week3-cnn-project

April 14, 2023

1 WEEK3_CNN_Project

1.1 Kavitha Sundaram

[1]:
```
# Basic libraries
import pandas as pd
import numpy as np
import os

# Plots
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image, ImageDraw
# Data processing, metrics and modeling
from sklearn.utils import resample
from sklearn.model_selection import train_test_split
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
import tensorflow as tf
print(tf.__version__)
#print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

from tensorflow import keras
from tensorflow.keras import datasets, layers, models
from keras.layers import Dense, Activation, Flatten, Dropout, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import PReLU
from keras.initializers import Constant
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers.legacy import Adam

import warnings
warnings.filterwarnings('ignore')
# Prints the current working directory
os.getcwd()
#changing my working directory as per project folder BBC files.
%cd "/Users/kavithasundaram/Documents/SKavitha/spring march-may 2023/DTSA-5511/
↪week3/cnn-project/histopathologic-cancer-detection"
```

```
2.12.0
/Users/kavithasundaram/Documents/SKavitha/spring march-may
2023/DTSA-5511/week3/cnn-project/histopathologic-cancer-detection
```

[2]:
```
#list of datafiles from kaggle dataset
os.listdir("./")
```

[2]:
```
['train_labels.csv',
 '.DS_Store',
 'test',
 'pngContainer',
 'cnn-kaggle.png',
 'train',
 'cnn-kaggle.csv',
 'sample_submission.csv']
```

[3]:
```
# Load in cnn data
cnn_train = pd.read_csv("./train_labels.csv")
display(cnn_train.info(),cnn_train.head(),cnn_train.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220025 entries, 0 to 220024
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   id      220025 non-null  object
 1   label   220025 non-null  int64
dtypes: int64(1), object(1)
memory usage: 3.4+ MB
```

```
None
```

```
                                         id  label
0  f38a6374c348f90b587e046aac6079959adf3835      0
1  c18f2d887b7ae4f6742ee445113fa1aef383ed77      1
2  755db6279dae599ebb4d39a9123cce439965282d      0
3  bc3f0c64fb968ff4a8bd33af6971ecae77c75e08      0
4  068aba587a4950175d04c680d38943fd488d6a9d      0
```

```
               label
count  220025.000000
mean        0.405031
std         0.490899
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         1.000000
```

1.2 Checking training and testing samples:

[4]:
```
train_cnn = './train/'
test_cnn = './test/'

print(f"There is {len(os.listdir(test_cnn))} training samples")
print(f"There is {len(os.listdir(train_cnn))} testing samples")
```

```
There is 57458 training samples
There is 220025 testing samples
```

[5]:
```
# Create a copy of train_labels datagrame
train = cnn_train.copy()

# Count per label
train_count = train['label'].value_counts()
train_count
```

[5]:
```
label
0    130908
1     89117
Name: count, dtype: int64
```

[6]:
```
can_cnn = cnn_train[cnn_train['label'] == 1]['id']
noncan_cnn = cnn_train[cnn_train['label'] == 0]['id']
```

1.3 1. Brief description of the problem and data (5 pts)

Briefly describe the challenge problem and NLP. Describe the size, dimension, structure, etc., of the data.

