001191074

April 26, 2022

[1]:
```
import matplotlib.pyplot as plt
from PIL import Image
import seaborn as sns
import numpy as np
import pandas as pd
import os
from tensorflow.keras.utils import to_categorical
from glob import glob
```

[2]:
```
df = pd.read_csv('filename_to_category_map_surnamesIJKLM.csv')
```

[3]:
```
df
```

[3]:

	image_id	cell_type	is_benign	localization
0	ISIC_0027419	Benign keratosis-like lesions	1.0	scalp
1	ISIC_0026769	Benign keratosis-like lesions	1.0	scalp
2	ISIC_0031633	Benign keratosis-like lesions	1.0	ear
3	ISIC_0029176	Benign keratosis-like lesions	1.0	face
4	ISIC_0029068	Benign keratosis-like lesions	1.0	face
…	…	…	…	…
5371	ISIC_0032134	Melanocytic nevi	1.0	NaN
5372	ISIC_0032861	Benign keratosis-like lesions	1.0	NaN
5373	ISIC_0026054	Benign keratosis-like lesions	1.0	NaN
5374	ISIC_0032052	Melanoma	1.0	NaN
5375	ISIC_0031933	Benign keratosis-like lesions	1.0	NaN

[5376 rows x 4 columns]

[4]:
```
df.dtypes
```

[4]:
```
image_id         object
cell_type        object
is_benign       float64
localization     object
dtype: object
```

[5]:
```
df.describe()
```

[5]:
```
          is_benign
count  5276.000000
mean      0.807051
std       0.394651
min       0.000000
25%       1.000000
50%       1.000000
75%       1.000000
max       1.000000
```

[30]:
```
df.isnull().sum()
```

[30]:
```
image_id          0
cell_type       100
is_benign       100
localization     18
dtype: int64
```

[31]:
```
df1 = df.dropna()
df1.isnull().sum()
```

[31]:
```
image_id        0
cell_type       0
is_benign       0
localization    0
dtype: int64
```

[ ]:
```

```

[32]:
```
df1.isnull().sum()
df1.to_csv('exclusion_dataset_task2.csv')
```

[33]:
```
base_skin_dir = './ham_data_surnamesIJKLM'

imageid_path_dict = {os.path.splitext(os.path.basename(x))[0]: x
                     for x in glob(os.path.join(base_skin_dir, '*.jpg'))}
```

[34]:
```
df1['path'] = df1['image_id'].map(imageid_path_dict.get)
df1.head()
```

```
C:\Users\wajah\AppData\Local\Temp/ipykernel_20744/2121549243.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
df1['path'] = df1['image_id'].map(imageid_path_dict.get)
```

[34]:
```
        image_id                        cell_type  is_benign localization  \
0  ISIC_0027419  Benign keratosis-like lesions         1.0        scalp
1  ISIC_0026769  Benign keratosis-like lesions         1.0        scalp
2  ISIC_0031633  Benign keratosis-like lesions         1.0          ear
3  ISIC_0029176  Benign keratosis-like lesions         1.0         face
4  ISIC_0029068  Benign keratosis-like lesions         1.0         face

                                      path
0  ./ham_data_surnamesIJKLM\ISIC_0027419.jpg
1  ./ham_data_surnamesIJKLM\ISIC_0026769.jpg
2  ./ham_data_surnamesIJKLM\ISIC_0031633.jpg
3  ./ham_data_surnamesIJKLM\ISIC_0029176.jpg
4  ./ham_data_surnamesIJKLM\ISIC_0029068.jpg
```

[35]:
```
df1['image'] = df1['path'].map(lambda x: np.asarray(Image.open(x).
↪resize((125,100))))
```

```
C:\Users\wajah\AppData\Local\Temp/ipykernel_20744/3360847188.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['image'] = df1['path'].map(lambda x:
np.asarray(Image.open(x).resize((125,100))))
```

[ ]:
```
```

[36]:
```
n_samples = 5
fig, m_axs = plt.subplots(7, n_samples, figsize = (4*n_samples, 3*7))
for n_axs, (type_name, type_rows) in zip(m_axs,
                                         df1.sort_values(['cell_type']).
↪groupby('cell_type')):
    n_axs[0].set_title(type_name)
    for c_ax, (_, c_row) in zip(n_axs, type_rows.sample(n_samples,␣
↪random_state=2018).iterrows()):
        c_ax.imshow(c_row['image'])
        c_ax.axis('off')
fig.savefig('category_samples.png', dpi=300)
```

[ ]:
```
df1['image'].map(lambda x: x.shape).value_counts()
```

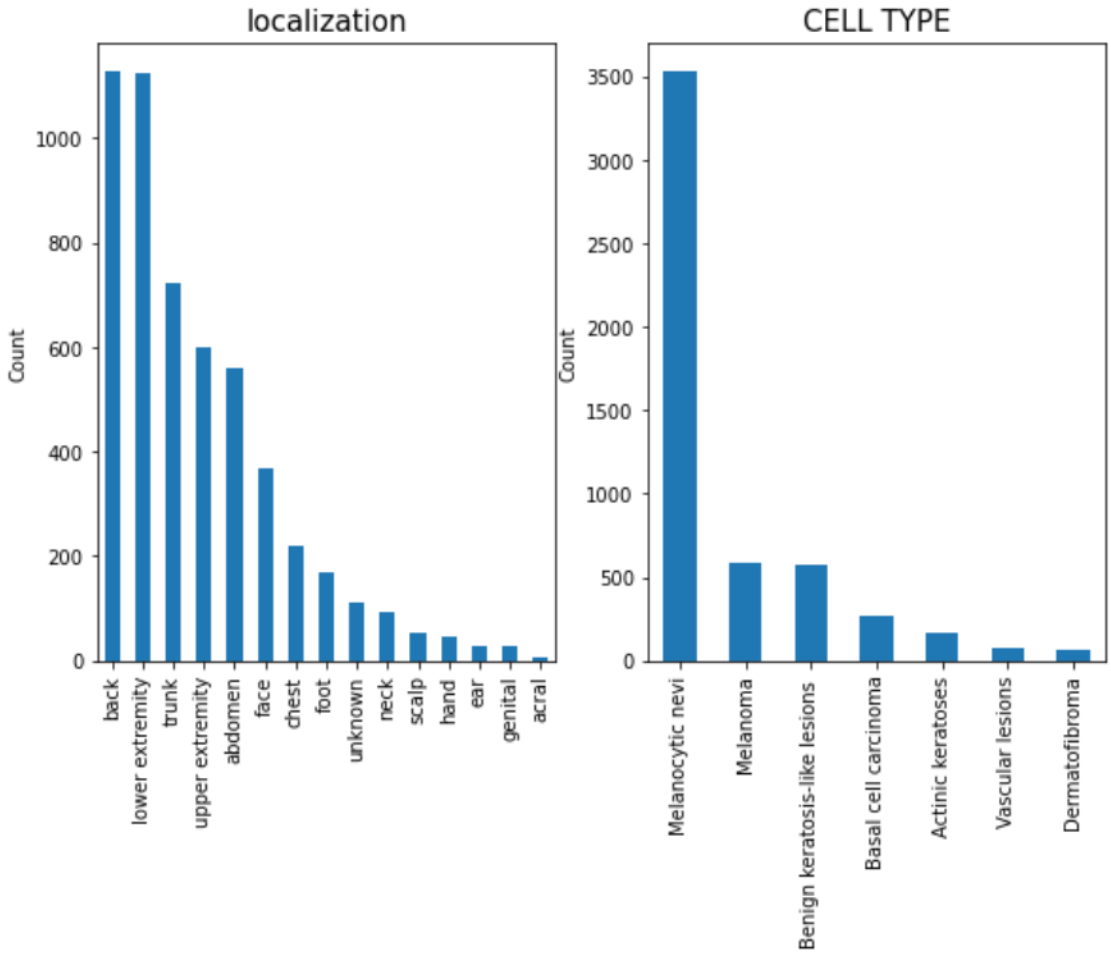
[ ]:
```
(100, 125, 3)    5258
Name: image, dtype: int64
```

