SVM_Models_VGG16

March 29, 2019

1 Import Library

In [1]:
```
import os
import pandas as pd
import numpy as np
import pickle
import time
# Machine Learning Algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score,recall_score,f1_score
from sklearn.metrics import precision_recall_fscore_support
from sklearn.model_selection import validation_curve,learning_curve
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support
from sklearn.svm import SVC
from joblib import dump,load
```

2 Loading Paths

In [2]:
```
train_path="A:\\Projects\\Major Project\\Extracted CNN Features\\VGG16\\train"
test_path="A:\\Projects\\Major Project\\Extracted CNN Features\\VGG16\\test"
```

In [3]:
```
param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]

param_grid = [{'svc__C': param_range,
               'svc__kernel': ['linear']},
              {'svc__C': param_range,
               'svc__gamma': ['auto'],
               'svc__kernel': ['rbf']}]
pipe_svc = make_pipeline(SVC(random_state=1))
```

In [4]:
```
# Training Paths
X_train=np.load(train_path+"\\data_cnn_VGG16_train.npy")
```

```
Y_train=np.load(train_path+"\\data_mag_VGG16_train.npy")
# Cancer class
cancerclass_train=np.load(train_path+"\\data_cancerclass_VGG16_train.npy")
# Cancer type
cancertype_train=np.load(train_path+"\\data_cancertype_VGG16_train.npy")
# Testing Paths
X_test=np.load(test_path+"\\data_cnn_VGG16_test.npy")
Y_test=np.load(test_path+"\\data_mag_VGG16_test.npy")
# Cancer class
cancerclass_test=np.load(test_path+"\\data_cancerclass_VGG16_test.npy")
# Cancer type
cancertype_test=np.load(test_path+"\\data_cancertype_VGG16_test.npy")
```

2.1 Magnification classification

```
In [18]: start_time = time.clock()
```

```
param_grid = [{'svc__C': param_range,
               'svc__kernel': ['linear']},
              {'svc__C': param_range,
               'svc__gamma': ['auto'],
               'svc__kernel': ['rbf']}]

gs = GridSearchCV(estimator=pipe_svc,
                  param_grid=param_grid,
                  scoring='accuracy',
                  cv=10,
                  n_jobs=-1)
gs = gs.fit(X_train, Y_train)
print(gs.best_score_)
print(gs.best_params_)
print(time.clock() - start_time, "seconds")
```

```
clf = gs.best_estimator_
clf.fit(X_train, y_train)
print('Test accuracy: %.3f' % clf.score(X_test, Y_test))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  """Entry point for launching an IPython kernel.


0.8313291139240506
{'svc__C': 0.001, 'svc__kernel': 'linear'}
```

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

NameError                                 Traceback (most recent call last)

<ipython-input-18-685885a6b1f9> in <module>
 23
 24 clf = gs.best_estimator_
---> 25 clf.fit(X_train, y_train)
 26 print('Test accuracy: %.3f' % clf.score(X_test, y_test))
 27 print(time.clock() - start_time, "seconds")


NameError: name 'y_train' is not defined
```

```
In [19]: clf = gs.best_estimator_
         clf.fit(X_train, Y_train)
         print('Test accuracy: %.3f' % clf.score(X_test, Y_test))
         print(time.clock() - start_time, "seconds")
```

```
Test accuracy: 0.902
28811.304737958 seconds


c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  after removing the cwd from sys.path.
```

```
In [20]: pred=clf.predict(X_test)

In [21]: con=confusion_matrix(Y_test,pred)

In [22]: print(con)

[[375  22   0   0]
 [ 15 347  30   0]
 [  5  18 344  27]
 [  1   0  37 359]]


In [23]: precision_recall_fscore_support(Y_test,pred)

Out[23]: (array([0.9469697 , 0.89664083, 0.83698297, 0.93005181]),
          array([0.94458438, 0.88520408, 0.87309645, 0.90428212]),
          array([0.94577554, 0.89088575, 0.85465839, 0.91698595]),
          array([397, 392, 394, 397], dtype=int64))
```

2.2 CancerClass Magnification Classification

In [6]:
```
Y_train_40=[]
X_train_40=[]