Size,Dimension,Structure of CNN dataset |

|——————————-|——————————————|

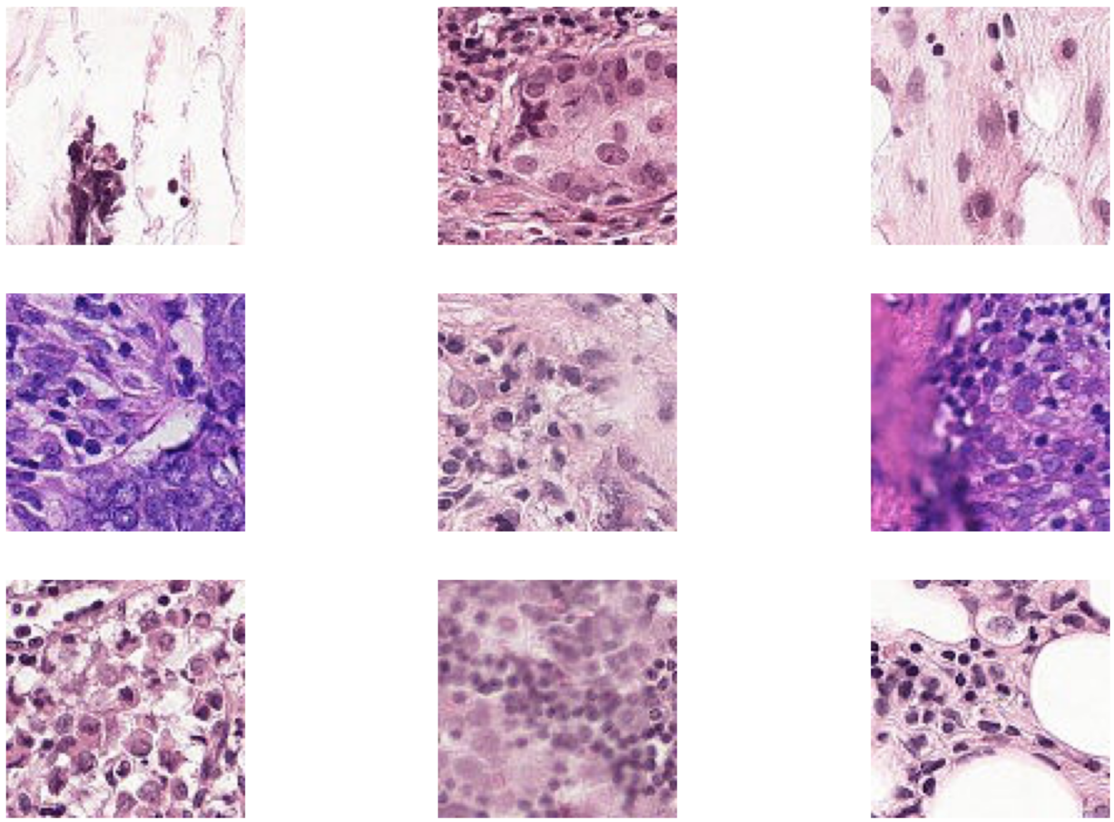
Training samples 57458 |Testing samples=220025 |
label 1 = 89117 = Cancerous = |label 0 = 130908 = 50.9% Non-Cancerous|

1.4 Exploratory Data Analysis (EDA) — Inspect, Visualize and Clean the Data (15 pts)

[7]:
```
fig, ax = plt.subplots(3, 3, figsize=(15,10))
fig.suptitle('Cancerous Samples', fontsize = 25)
count = 1
for index, name in enumerate(can_cnn.head(9)):
    file_name = 'train//'+ name + '.tif'
    img = Image.open(file_name)
    plt.subplot(3, 3, count)
    plt.imshow(np.array(img))
```

```
    plt.axis('off')
    count += 1
```

Cancerous Samples



```
[ ]: fig, ax = plt.subplots(3, 3, figsize=(15,10))
     fig.suptitle('Non-Cancerous Samples', fontsize = 25)
     count = 1
     for index, name in enumerate(noncan_cnn.head(9)):
         file_name = 'train//'+ name + '.tif'
         img = Image.open(file_name)
         plt.subplot(3, 3, count)
         plt.imshow(np.array(img))
         plt.axis('off')
         count += 1
```

1.5 DModel Architecture (25 pts)

1.6 Training and validation dataset

1.6.1 Creating basic CNN model to check the sequential and accuracy of the model.

Lets use Convolutional Neural Network (CNN) to classify our test images. Convolution is a useful tactic in dealing with images because, simply put, images have a lot of data (pixel height times pixel length times dimensions, oftentimes three). Because of this, a densely connected network will tend to run slow, and importantly, to overfit. Convolution networks try to overcome these shortcomings.