[ ]:
```

```

[ ]:
```
plt.figure(figsize=(20,10))
plt.subplots_adjust(left=0.125, bottom=1, right=0.9, top=2, hspace=0.2)
```

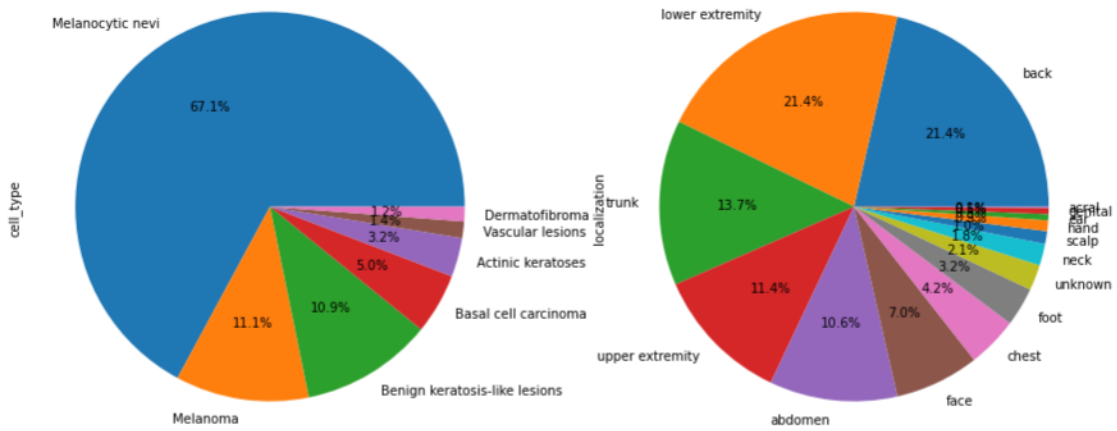
```
plt.subplot(2,4,3)
plt.title("localization",fontsize=15)
plt.ylabel("Count")
plt.xticks(rotation=45)
df1['localization'].value_counts().plot.bar()

plt.subplot(2,4,4)
plt.title("CELL TYPE",fontsize=15)
plt.ylabel("Count")
df1['cell_type'].value_counts().plot.bar()
```

[ ]: <AxesSubplot:title={'center':'CELL TYPE'}, ylabel='Count'>



[ ]:
```
plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
df1['cell_type'].value_counts().plot.pie(autopct="%1.1f%%")
plt.subplot(1,2,2)
```

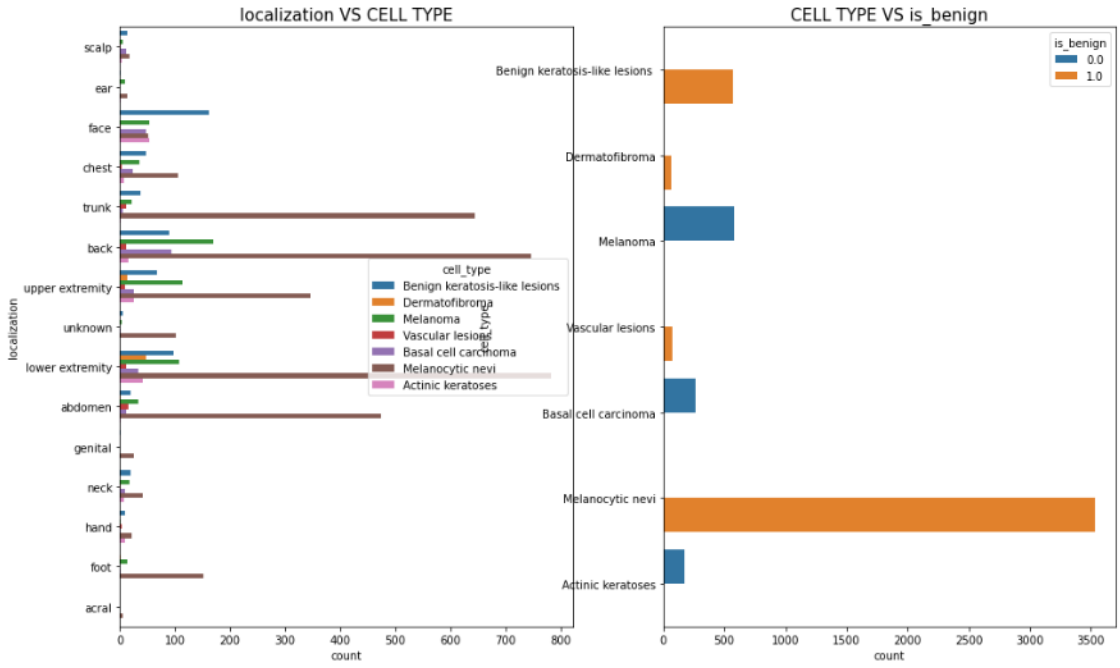
```
df1['localization'].value_counts().plot.pie(autopct="%1.1f%%")
plt.show()
```



- The face is infected the most by Benign keratosis-like lesions.
- Body parts(except face) are infected the most by Melanocytic nevi.