Y_train_100=[]
X_train_100=[]

Y_train_200=[]
X_train_200=[]

Y_train_400=[]
X_train_400=[]

for i in range(0,len(Y_train)):
    if(Y_train[i]==40):
        Y_train_40.append(cancerclass_train[i])
        X_train_40.append(X_train[i])
    if(Y_train[i]==100):
        Y_train_100.append(cancerclass_train[i])
        X_train_100.append(X_train[i])
    if(Y_train[i]==200):
        Y_train_200.append(cancerclass_train[i])
        X_train_200.append(X_train[i])
    if(Y_train[i]==400):
        Y_train_400.append(cancerclass_train[i])
        X_train_400.append(X_train[i])

X_train_40=np.array(X_train_40)
X_train_100=np.array(X_train_100)
X_train_200=np.array(X_train_200)
X_train_400=np.array(X_train_400)
Y_train_40=np.array(Y_train_40)
Y_train_100=np.array(Y_train_100)
Y_train_200=np.array(Y_train_200)
Y_train_400=np.array(Y_train_400)
print(Y_train_40.size)

Y_test_40=[]
X_test_40=[]

Y_test_100=[]
X_test_100=[]

Y_test_200=[]
X_test_200=[]

Y_test_400=[]
```

```
X_test_400=[]

for i in range(0,len(Y_test)):
    if(Y_test[i]==40):
        Y_test_40.append(cancerclass_test[i])
        X_test_40.append(X_test[i])
    if(Y_test[i]==100):
        Y_test_100.append(cancerclass_test[i])
        X_test_100.append(X_test[i])
    if(Y_test[i]==200):
        Y_test_200.append(cancerclass_test[i])
        X_test_200.append(X_test[i])
    if(Y_test[i]==400):
        Y_test_400.append(cancerclass_test[i])
        X_test_400.append(X_test[i])

X_test_40=np.array(X_test_40)
X_test_100=np.array(X_test_100)
X_test_200=np.array(X_test_200)
X_test_400=np.array(X_test_400)
Y_test_40=np.array(Y_test_40)
Y_test_100=np.array(Y_test_100)
Y_test_200=np.array(Y_test_200)
Y_test_400=np.array(Y_test_400)
```

```
1596
```

2.3 CancerClass Magnification Classification-40

```
In [5]: start_time = time.clock()
        pipe_svc = make_pipeline(SVC(random_state=1))

        param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]

        param_grid = [{'svc__C': param_range,
                       'svc__kernel': ['linear']},
                      {'svc__C': param_range,
                       'svc__gamma': ['auto'],
                       'svc__kernel': ['rbf']}]

        gs = GridSearchCV(estimator=pipe_svc,
                          param_grid=param_grid,
                          scoring='accuracy',
                          cv=10,
                          n_jobs=-1)
        gs = gs.fit(X_train_40, Y_train_40)
        print(gs.best_score_)
```

```
        print(gs.best_params_)
        print(time.clock() - start_time, "seconds")


        clf = gs.best_estimator_
        clf.fit(X_train_40, Y_train_40)
        print('Test accuracy: %.3f' % clf.score(X_test_40, Y_test_40))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  """Entry point for launching an IPython kernel.


0.8596491228070176
{'svc__C': 0.001, 'svc__kernel': 'linear'}
632.631926201 seconds


c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche


Test accuracy: 0.786
```

2.4 CancerClass Magnification Classification-100

```
In [7]: start_time = time.clock()
        pipe_svc = make_pipeline(SVC(random_state=1))

        param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]

        param_grid = [{'svc__C': param_range,
                       'svc__kernel': ['linear']},
                      {'svc__C': param_range,
                       'svc__gamma': ['auto'],
                       'svc__kernel': ['rbf']}]

        gs = GridSearchCV(estimator=pipe_svc,
                          param_grid=param_grid,
                          scoring='accuracy',
                          cv=10,
                          n_jobs=-1)
        gs = gs.fit(X_train_100, Y_train_100)
        print(gs.best_score_)
        print(gs.best_params_)
        print(time.clock() - start_time, "seconds")


        clf = gs.best_estimator_
        clf.fit(X_train_100, Y_train_100)
        print('Test accuracy: %.3f' % clf.score(X_test_100, Y_test_100))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  """Entry point for launching an IPython kernel.