[12]:
```
#citation:https://stackoverflow.com/questions/42443936/
↪keras-split-train-test-set-when-using-imagedatagenerator
model = Sequential([
    layers.Conv2D(32, 3, padding='same', activation='relu', input_shape = (96,␣
↪96, 3)),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
    layers.Conv2D(128, 3, padding='same', activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(256, activation ='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(1, activation = "sigmoid")
])
```

```
Metal device set to: Apple M1 Pro
```

[13]:
```
from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau
from tensorflow.keras.callbacks import EarlyStopping
import time
cnn_train['label'] = cnn_train['label'].astype(str)
cnn_train['id'] = cnn_train['id'] + '.tif'
batch_size, im_size = 256, (96, 96)
generator = ImageDataGenerator(rescale = 1.0 / 255, validation_split = 0.1)
cnn_train_data = generator.flow_from_directory(batch_size=batch_size,
                                               directory='./',
                                               shuffle=True,
                                               target_size=im_size,
                                               subset="training",
```

```
                                                    class_mode='categorical')

cnn_valid_data = generator.flow_from_directory(batch_size=batch_size,
                                                    directory='./',
                                                    shuffle=True,
                                                    target_size=im_size,
                                                    subset="validation",
                                                    class_mode='categorical')
```

```
Found 249736 images belonging to 3 classes.
Found 27747 images belonging to 3 classes.
```

[14]:
```
model_opt = Adam(learning_rate = 0.0001)
model_epochs = 6

startTime = time.time()
model.compile(optimizer = model_opt, loss = 'binary_crossentropy', metrics =␣
 ↪['accuracy'])
history = model.fit(cnn_train_data, validation_data = cnn_valid_data, epochs =␣
 ↪model_epochs, steps_per_epoch = len(cnn_train_data) / 5)
endTime = time.time()
```

```
Epoch 1/6

2023-04-14 12:54:37.798883: W
tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU
frequency: 0 Hz

195/195 [==============================] - 37s 189ms/step - loss: 0.7087 -
accuracy: 0.5987 - val_loss: 1.7830 - val_accuracy: 0.6667
Epoch 2/6
195/195 [==============================] - 36s 186ms/step - loss: 0.6473 -
accuracy: 0.6606 - val_loss: 2.5728 - val_accuracy: 0.6651
Epoch 3/6
195/195 [==============================] - 37s 188ms/step - loss: 0.6425 -
accuracy: 0.6657 - val_loss: 0.8396 - val_accuracy: 0.6008
Epoch 4/6
195/195 [==============================] - 37s 187ms/step - loss: 0.6406 -
accuracy: 0.6665 - val_loss: 0.7185 - val_accuracy: 0.6049
Epoch 5/6
195/195 [==============================] - 37s 187ms/step - loss: 0.6396 -
accuracy: 0.6666 - val_loss: 0.6654 - val_accuracy: 0.6528
Epoch 6/6
195/195 [==============================] - 37s 187ms/step - loss: 0.6390 -
accuracy: 0.6666 - val_loss: 1.1334 - val_accuracy: 0.6655
```