[ ]:
```
plt.figure(figsize=(25,10))
plt.subplot(131)
plt.title('localization VS CELL TYPE',fontsize = 15)
sns.countplot(y='localization', hue='cell_type',data=df1)
plt.subplot(132)
plt.title('CELL TYPE VS is_benign',fontsize = 15)
sns.countplot(y='cell_type', hue='is_benign',data=df1)
```

[ ]:
```
<AxesSubplot:title={'center':'CELL TYPE VS is_benign'}, xlabel='count',
ylabel='cell_type'>
```



[ ]:
```
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
```

0.1 ANN

[ ]:
```
features=df1.drop(columns=['is_benign'],axis=1)
target=df1['is_benign']
```

[ ]:
```
features.head()
```

[ ]:
```
        image_id                     cell_type localization  \
0  ISIC_0027419  Benign keratosis-like lesions         scalp
1  ISIC_0026769  Benign keratosis-like lesions         scalp
2  ISIC_0031633  Benign keratosis-like lesions           ear
3  ISIC_0029176  Benign keratosis-like lesions          face
4  ISIC_0029068  Benign keratosis-like lesions          face

                                  path  \
0  ham_data_surnamesIJKLM\ISIC_0027419.jpg
1  ham_data_surnamesIJKLM\ISIC_0026769.jpg
2  ham_data_surnamesIJKLM\ISIC_0031633.jpg
```

```
3  ham_data_surnamesIJKLM\ISIC_0029176.jpg
4  ham_data_surnamesIJKLM\ISIC_0029068.jpg

                                                image
0  [[[189, 152, 194], [192, 156, 198], [191, 154,…
1  [[[186, 127, 135], [189, 133, 145], [192, 135,…
2  [[[131, 88, 110], [142, 97, 120], [152, 107, 1…
3  [[[190, 144, 126], [192, 145, 130], [194, 146,…
4  [[[147, 103, 83], [153, 111, 91], [159, 119, 1…
```

[ ]:
```
x_train_o, x_test_o, y_train_o, y_test_o = train_test_split(features, target,␣
↪test_size=0.25,random_state=666)
tf.unique(x_train_o.cell_type.values)
```

[ ]:
```
Unique(y=<tf.Tensor: shape=(7,), dtype=string, numpy=
array([b'Melanocytic nevi', b'Basal cell carcinoma',
       b'Benign keratosis-like lesions ', b'Melanoma',
       b'Actinic keratoses', b'Dermatofibroma', b'Vascular lesions'],
      dtype=object)>, idx=<tf.Tensor: shape=(3943,), dtype=int32,
numpy=array([0, 0, 0, …, 0, 0, 0])>)
```

[ ]:
```
x_train = np.asarray(x_train_o['image'].tolist())
x_test = np.asarray(x_test_o['image'].tolist())

x_train_mean = np.mean(x_train)
x_train_std = np.std(x_train)

x_test_mean = np.mean(x_test)
x_test_std = np.std(x_test)

x_train = (x_train - x_train_mean)/x_train_std
x_test = (x_test - x_test_mean)/x_test_std
```

[ ]:
```
y_train = to_categorical(y_train_o, num_classes = 7)
y_test = to_categorical(y_test_o, num_classes = 7)
y_test
```

[ ]:
```
array([[0., 1., 0., …, 0., 0., 0.],
       [0., 1., 0., …, 0., 0., 0.],
       [0., 1., 0., …, 0., 0., 0.],
       …,
       [0., 1., 0., …, 0., 0., 0.],
       [0., 1., 0., …, 0., 0., 0.],
       [0., 1., 0., …, 0., 0., 0.]], dtype=float32)
```

[ ]:
```
x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train,␣
↪test_size = 0.1, random_state = 999)
```

```
x_train = x_train.reshape(x_train.shape[0], *(100, 125, 3))
x_test = x_test.reshape(x_test.shape[0], *(100, 125, 3))
x_validate = x_validate.reshape(x_validate.shape[0], *(100, 125, 3))
```

[ ]:
```
x_train = x_train.reshape(3548,125*100*3)
x_test = x_test.reshape(1315,125*100*3)
print(x_train.shape)
print(x_test.shape)
```

```
(3548, 37500)
(1315, 37500)
```

[ ]:
```
model = Sequential()

model.add(Dense(units= 64, kernel_initializer = 'uniform', activation = 'relu',␣
 ↪input_dim = 37500))
model.add(Dense(units= 64, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units= 64, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units= 64, kernel_initializer = 'uniform', activation = 'relu'))
model.add(Dense(units = 7, kernel_initializer = 'uniform', activation =␣
 ↪'softmax'))

optimizer = tf.keras.optimizers.Adam(learning_rate = 0.00075,
                                     beta_1 = 0.9,
                                     beta_2 = 0.999,
                                     epsilon = 1e-8)


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

history = model.fit(x_train, y_train, batch_size = 10, epochs = 50)

accuracy = model.evaluate(x_test, y_test, verbose=1)[1]
print("Test: accuracy = ",accuracy*100,"%")
```

```
Epoch 1/50
355/355 [==============================] - 8s 18ms/step - loss: 0.5011 -
accuracy: 0.8047
Epoch 2/50
355/355 [==============================] - 8s 21ms/step - loss: 0.4329 -
accuracy: 0.8075
Epoch 3/50
355/355 [==============================] - 8s 22ms/step - loss: 0.4060 -
accuracy: 0.8120 4s - loss: 0.3981 - accuracy: 0.81 - ETA: 4s - loss: 0.3
Epoch 4/50
355/355 [==============================] - 9s 25ms/step - loss: 0.3898 -
```

