Differentiating a weed from a crop seedling

```
# Importing the necessary packages

import pandas as pd
import numpy as np
import matplotlib

import tensorflow as tf
from zipfile import ZipFile
import os

from keras.models import Sequential
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers import Activation, Flatten, Dense, Dropout, BatchNormalization, LeakyReLU
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from keras.backend import clear_session
from sklearn.metrics import accuracy_score

import random
import sys
import cv2
from keras.utils import to_categorical
```

```
# Mounting the google drive

from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True
#  Converting labels to classes and assigning numbers

def classes_to_int(label):
    # label = classes.index(directory)
    label = label.strip()
    if label == "Black-grass":  return 0
    if label == "Charlock":  return 1
    if label == "Cleavers":  return 2
    if label == "Common Chickweed":  return 3
    if label == "Common wheat":  return 4
    if label == "Fat Hen":  return 5
    if label == "Loose Silky-bent": return 6
    if label == "Maize":  return 7
    if label == "Scentless Mayweed": return 8
    if label == "Shepherds Purse": return 9
    if label == "Small-flowered Cranesbill": return 10
    if label == "Sugar beet": return 11
    print("Invalid Label", label)
    return 12


#  Converting back to labels from numbers

def int_to_classes(i):
    if i == 0: return "Black-grass"
    elif i == 1: return "Charlock"
    elif i == 2: return "Cleavers"
    elif i == 3: return "Common Chickweed"
    elif i == 4: return "Common wheat"
    elif i == 5: return "Fat Hen"
    elif i == 6: return "Loose Silky-bent"
    elif i == 7: return "Maize"
    elif i == 8: return "Scentless Mayweed"
    elif i == 9: return "Shepherds Purse"
    elif i == 10: return "Small-flowered Cranesbill"
    elif i == 11: return "Sugar beet"
    print("Invalid class ", i)
    return "Invalid Class"
```

1. Read the images and generate the train and test dataset

```
# Extracted the data of Zip file through the commands:
#with ZipFile('test.zip', 'r') as z:
#  z.extractall()
```

```
# Opening train folder
os.chdir('/content/train')
```

```
# Listing the contents of the train folder
os.listdir()
```

```
['Maize',
 'Shepherds Purse',
 'Fat Hen',
 'Common wheat',
 'Loose Silky-bent',
 'Cleavers',
 'Charlock',
 'Sugar beet',
 'Scentless Mayweed',
 'Black-grass',
 'Common Chickweed',
 'Small-flowered Cranesbill']
```

TRAIN DATA

```
# Loading all the images, pre-processing them, and storing them in a list of train data

def readTrainData(trainDir):
    data = []
    labels = []
    directories = os.listdir()
    for directory in directories:
        absDirPath = os.path.join(os.path.sep, trainDir, directory)
        images = os.listdir(absDirPath)

        for imageFileName in images:
            imageFullPath = os.path.join(trainDir, directory, imageFileName)
            img = load_img(imageFullPath)
            arr = img_to_array(img)  #Converting image to array
            arr = cv2.resize(arr, (128, 128)) #Resizing the array
            data.append(arr)
            label = classes_to_int(directory)
            labels.append(label)
    return data, labels
```

```
path = os.getcwd()
X, Y = readTrainData(path)
```

```
# Scaling the data
X = np.array(X, dtype="float") / 255.0
Y = np.array(Y)
```

```
# Converting the target column to 12 categorical classes
Y =  to_categorical(Y, num_classes=12)
```

TEST DATA

```
# Loading all the images, pre-processing them, and storing them in a list of test data

def readTestData(testDir):
    data2 = []
    filenames = []
    images = os.listdir(testDir)

    for imageFileName in images:
        imageFullPath = os.path.join(testDir, imageFileName)
        img = load_img(imageFullPath)
        arr = img_to_array(img)
        arr = cv2.resize(arr, (128, 128))
        data2.append(arr)
        filenames.append(imageFileName)
    return data2, filenames

path2 = '/content/gdrive/My Drive/Colab Notebooks/plant-seedlings-classification/test/'
X_test, filenames = readTestData(path2)

# Scaling the data
X_test = np.array(X_test, dtype="float") / 255.0
```

2. Divide the data set into Train and validation data sets

```
# Dividing the data set into train and validation datasets

(X_train, X_val, Y_train, Y_val) = train_test_split(X, Y, test_size = 0.3, random_state = 47)
```

