/step - loss: 0.8883 -
accuracy: 0.5773 - val_loss: 0.9232 - val_accuracy: 0.5286
Epoch 10/20
122/122 [==============================] - ETA: 0s - loss: 0.8870 - accuracy:
0.5905
Epoch 10: val_accuracy improved from 0.54286 to 0.64286, saving model to
.\ct_cnn_best_model.hdf5
122/122 [==============================] - 128s 1s/step - loss: 0.8870 -
accuracy: 0.5905 - val_loss: 0.8894 - val_accuracy: 0.6429
Epoch 11/20
122/122 [==============================] - ETA: 0s - loss: 0.8897 - accuracy:
0.5806
Epoch 11: val_accuracy did not improve from 0.64286
122/122 [==============================] - 117s 953ms/step - loss: 0.8897 -
accuracy: 0.5806 - val_loss: 0.8833 - val_accuracy: 0.5714
Epoch 12/20
```

```
122/122 [==============================] - ETA: 0s - loss: 0.8643 - accuracy:
0.5921
Epoch 12: val_accuracy improved from 0.64286 to 0.68571, saving model to
.\ct_cnn_best_model.hdf5
122/122 [==============================] - 122s 998ms/step - loss: 0.8643 -
accuracy: 0.5921 - val_loss: 0.8247 - val_accuracy: 0.6857
Epoch 13/20
122/122 [==============================] - ETA: 0s - loss: 0.9131 - accuracy:
0.5987
Epoch 13: val_accuracy did not improve from 0.68571
122/122 [==============================] - 132s 1s/step - loss: 0.9131 -
accuracy: 0.5987 - val_loss: 0.8708 - val_accuracy: 0.6000
Epoch 14/20
122/122 [==============================] - ETA: 0s - loss: 0.8744 - accuracy:
0.5691
Epoch 14: val_accuracy did not improve from 0.68571
122/122 [==============================] - 122s 1s/step - loss: 0.8744 -
accuracy: 0.5691 - val_loss: 0.8999 - val_accuracy: 0.5714
Epoch 15/20
122/122 [==============================] - ETA: 0s - loss: 0.8378 - accuracy:
0.5872
Epoch 15: val_accuracy did not improve from 0.68571
122/122 [==============================] - 121s 993ms/step - loss: 0.8378 -
accuracy: 0.5872 - val_loss: 1.5529 - val_accuracy: 0.4143
Epoch 16/20
122/122 [==============================] - ETA: 0s - loss: 0.9665 - accuracy:
0.5559
Epoch 16: val_accuracy did not improve from 0.68571
122/122 [==============================] - 128s 1s/step - loss: 0.9665 -
accuracy: 0.5559 - val_loss: 0.8533 - val_accuracy: 0.6429
Epoch 17/20
122/122 [==============================] - ETA: 0s - loss: 0.8745 - accuracy:
0.5757
Epoch 17: val_accuracy did not improve from 0.68571
122/122 [==============================] - 129s 1s/step - loss: 0.8745 -
accuracy: 0.5757 - val_loss: 0.9163 - val_accuracy: 0.5000
Epoch 18/20
122/122 [==============================] - ETA: 0s - loss: 0.8335 - accuracy:
0.6003
Epoch 18: val_accuracy did not improve from 0.68571
122/122 [==============================] - 130s 1s/step - loss: 0.8335 -
accuracy: 0.6003 - val_loss: 1.0115 - val_accuracy: 0.5143
Epoch 19/20
122/122 [==============================] - ETA: 0s - loss: 0.8479 - accuracy:
0.6135
Epoch 19: val_accuracy did not improve from 0.68571
122/122 [==============================] - 130s 1s/step - loss: 0.8479 -
accuracy: 0.6135 - val_loss: 0.7970 - val_accuracy: 0.6429
```

```
Epoch 20/20
122/122 [==============================] - ETA: 0s - loss: 0.8141 - accuracy:
0.6184
Epoch 20: val_accuracy did not improve from 0.68571
122/122 [==============================] - 122s 995ms/step - loss: 0.8141 -
accuracy: 0.6184 - val_loss: 0.9194 - val_accuracy: 0.4857
```

[34]:
```python
# Loading the Best Fit Model
model = load_model("./ct_cnn_best_model.hdf5")
```

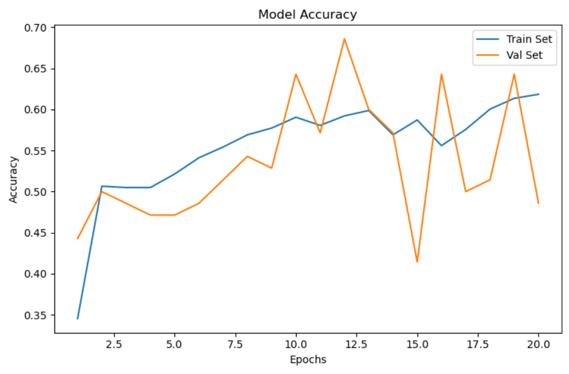
[35]:
```python
# Checking the Accuracy of the Model
accuracy_cnn = model.evaluate_generator(generator= test_data)[1]
print(f"The accuracy of the model is = {accuracy_cnn*100} %")
```

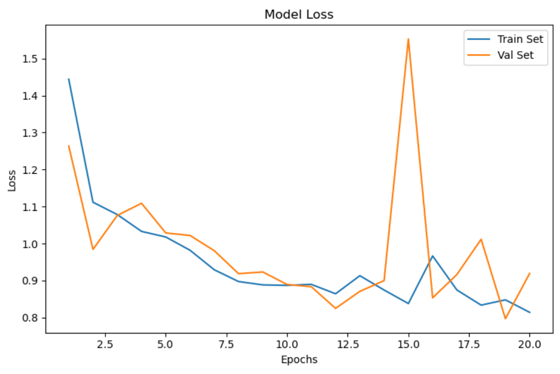
```
The accuracy of the model is = 50.47619342803955 %
```

[18]:
```python
cnn.history.keys()
```

[18]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

[19]:
```python
# Plot model performance
acc = cnn.history['accuracy']
val_acc = cnn.history['val_accuracy']
loss = cnn.history['loss']
val_loss = cnn.history['val_loss']
epochs_range = range(1, len(cnn.epoch) + 1)

plt.figure(figsize=(15,5))

plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Set')
plt.plot(epochs_range, val_acc, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Set')
plt.plot(epochs_range, val_loss, label='Val Set')
plt.legend(loc="best")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss')

plt.tight_layout()
plt.show()
```


Model Accuracy
Accuracy
Epochs
Train Set
Val Set
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Model Loss
Loss
Epochs
Train Set
Val Set
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0