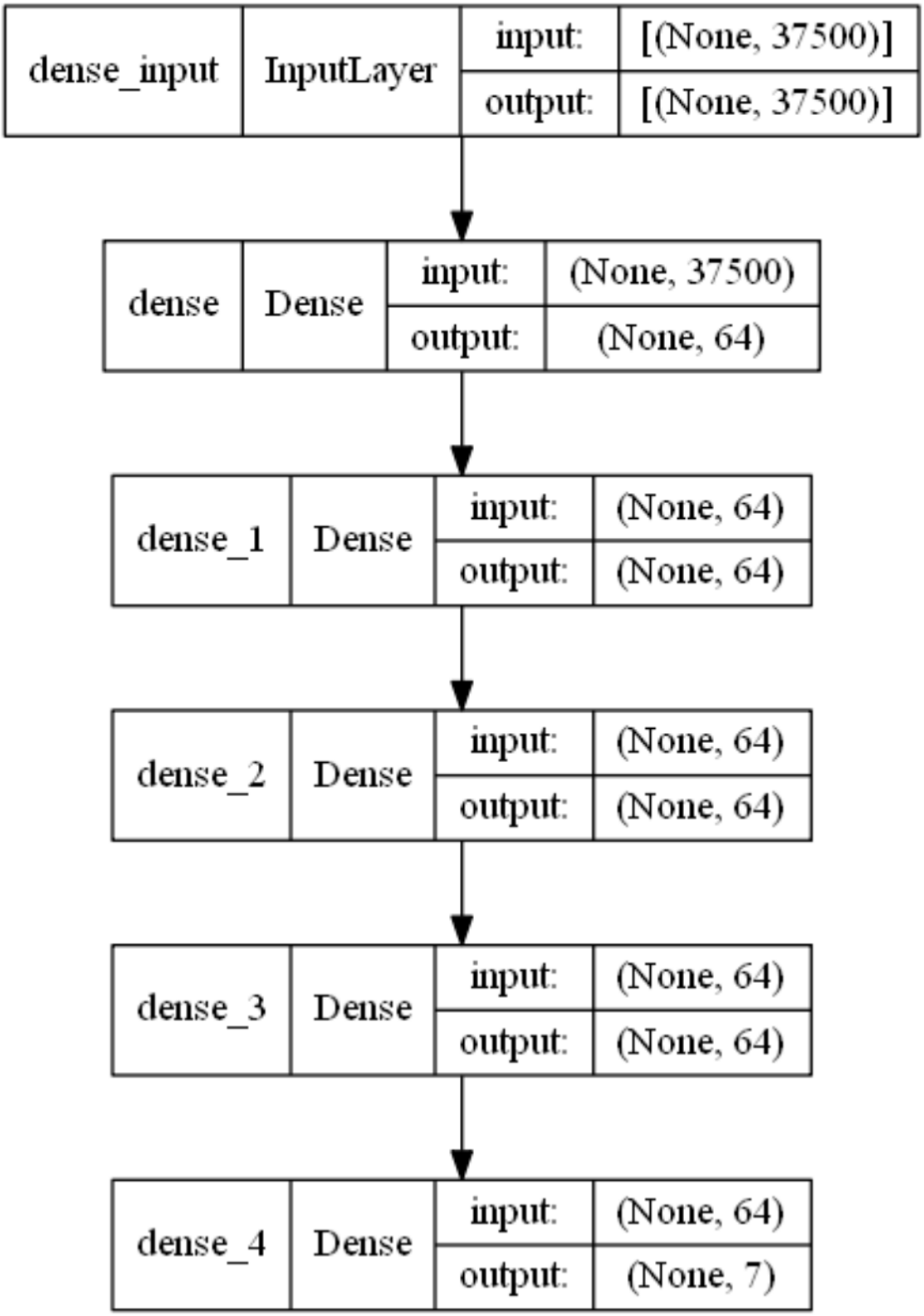
```
accuracy: 0.8247
Epoch 5/50
355/355 [==============================] - 9s 24ms/step - loss: 0.3829 -
accuracy: 0.8269
Epoch 6/50
355/355 [==============================] - 9s 25ms/step - loss: 0.3559 -
accuracy: 0.8391
Epoch 7/50
355/355 [==============================] - 7s 20ms/step - loss: 0.3482 -
accuracy: 0.8464
Epoch 8/50
355/355 [==============================] - 9s 24ms/step - loss: 0.3309 -
accuracy: 0.8591
Epoch 9/50
355/355 [==============================] - 9s 25ms/step - loss: 0.3120 -
accuracy: 0.8630
Epoch 10/50
355/355 [==============================] - 9s 24ms/step - loss: 0.3024 -
accuracy: 0.8701
Epoch 11/50
355/355 [==============================] - 9s 25ms/step - loss: 0.2753 -
accuracy: 0.8842
Epoch 12/50
355/355 [==============================] - 8s 22ms/step - loss: 0.2632 -
accuracy: 0.8943
Epoch 13/50
355/355 [==============================] - 6s 17ms/step - loss: 0.2401 -
accuracy: 0.9042
Epoch 14/50
355/355 [==============================] - 6s 17ms/step - loss: 0.2369 -
accuracy: 0.9030
Epoch 15/50
355/355 [==============================] - 6s 17ms/step - loss: 0.2137 -
accuracy: 0.9169
Epoch 16/50
355/355 [==============================] - 7s 20ms/step - loss: 0.2193 -
accuracy: 0.9101
Epoch 17/50
355/355 [==============================] - 6s 18ms/step - loss: 0.1817 -
accuracy: 0.9304
Epoch 18/50
355/355 [==============================] - 6s 17ms/step - loss: 0.1892 -
accuracy: 0.9278
Epoch 19/50
355/355 [==============================] - 6s 18ms/step - loss: 0.1669 -
accuracy: 0.9346
Epoch 20/50
355/355 [==============================] - 7s 20ms/step - loss: 0.1586 -
```

```
accuracy: 0.9360
Epoch 21/50
355/355 [==============================] - 6s 16ms/step - loss: 0.1533 -
accuracy: 0.9394
Epoch 22/50
355/355 [==============================] - 6s 18ms/step - loss: 0.1379 -
accuracy: 0.9490
Epoch 23/50
355/355 [==============================] - 6s 17ms/step - loss: 0.1384 -
accuracy: 0.9467
Epoch 24/50
355/355 [==============================] - 6s 17ms/step - loss: 0.1369 -
accuracy: 0.9467
Epoch 25/50
355/355 [==============================] - 6s 17ms/step - loss: 0.1268 -
accuracy: 0.9512
Epoch 26/50
355/355 [==============================] - 6s 16ms/step - loss: 0.1239 -
accuracy: 0.9498
Epoch 27/50
355/355 [==============================] - 6s 17ms/step - loss: 0.1092 -
accuracy: 0.9583
Epoch 28/50
355/355 [==============================] - 6s 16ms/step - loss: 0.1105 -
accuracy: 0.9583
Epoch 29/50
355/355 [==============================] - 6s 17ms/step - loss: 0.0962 -
accuracy: 0.9636
Epoch 30/50
355/355 [==============================] - 6s 17ms/step - loss: 0.0931 -
accuracy: 0.9673
Epoch 31/50
355/355 [==============================] - 6s 18ms/step - loss: 0.0999 -
accuracy: 0.9636
Epoch 32/50
355/355 [==============================] - 7s 18ms/step - loss: 0.0807 -
accuracy: 0.9710
Epoch 33/50
355/355 [==============================] - 6s 17ms/step - loss: 0.0750 -
accuracy: 0.9690
Epoch 34/50
355/355 [==============================] - 6s 17ms/step - loss: 0.0884 -
accuracy: 0.9676
Epoch 35/50
355/355 [==============================] - 6s 17ms/step - loss: 0.0882 -
accuracy: 0.9713
Epoch 36/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0648 -
```

```
accuracy: 0.9741
Epoch 37/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0952 -
accuracy: 0.9696
Epoch 38/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0634 -
accuracy: 0.9766
Epoch 39/50
355/355 [==============================] - 5s 15ms/step - loss: 0.0766 -
accuracy: 0.9732
Epoch 40/50
355/355 [==============================] - 5s 15ms/step - loss: 0.0669 -
accuracy: 0.9760
Epoch 41/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0727 -
accuracy: 0.9797
Epoch 42/50
355/355 [==============================] - 5s 14ms/step - loss: 0.0797 -
accuracy: 0.9760
Epoch 43/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0624 -
accuracy: 0.9820
Epoch 44/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0586 -
accuracy: 0.9794
Epoch 45/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0720 -
accuracy: 0.9752
Epoch 46/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0561 -
accuracy: 0.9825
Epoch 47/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0596 -
accuracy: 0.9763
Epoch 48/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0568 -
accuracy: 0.9822
Epoch 49/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0632 -
accuracy: 0.9780 0s - loss: 0.0
Epoch 50/50
355/355 [==============================] - 6s 16ms/step - loss: 0.0777 -
accuracy: 0.9735
42/42 [==============================] - 1s 7ms/step - loss: 1.2581 - accuracy:
0.8205
Test: accuracy =  82.05323219299316 %
```