3. Initialize & build the model

```
# Clear out tensorflow memory
clear_session()

# Define Model
model = Sequential()
model.add(BatchNormalization(input_shape = (128,128,3)))

# 1st Conv Layer
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(128, 128, 3), padding="same"))
#kernel_initializer = 'he_normal'

# Max Pooling layer
# Max Pooling layer
model.add(MaxPooling2D(pool_size=2))

# Dropout
model.add(Dropout(rate = 0.2))

# 2nd Conv Layer
model.add(Conv2D(filters=64, kernel_size=5, kernel_initializer = 'he_normal', padding="same"))
model.add(Activation("relu"))

# Max Pooling layer
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Dropout
model.add(Dropout(rate = 0.2))

# Flattening the data
model.add(Flatten())

# 1st dense layer
model.add(Dense(128, kernel_initializer = 'he_normal'))
model.add(Activation("relu"))

# Dropout
model.add(Dropout(rate = 0.3))

# 2nd dense layer
model.add(Dense(64, kernel_initializer = 'he_normal'))
model.add(Activation("relu"))

# Output layer
model.add(Dense(output_dim=12, activation = 'softmax'))

model.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:107: The name tf.reset_defaul

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:111: The name tf.placeholder_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_g

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder 

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto 

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is d

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_varia

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_in

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:2041: The name tf.nn.fused_ba

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_unif

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from t
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4479: The name tf.truncated_n

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
batch_normalization_1 (Batch (None, 128, 128, 3)       12        
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 32)      896       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 32)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 64, 64, 32)        0         
```

```
dropout_1 (Dropout)          (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        51264
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 65536)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8388736
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 12)                780
=================================================================
Total params: 8,449,944
Trainable params: 8,449,938
Non-trainable params: 6
_________________________________________________________________
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:42: UserWarning: Update your `Dense` call to the Keras 2 API: `Dens
```

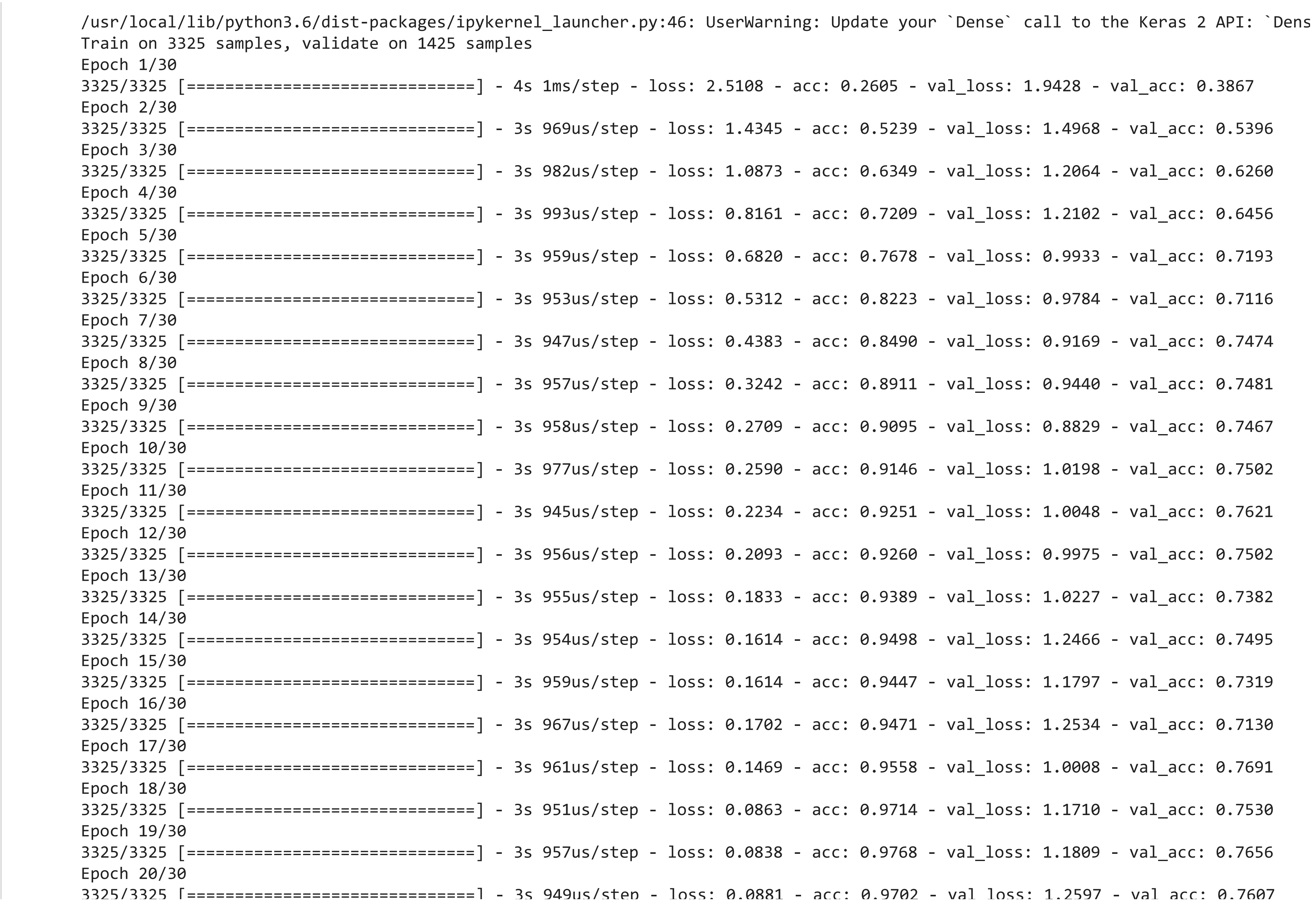
```
# Loss and Optimizer
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Training the model
model.fit(X_train, Y_train, batch_size=60, epochs=10, validation_data=(X_val, Y_val))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecate

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is depr

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorf
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is d

Train on 3325 samples, validate on 1425 samples
Epoch 1/10
3325/3325 [==============================] - 6s 2ms/step - loss: 2.9496 - acc: 0.1991 - val_loss: 1.9612 - val_acc: 0.4456
Epoch 2/10
3325/3325 [==============================] - 3s 843us/step - loss: 1.7604 - acc: 0.4553 - val_loss: 1.5154 - val_acc: 0.5404
Epoch 3/10
3325/3325 [==============================] - 3s 868us/step - loss: 1.3089 - acc: 0.5765 - val_loss: 1.0574 - val_acc: 0.6688
Epoch 4/10
3325/3325 [==============================] - 3s 862us/step - loss: 0.9829 - acc: 0.6692 - val_loss: 0.9722 - val_acc: 0.6625
Epoch 5/10
3325/3325 [==============================] - 3s 867us/step - loss: 0.8258 - acc: 0.7224 - val_loss: 0.8658 - val_acc: 0.6947
Epoch 6/10
3325/3325 [==============================] - 3s 862us/step - loss: 0.6694 - acc: 0.7711 - val_loss: 0.7793 - val_acc: 0.7453
Epoch 7/10
3325/3325 [==============================] - 3s 875us/step - loss: 0.5211 - acc: 0.8202 - val_loss: 0.7683 - val_acc: 0.7319
Epoch 8/10
3325/3325 [==============================] - 3s 860us/step - loss: 0.4632 - acc: 0.8376 - val_loss: 0.8594 - val_acc: 0.7102
Epoch 9/10
3325/3325 [==============================] - 3s 857us/step - loss: 0.3938 - acc: 0.8638 - val_loss: 0.8914 - val_acc: 0.7263
Epoch 10/10
3325/3325 [==============================] - 3s 857us/step - loss: 0.3060 - acc: 0.8941 - val_loss: 0.8744 - val_acc: 0.7389
<keras.callbacks.History at 0x7f66ac803710>
```