0.8601066982809722
{'svc__C': 0.01, 'svc__kernel': 'linear'}
807.2974345060001 seconds


c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche


Test accuracy: 0.852


In [8]: print(gs.best_estimator_)

Pipeline(memory=None,
     steps=[('svc', SVC(C=0.01, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=1,
  shrinking=True, tol=0.001, verbose=False))])
```

2.5 CancerClass Magnification Classification-200

```
In [9]: tart_time = time.clock()
        pipe_svc = make_pipeline(SVC(random_state=1))

        param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]

        param_grid = [{'svc__C': param_range,
                       'svc__kernel': ['linear']},
                      {'svc__C': param_range,
                       'svc__gamma': ['auto'],
                       'svc__kernel': ['rbf']}]

        gs = GridSearchCV(estimator=pipe_svc,
                          param_grid=param_grid,
                          scoring='accuracy',
                          cv=10,
                          n_jobs=-1)
        gs = gs.fit(X_train_200, Y_train_200)
        print(gs.best_score_)
        print(gs.best_params_)
        print(time.clock() - start_time, "seconds")


        clf = gs.best_estimator_
```

```
        clf.fit(X_train_200, Y_train_200)
        print('Test accuracy: %.3f' % clf.score(X_test_200, Y_test_200))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  """Entry point for launching an IPython kernel.
```

```
0.8670377241805813
{'svc__C': 0.001, 'svc__kernel': 'linear'}
1584.021985761 seconds
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
```

```
Test accuracy: 0.820
```

2.6 CancerClass Magnification Classification-400

```
In [9]: start_time = time.clock()
        pipe_svc = make_pipeline(SVC(random_state=1))



        gs = GridSearchCV(estimator=pipe_svc,
                          param_grid=param_grid,
                          scoring='accuracy',
                          cv=10,
                          n_jobs=-1)
        gs = gs.fit(X_train_400, Y_train_400)
        print(gs.best_score_)
        print(gs.best_params_)
        print(time.clock() - start_time, "seconds")


        clf = gs.best_estimator_
        clf.fit(X_train_400, Y_train_400)
        print('Test accuracy: %.3f' % clf.score(X_test_400, Y_test_400))
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  """Entry point for launching an IPython kernel.
```

```
0.8549295774647887
{'svc__C': 0.001, 'svc__kernel': 'linear'}
536.3881946619999 seconds
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
```

```
Test accuracy: 0.796
```

2.7 Benign Sub-Classification Using Cancer Classification

```
In [ ]: Y_train_1=[]
        X_train_1=[]

        for i in range(0,len(Y_train)):
            if(cancerclass_train[i]==1):
                Y_train_1.append(cancertype_train[i])
                X_train_1.append(X_train[i])

        X_train_1=np.array(X_train_1)
        Y_train_1=np.array(Y_train_1)
        print(Y_train_1.size)

        Y_test_1=[]
        X_test_1=[]


        for i in range(0,len(Y_test)):
            if(cancerclass_test[i]==1):
                Y_test_1.append(cancertype_test[i])
                X_test_1.append(X_test[i])

        X_test_1=np.array(X_test_1)
        Y_test_1=np.array(Y_test_1)
```

```
In [ ]:
```

```
In [51]: classes=[11,12,13,14]
```

```
In [42]:
```

```
[1.]
```

```
In [52]: from sklearn.utils.class_weight import compute_class_weight
```

```
In [53]: class_weight=compute_class_weight(class_weight='balanced', classes=classes,y=Y_train_
```