[ ]:
```
from keras.utils.vis_utils import plot_model
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True,␣
↪show_layer_names=True)
```

[ ]:


dense_input
InputLayer
input: [(None, 37500)]
output: [(None, 37500)]
dense
Dense
input: (None, 37500)
output: (None, 64)
dense_1
Dense
input: (None, 64)
output: (None, 64)
dense_2
Dense
input: (None, 64)
output: (None, 64)
dense_3
Dense
input: (None, 64)
output: (None, 64)
dense_4
Dense
input: (None, 64)
output: (None, 7)

0.2 CNN

[ ]:
```
from tensorflow.keras.layers import␣
↪Flatten,Dense,Dropout,BatchNormalization,Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[ ]:
```
input_shape = (100, 125, 3)
num_classes = 7

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding =␣
↪'Same',input_shape=input_shape))
model.add(Conv2D(32,kernel_size=(3, 3), activation='relu',padding = 'Same',))
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(0.16))

model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding = 'Same'))
model.add(Conv2D(32,kernel_size=(3, 3), activation='relu',padding = 'Same',))
model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(0.20))

model.add(Conv2D(64, (3, 3), activation='relu',padding = 'same'))
model.add(Conv2D(64, (3, 3), activation='relu',padding = 'Same'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 125, 32)      896

 conv2d_1 (Conv2D)           (None, 100, 125, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 50, 62, 32)       0
 )

 dropout (Dropout)           (None, 50, 62, 32)        0
```

```
conv2d_2 (Conv2D)            (None, 50, 62, 32)        9248

conv2d_3 (Conv2D)            (None, 50, 62, 32)        9248

max_pooling2d_1 (MaxPooling  (None, 25, 31, 32)        0
2D)

dropout_1 (Dropout)          (None, 25, 31, 32)        0

conv2d_4 (Conv2D)            (None, 25, 31, 64)        18496

conv2d_5 (Conv2D)            (None, 25, 31, 64)        36928

max_pooling2d_2 (MaxPooling  (None, 12, 15, 64)        0
2D)

dropout_2 (Dropout)          (None, 12, 15, 64)        0

flatten (Flatten)            (None, 11520)             0

dense_5 (Dense)              (None, 256)               2949376

dense_6 (Dense)              (None, 128)               32896

dropout_3 (Dropout)          (None, 128)               0

dense_7 (Dense)              (None, 7)                 903

=================================================================
Total params: 3,067,239
Trainable params: 3,067,239
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
optimizer = Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0,␣
↪amsgrad=False)
```

```
C:\Users\wajah\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105:
UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

[ ]:
```
model.compile(optimizer = optimizer , loss = "categorical_crossentropy",␣
↪metrics=["accuracy"])
```

[ ]:
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                            patience=4,
                                            verbose=1,
```

```
                                            factor=0.5,
                                            min_lr=0.00001)
```

[ ]:
```
x_train, x_validate, y_train, y_validate = train_test_split(x_train, y_train,␣
 ↪test_size = 0.1, random_state = 999)
x_train = x_train.reshape(x_train.shape[0], *(100, 125, 3))
x_test = x_test.reshape(x_test.shape[0], *(100, 125, 3))
x_validate = x_validate.reshape(x_validate.shape[0], *(100, 125, 3))

datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=10,
        zoom_range = 0.1,
        width_shift_range=0.12,
        height_shift_range=0.12,
        horizontal_flip=True,
        vertical_flip=True)

datagen.fit(x_train)
```

[ ]:
```
epochs = 60
batch_size = 16
history = model.fit_generator(datagen.flow(x_train,y_train,␣
 ↪batch_size=batch_size),
                              epochs = epochs, validation_data =␣
 ↪(x_validate,y_validate),
                              verbose = 1, steps_per_epoch=x_train.shape[0] //␣
 ↪batch_size
                              , callbacks=[learning_rate_reduction])

from tensorflow.keras.metrics import Recall
from sklearn.metrics import classification_report,confusion_matrix
```

```
C:\Users\wajah\AppData\Local\Temp/ipykernel_7072/87963123.py:4: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  history = model.fit_generator(datagen.flow(x_train,y_train,
batch_size=batch_size),