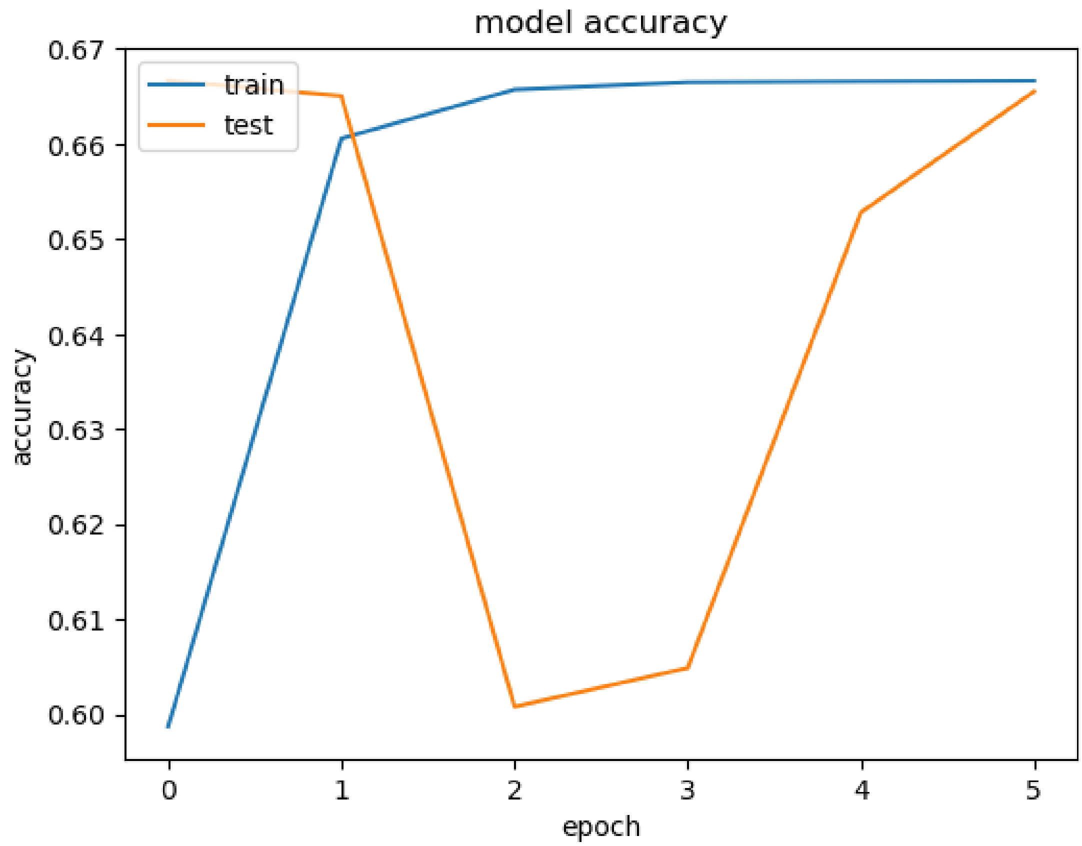
[18]:
```
#citation:https://www.kaggle.com/code/tobikaggle/keras-mnist-cnn-learning-curve
# plot learning curves
```

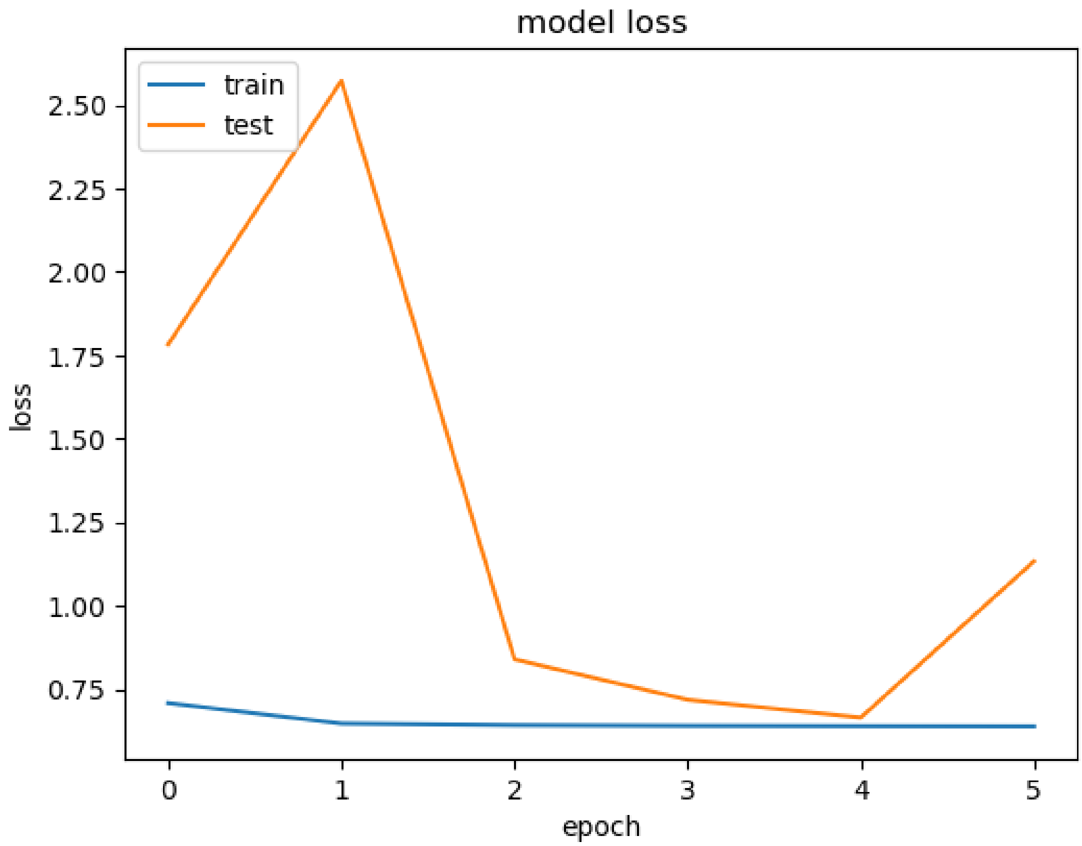
```python
print("\n")
print(history.history.keys())

# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

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


model accuracy
0.67
0.66
0.65
0.64
0.63
0.62
0.61
0.60
accuracy
train
test
0
1
2
3
4
5
epoch



[16]:
```
loss, accuracy = model.evaluate(cnn_valid_data)
print("Loss: ", loss)
print("Accuracy: ", accuracy)
```