```
In [54]: print(class_weight)
```

```
[1.66964286 0.51752768 1.67629482 1.14645777]
```

```
In [55]: print(np.unique(Y_train_1))
```

```
[11 12 13 14]
```

```
In [56]: print(len(X_train_1))
```

```
1683
```

```
In [57]: print(len(Y_test_1))
```

```
792
```

```
In [58]: d = dict(enumerate(class_weight, 1))
```

```
In [59]: print(d)
```

```
{1: 1.6696428571428572, 2: 0.5175276752767528, 3: 1.6762948207171315, 4: 1.146457765667575}
```

```
In [60]: d1={1:11,2:12,3:13,4:14}
```

```
In [61]: d=dict((d1[key], value) for (key, value) in d.items())
```

```
In [62]: d
```

```
Out[62]: {11: 1.6696428571428572,
          12: 0.5175276752767528,
          13: 1.6762948207171315,
          14: 1.146457765667575}
```

```
In [ ]:
```

```
In [ ]:
```

```
In [20]: pipe_svc = make_pipeline(SVC(random_state=1,class_weight=d))
         gs3=GridSearchCV(estimator=pipe_svc,
                          param_grid=param_grid,
                          scoring='accuracy',
                          cv=10,
                          n_jobs=-1)
         start_time = time.clock()
         #Training of Model
         gs3.fit(X_train_1,Y_train_1)
         print(time.clock() - start_time, "seconds")

         print(gs3.best_score_)
         print(gs3.best_params_)
```

```
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  import sys
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\model_sel
  DeprecationWarning)


1104.997306413 seconds
0.6934046345811051
{'svc__C': 0.01, 'svc__kernel': 'linear'}


c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  # Remove the CWD from sys.path while we load stuff.


In [21]: clf4=gs3.best_estimator_
         clf4.fit(X_train_1,Y_train_1)
         print(clf4.score(X_test_1,Y_test_1))

0.41414141414141414


In [63]: clf=SVC(C=.01,kernel='linear',class_weight=d)
         clf.fit(X_train_1,Y_train_1)
         print(clf.score(X_test_1,Y_test_1))

0.41414141414141414


In [64]: dump(clf,'models/SVM/SVM_Models_VGG16_CancerType_Benign.joblib')

Out[64]: ['models/SVM/SVM_Models_VGG16_CancerType_Benign.joblib']
```

2.8 Malignant Sub-Classification Using Cancer Classification

```
In [5]: Y_train_2=[]
        X_train_2=[]

        for i in range(0,len(Y_train)):
            if(cancerclass_train[i]==2):
                Y_train_2.append(cancertype_train[i])
                X_train_2.append(X_train[i])

        X_train_2=np.array(X_train_2)
        Y_train_2=np.array(Y_train_2)
        print(Y_train_2.size)

        Y_test_2=[]
        X_test_2=[]
```

```
        for i in range(0,len(Y_test)):
            if(cancerclass_test[i]==2):
                Y_test_2.append(cancertype_test[i])
                X_test_2.append(X_test[i])

        X_test_2=np.array(X_test_2)
        Y_test_2=np.array(Y_test_2)
```

```
4637
```

```
In [6]: classes=[21,22,23,24]
```

```
In [7]: from sklearn.utils.class_weight import compute_class_weight
```

```
In [8]: class_weight=compute_class_weight(class_weight='balanced', classes=classes,y=Y_train_2
```

```
In [9]: print(class_weight)
```

```
[0.35669231 2.72764706 1.96150592 3.12466307]
```

```
In [10]: print(np.unique(Y_train_2))
```

```
[21 22 23 24]
```

```
In [11]: print(len(X_train_2))
```

```
4637
```

```
In [12]: print(len(Y_test_2))
```

```
788
```

```
In [13]: d = dict(enumerate(class_weight, 1))
```

```
In [14]: print(d)
```

```
{1: 0.3566923076923077, 2: 2.7276470588235293, 3: 1.9615059221658206, 4: 3.1246630727762805}
```

```
In [15]: d1={1:21,2:22,3:23,4:24}
```

```
In [16]: d=dict((d1[key], value) for (key, value) in d.items())
```

```
In [17]: d
```

```
Out[17]: {21: 0.3566923076923077,
          22: 2.7276470588235293,
          23: 1.9615059221658206,
          24: 3.1246630727762805}