Epoch 1/60
199/199 [==============================] - 126s 629ms/step - loss: 0.5866 -
accuracy: 0.7919 - val_loss: 0.5652 - val_accuracy: 0.8113 - lr: 1.0000e-04
Epoch 2/60
```

```
199/199 [==============================] - 113s 568ms/step - loss: 0.5008 -
accuracy: 0.8023 - val_loss: 0.4964 - val_accuracy: 0.8113 - lr: 1.0000e-04
Epoch 3/60
199/199 [==============================] - 113s 570ms/step - loss: 0.4727 -
accuracy: 0.8017 - val_loss: 0.5108 - val_accuracy: 0.8113 - lr: 1.0000e-04
Epoch 4/60
199/199 [==============================] - 113s 569ms/step - loss: 0.4502 -
accuracy: 0.8011 - val_loss: 0.4127 - val_accuracy: 0.8113 - lr: 1.0000e-04
Epoch 5/60
199/199 [==============================] - ETA: 0s - loss: 0.4285 - accuracy:
0.8061
Epoch 00005: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
199/199 [==============================] - 112s 565ms/step - loss: 0.4285 -
accuracy: 0.8061 - val_loss: 0.4019 - val_accuracy: 0.8113 - lr: 1.0000e-04
Epoch 6/60
199/199 [==============================] - 113s 567ms/step - loss: 0.4099 -
accuracy: 0.8067 - val_loss: 0.3872 - val_accuracy: 0.8085 - lr: 5.0000e-05
Epoch 7/60
199/199 [==============================] - 113s 569ms/step - loss: 0.4052 -
accuracy: 0.8001 - val_loss: 0.3824 - val_accuracy: 0.8197 - lr: 5.0000e-05
Epoch 8/60
199/199 [==============================] - 113s 568ms/step - loss: 0.4018 -
accuracy: 0.8111 - val_loss: 0.3852 - val_accuracy: 0.8085 - lr: 5.0000e-05
Epoch 9/60
199/199 [==============================] - 113s 568ms/step - loss: 0.3900 -
accuracy: 0.8061 - val_loss: 0.3776 - val_accuracy: 0.8197 - lr: 5.0000e-05
Epoch 10/60
199/199 [==============================] - 113s 567ms/step - loss: 0.3901 -
accuracy: 0.8048 - val_loss: 0.3829 - val_accuracy: 0.8085 - lr: 5.0000e-05
Epoch 11/60
199/199 [==============================] - ETA: 0s - loss: 0.3885 - accuracy:
0.8149
Epoch 00011: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.
199/199 [==============================] - 113s 569ms/step - loss: 0.3885 -
accuracy: 0.8149 - val_loss: 0.3723 - val_accuracy: 0.8169 - lr: 5.0000e-05
Epoch 12/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3902 -
accuracy: 0.8130 - val_loss: 0.3709 - val_accuracy: 0.8225 - lr: 2.5000e-05
Epoch 13/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3796 -
accuracy: 0.8118 - val_loss: 0.3708 - val_accuracy: 0.8169 - lr: 2.5000e-05
Epoch 14/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3820 -
accuracy: 0.8137 - val_loss: 0.3743 - val_accuracy: 0.8141 - lr: 2.5000e-05
Epoch 15/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3781 -
accuracy: 0.8171 - val_loss: 0.3644 - val_accuracy: 0.8366 - lr: 2.5000e-05
Epoch 16/60
```