```
109/109 [==============================] - 7s 63ms/step - loss: 1.1334 -
accuracy: 0.6655
Loss:  1.133424997329712
Accuracy:  0.6655254364013672
```

Model has accuracy of 66.5% and loss 1.33%

1.7 Prediction and analysis:

[17]:
```
# create the test dataframe
test_set = os.listdir(test_cnn)
test_df = pd.DataFrame(test_set)
test_df.columns = ['id']

# create the test subset
test_datagen=ImageDataGenerator(rescale=1/255)
```

```
test_generator=test_datagen.
 ↪flow_from_dataframe(dataframe=test_df,directory=test_cnn,
                x_col="id",batch_size=64,seed=1234,shuffle=False,
                class_mode=None,target_size=(96,96))
```

```
Found 57458 validated image filenames.
```

[19]:
```
# generate the prediction
size_test=test_generator.n/2
preds = model.predict_generator(generator=test_generator, steps=size_test,␣
 ↪verbose = 1)
# generate the labels with value 1 and 0
test_pred = []

for pred in preds:
    if pred >= 0.5:
        test_pred.append(1)
    else:
        test_pred.append(0)

test_pred[:10]
```

```
  897/28729 […] - ETA: 12:09WARNING:tensorflow:Your
input ran out of data; interrupting training. Make sure that your dataset or
generator can generate at least `steps_per_epoch * epochs` batches (in this
case, 28729.0 batches). You may need to use the repeat() function when building
your dataset.
28729/28729 [==============================] - 24s 836us/step
```

[19]:
```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

[20]:
```
# generate the submission file
submission = test_df.copy()
submission['id']=submission['id'].str[:-4]
submission['label']=test_pred
submission.head()
submission.to_csv('./cnn-kaggle.csv',index=False)
```

1.8 CONCLUSION:

There are many ways to fine tune the model, we can keep changing the architecture of the model to see if it can get a better result, but it is really time consuming.Traiuning and predicting CNN model is pretty tough job. It has overfitting problem and had to use more data to generalized the better model and fine tuning the dropout parameter to achieve better regularization to the network. Overall, it is challenging to train a model to identify metastatic cancer, data science is really bringing a greate future for human civilization.

1.8.1 GITHUB REPOSITORY URL

https://github.com/kavishant87/WEEK3-CNN-PROJECT