In [18]: pipe_svc = make_pipeline(SVC(random_state=1,class_weight=d))
         gs3=GridSearchCV(estimator=pipe_svc,
                          param_grid=param_grid,
                          scoring='accuracy',
                          cv=10,
                          n_jobs=-1)
         start_time = time.clock()
         #Training of Model
         gs3.fit(X_train_2,Y_train_2)
         print(time.clock() - start_time, "seconds")

         print(gs3.best_score_)
         print(gs3.best_params_)

c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  import sys
c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\model_se
  DeprecationWarning)


9759.513311783001 seconds
0.7088634893249947
{'svc__C': 10.0, 'svc__gamma': 'auto', 'svc__kernel': 'rbf'}


c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launche
  # Remove the CWD from sys.path while we load stuff.


In [19]: clf4=gs3.best_estimator_
         clf4.fit(X_train_2,Y_train_2)
         print(clf4.score(X_test_2,Y_test_2))

0.2779187817258883


In [20]: pred=clf4.predict(X_test_2)

In [21]: precision_recall_fscore_support(Y_test_2,pred)

c:\users\karan gupta\appdata\local\programs\python\python37\lib\site-packages\sklearn\metrics\
  'precision', 'predicted', average, warn_for)
```

```
Out[21]: (array([0.25945241, 1.        , 0.88888889, 0.        ]),
          array([0.995, 0.06 , 0.04 , 0.   ]),
          array([0.41158221, 0.11320755, 0.07655502, 0.        ]),
          array([200, 200, 200, 188], dtype=int64))

In [22]: confusion_matrix(Y_test_2,pred)

Out[22]: array([[199,   0,   1,   0],
                [188,  12,   0,   0],
                [192,   0,   8,   0],
                [188,   0,   0,   0]], dtype=int64)

In [49]: dump(clf4,'models/SVM/SVM_models_VGG16_CancerType_Malignant.joblib')

Out[49]: ['models/SVM/SVM_models_VGG16_CancerType_Malignant.joblib']
```

3 Dumping Models

```
In [27]: clf=SVC(C=.001,kernel='linear')
         clf.fit(X_train,Y_train)
         clf.score(X_test,Y_test)

Out[27]: 0.9018987341772152

In [28]: dump(clf,'models/SVM/SVM_models_VGG16_Magnification.joblib')

Out[28]: ['models/SVM/SVM_models_VGG16_Magnification.joblib']

In [40]: clf=SVC(C=.001,kernel='linear')
         clf.fit(X_train_40,Y_train_40)
         clf.score(X_test_40,Y_test_40)

Out[40]: 0.7858942065491183

In [41]: dump(clf,'models/SVM/SVM_models_VGG16_Magnification_40.joblib')

Out[41]: ['models/SVM/SVM_models_VGG16_Magnification_40.joblib']

In [42]: clf=SVC(C=.01,kernel='linear')
         clf.fit(X_train_100,Y_train_100)
         clf.score(X_test_100,Y_test_100)

Out[42]: 0.8520408163265306

In [43]: dump(clf,'models/SVM/SVM_models_VGG16_Magnification_100.joblib')

Out[43]: ['models/SVM/SVM_models_VGG16_Magnification_100.joblib']

In [44]: clf=SVC(C=.001,kernel='linear')
         clf.fit(X_train_200,Y_train_200)
         clf.score(X_test_200,Y_test_200)
```

```
Out[44]: 0.8197969543147208

In [45]: dump(clf,'models/SVM/SVM_models_VGG16_Magnification_200.joblib')

Out[45]: ['models/SVM/SVM_models_VGG16_Magnification_200.joblib']

In [46]: clf=SVC(C=.001,kernel='linear')
         clf.fit(X_train_400,Y_train_400)
         clf.score(X_test_400,Y_test_400)

Out[46]: 0.7959697732997482

In [47]: dump(clf,'models/SVM/SVM_models_VGG16_Magnification_400.joblib')

Out[47]: ['models/SVM/SVM_models_VGG16_Magnification_400.joblib']

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```