```
199/199 [==============================] - 112s 561ms/step - loss: 0.3768 -
accuracy: 0.8174 - val_loss: 0.3711 - val_accuracy: 0.8141 - lr: 2.5000e-05
Epoch 17/60
199/199 [==============================] - 112s 562ms/step - loss: 0.3777 -
accuracy: 0.8174 - val_loss: 0.3614 - val_accuracy: 0.8338 - lr: 2.5000e-05
Epoch 18/60
199/199 [==============================] - 112s 565ms/step - loss: 0.3723 -
accuracy: 0.8196 - val_loss: 0.3674 - val_accuracy: 0.8310 - lr: 2.5000e-05
Epoch 19/60
199/199 [==============================] - ETA: 0s - loss: 0.3727 - accuracy:
0.8149
Epoch 00019: ReduceLROnPlateau reducing learning rate to 1.249999968422344e-05.
199/199 [==============================] - 112s 565ms/step - loss: 0.3727 -
accuracy: 0.8149 - val_loss: 0.3679 - val_accuracy: 0.8338 - lr: 2.5000e-05
Epoch 20/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3721 -
accuracy: 0.8155 - val_loss: 0.3587 - val_accuracy: 0.8394 - lr: 1.2500e-05
Epoch 21/60
199/199 [==============================] - 112s 562ms/step - loss: 0.3659 -
accuracy: 0.8140 - val_loss: 0.3718 - val_accuracy: 0.8310 - lr: 1.2500e-05
Epoch 22/60
199/199 [==============================] - 112s 562ms/step - loss: 0.3657 -
accuracy: 0.8171 - val_loss: 0.3630 - val_accuracy: 0.8310 - lr: 1.2500e-05
Epoch 23/60
199/199 [==============================] - 120s 604ms/step - loss: 0.3673 -
accuracy: 0.8162 - val_loss: 0.3622 - val_accuracy: 0.8310 - lr: 1.2500e-05
Epoch 24/60
199/199 [==============================] - ETA: 0s - loss: 0.3641 - accuracy:
0.8209
Epoch 00024: ReduceLROnPlateau reducing learning rate to 1e-05.
199/199 [==============================] - 113s 565ms/step - loss: 0.3641 -
accuracy: 0.8209 - val_loss: 0.3600 - val_accuracy: 0.8394 - lr: 1.2500e-05
Epoch 25/60
199/199 [==============================] - 112s 561ms/step - loss: 0.3598 -
accuracy: 0.8294 - val_loss: 0.3596 - val_accuracy: 0.8366 - lr: 1.0000e-05
Epoch 26/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3692 -
accuracy: 0.8196 - val_loss: 0.3582 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 27/60
199/199 [==============================] - 113s 565ms/step - loss: 0.3622 -
accuracy: 0.8234 - val_loss: 0.3592 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 28/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3641 -
accuracy: 0.8137 - val_loss: 0.3592 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 29/60
199/199 [==============================] - 113s 565ms/step - loss: 0.3622 -
accuracy: 0.8234 - val_loss: 0.3700 - val_accuracy: 0.8338 - lr: 1.0000e-05
Epoch 30/60
```

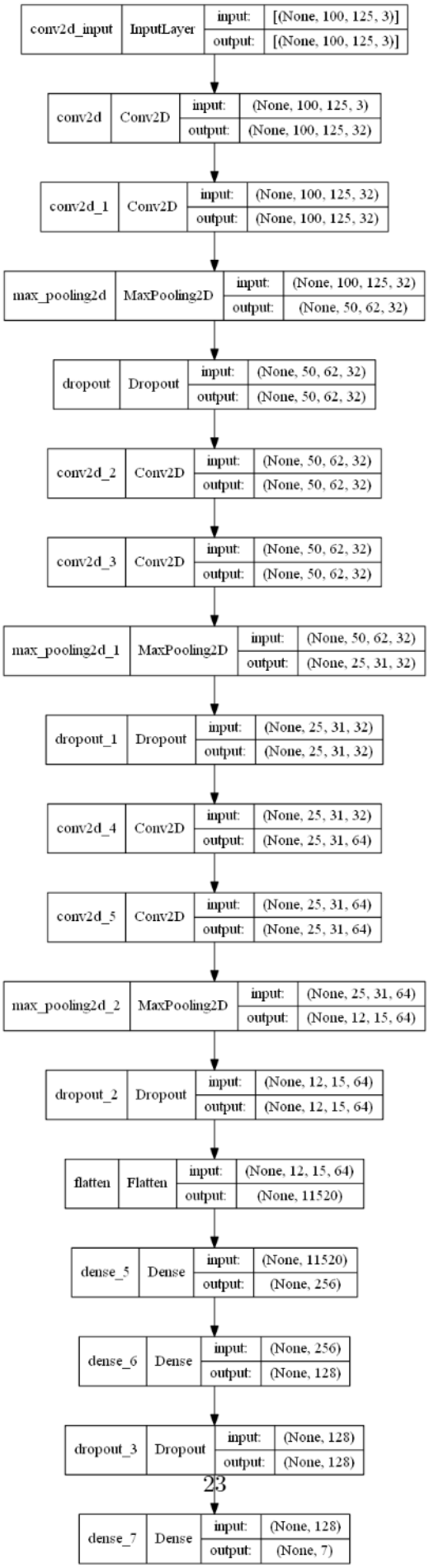
```
199/199 [==============================] - 112s 562ms/step - loss: 0.3617 -
accuracy: 0.8209 - val_loss: 0.3536 - val_accuracy: 0.8366 - lr: 1.0000e-05
Epoch 31/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3593 -
accuracy: 0.8181 - val_loss: 0.3534 - val_accuracy: 0.8366 - lr: 1.0000e-05
Epoch 32/60
199/199 [==============================] - 112s 561ms/step - loss: 0.3559 -
accuracy: 0.8228 - val_loss: 0.3563 - val_accuracy: 0.8338 - lr: 1.0000e-05
Epoch 33/60
199/199 [==============================] - 115s 578ms/step - loss: 0.3569 -
accuracy: 0.8234 - val_loss: 0.3570 - val_accuracy: 0.8366 - lr: 1.0000e-05
Epoch 34/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3567 -
accuracy: 0.8222 - val_loss: 0.3615 - val_accuracy: 0.8366 - lr: 1.0000e-05
Epoch 35/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3562 -
accuracy: 0.8225 - val_loss: 0.3546 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 36/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3597 -
accuracy: 0.8190 - val_loss: 0.3619 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 37/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3575 -
accuracy: 0.8234 - val_loss: 0.3555 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 38/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3538 -
accuracy: 0.8291 - val_loss: 0.3603 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 39/60
199/199 [==============================] - 113s 566ms/step - loss: 0.3575 -
accuracy: 0.8228 - val_loss: 0.3568 - val_accuracy: 0.8479 - lr: 1.0000e-05
Epoch 40/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3561 -
accuracy: 0.8250 - val_loss: 0.3585 - val_accuracy: 0.8451 - lr: 1.0000e-05
Epoch 41/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3551 -
accuracy: 0.8215 - val_loss: 0.3522 - val_accuracy: 0.8479 - lr: 1.0000e-05
Epoch 42/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3551 -
accuracy: 0.8275 - val_loss: 0.3569 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 43/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3529 -
accuracy: 0.8247 - val_loss: 0.3566 - val_accuracy: 0.8479 - lr: 1.0000e-05
Epoch 44/60
199/199 [==============================] - 112s 564ms/step - loss: 0.3560 -
accuracy: 0.8253 - val_loss: 0.3608 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 45/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3550 -
accuracy: 0.8237 - val_loss: 0.3556 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 46/60
```

```
199/199 [==============================] - 112s 563ms/step - loss: 0.3488 -
accuracy: 0.8341 - val_loss: 0.3608 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 47/60
199/199 [==============================] - 112s 561ms/step - loss: 0.3505 -
accuracy: 0.8332 - val_loss: 0.3534 - val_accuracy: 0.8451 - lr: 1.0000e-05
Epoch 48/60
199/199 [==============================] - 111s 558ms/step - loss: 0.3505 -
accuracy: 0.8250 - val_loss: 0.3497 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 49/60
199/199 [==============================] - 111s 560ms/step - loss: 0.3505 -
accuracy: 0.8281 - val_loss: 0.3543 - val_accuracy: 0.8451 - lr: 1.0000e-05
Epoch 50/60
199/199 [==============================] - 111s 559ms/step - loss: 0.3452 -
accuracy: 0.8341 - val_loss: 0.3667 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 51/60
199/199 [==============================] - 111s 559ms/step - loss: 0.3517 -
accuracy: 0.8269 - val_loss: 0.3552 - val_accuracy: 0.8507 - lr: 1.0000e-05
Epoch 52/60
199/199 [==============================] - 111s 560ms/step - loss: 0.3481 -
accuracy: 0.8332 - val_loss: 0.3563 - val_accuracy: 0.8479 - lr: 1.0000e-05
Epoch 53/60
199/199 [==============================] - 111s 559ms/step - loss: 0.3537 -
accuracy: 0.8269 - val_loss: 0.3504 - val_accuracy: 0.8479 - lr: 1.0000e-05
Epoch 54/60
199/199 [==============================] - 112s 562ms/step - loss: 0.3514 -
accuracy: 0.8291 - val_loss: 0.3519 - val_accuracy: 0.8394 - lr: 1.0000e-05
Epoch 55/60
199/199 [==============================] - 118s 590ms/step - loss: 0.3540 -
accuracy: 0.8275 - val_loss: 0.3544 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 56/60
199/199 [==============================] - 113s 566ms/step - loss: 0.3508 -
accuracy: 0.8288 - val_loss: 0.3583 - val_accuracy: 0.8423 - lr: 1.0000e-05
Epoch 57/60
199/199 [==============================] - 111s 559ms/step - loss: 0.3451 -
accuracy: 0.8322 - val_loss: 0.3593 - val_accuracy: 0.8479 - lr: 1.0000e-05
Epoch 58/60
199/199 [==============================] - 112s 562ms/step - loss: 0.3522 -
accuracy: 0.8218 - val_loss: 0.3578 - val_accuracy: 0.8451 - lr: 1.0000e-05
Epoch 59/60
199/199 [==============================] - 112s 563ms/step - loss: 0.3484 -
accuracy: 0.8347 - val_loss: 0.3604 - val_accuracy: 0.8451 - lr: 1.0000e-05
Epoch 60/60
199/199 [==============================] - 111s 560ms/step - loss: 0.3521 -
accuracy: 0.8303 - val_loss: 0.3562 - val_accuracy: 0.8451 - lr: 1.0000e-05
```

[ ]:
```
from keras.utils.vis_utils import plot_model
```

```
plot_model(model, to_file='model_plot.png', show_shapes=True,␣
↪show_layer_names=True)
```

[ ]:

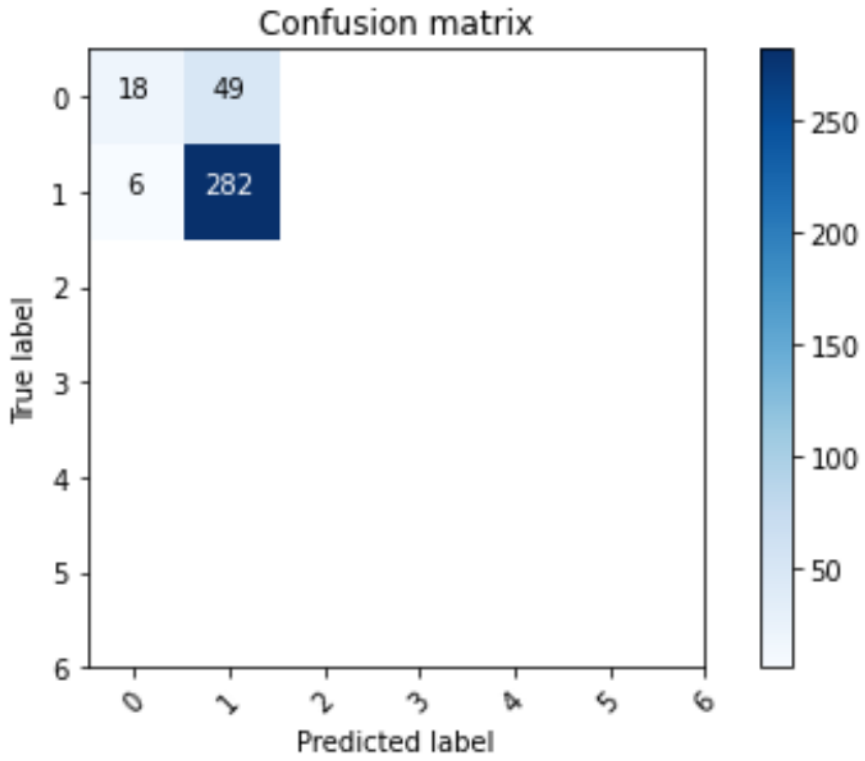

conv2d_input | InputLayer | input: [(None, 100, 125, 3)] | output: [(None, 100, 125, 3)]
conv2d | Conv2D | input: (None, 100, 125, 3) | output: (None, 100, 125, 32)
conv2d_1 | Conv2D | input: (None, 100, 125, 32) | output: (None, 100, 125, 32)
max_pooling2d | MaxPooling2D | input: (None, 100, 125, 32) | output: (None, 50, 62, 32)
dropout | Dropout | input: (None, 50, 62, 32) | output: (None, 50, 62, 32)
conv2d_2 | Conv2D | input: (None, 50, 62, 32) | output: (None, 50, 62, 32)
conv2d_3 | Conv2D | input: (None, 50, 62, 32) | output: (None, 50, 62, 32)
max_pooling2d_1 | MaxPooling2D | input: (None, 50, 62, 32) | output: (None, 25, 31, 32)
dropout_1 | Dropout | input: (None, 25, 31, 32) | output: (None, 25, 31, 32)
conv2d_4 | Conv2D | input: (None, 25, 31, 32) | output: (None, 25, 31, 64)
conv2d_5 | Conv2D | input: (None, 25, 31, 64) | output: (None, 25, 31, 64)
max_pooling2d_2 | MaxPooling2D | input: (None, 25, 31, 64) | output: (None, 12, 15, 64)
dropout_2 | Dropout | input: (None, 12, 15, 64) | output: (None, 12, 15, 64)
flatten | Flatten | input: (None, 12, 15, 64) | output: (None, 11520)
dense_5 | Dense | input: (None, 11520) | output: (None, 256)
dense_6 | Dense | input: (None, 256) | output: (None, 128)
dropout_3 | Dropout | input: (None, 128) | output: (None, 128)
dense_7 | Dense | input: (None, 128) | output: (None, 7)

[ ]:
```
loss, accuracy = model.evaluate(x_test, y_test, verbose=1)
loss_v, accuracy_v = model.evaluate(x_validate, y_validate, verbose=1)
print("Validation: accuracy = %f  ;  loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f  ;  loss = %f" % (accuracy, loss))
model.save("model.h5")
```

```
42/42 [==============================] - 10s 231ms/step - loss: 0.3513 -
accuracy: 0.8456
12/12 [==============================] - 3s 222ms/step - loss: 0.3562 -
accuracy: 0.8451
Validation: accuracy = 0.845070  ;  loss_v = 0.356162
Test: accuracy = 0.845627  ;  loss = 0.351293
```

[ ]:
```
import itertools
# Function to plot confusion matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Predict the values from the validation dataset
Y_pred = model.predict(x_validate)
```

```
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(y_validate,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)



# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(7))
```



[ ]:
```
# Predict the values from the validation dataset
Y_pred = model.predict(x_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(y_test,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
```

```
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(7))
```