4. Optimize the model

```
# Clear out tensorflow memory
```

```
clear_session()

# Define Model
model = Sequential()
model.add(BatchNormalization(input_shape = (128,128,3)))

# 1st Conv Layer
model.add(Conv2D(32, (3,3), input_shape=(128, 128, 3)))
model.add(LeakyReLU(alpha=0.1))

# Max Pooling layer
model.add(MaxPooling2D(pool_size=2))

# Dropout
model.add(Dropout(rate = 0.2))

# 2nd Conv Layer
model.add(Conv2D(filters=64, kernel_size=5, padding="same"))
model.add(LeakyReLU(alpha=0.1))

# Max Pooling layer
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Dropout
model.add(Dropout(rate = 0.2))

# Flattening the data
model.add(Flatten())

# 1st dense layer
model.add(Dense(128, kernel_initializer = 'he_normal'))
model.add(LeakyReLU(alpha=0.1))

# Dropout
model.add(Dropout(rate = 0.3))

# 2nd dense layer
model.add(Dense(64, kernel_initializer = 'he_normal'))
model.add(Dense(64, kernel_initializer = 'he_normal'))
model.add(LeakyReLU(alpha=0.1))

# 3rd dense layer
model.add(Dense(32, kernel_initializer = 'he_normal'))
model.add(LeakyReLU(alpha=0.1))

# Output layer
model.add(Dense(output_dim=12, activation = 'softmax'))

# Loss and Optimizer
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Training the model
model.fit(X_train, Y_train, batch_size=60, epochs=30, validation_data=(X_val, Y_val))
```



```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:46: UserWarning: Update your `Dense` call to the Keras 2 API: `Dens
Train on 3325 samples, validate on 1425 samples
Epoch 1/30
3325/3325 [==============================] - 4s 1ms/step - loss: 2.5108 - acc: 0.2605 - val_loss: 1.9428 - val_acc: 0.3867
Epoch 2/30
3325/3325 [==============================] - 3s 969us/step - loss: 1.4345 - acc: 0.5239 - val_loss: 1.4968 - val_acc: 0.5396
Epoch 3/30
3325/3325 [==============================] - 3s 982us/step - loss: 1.0873 - acc: 0.6349 - val_loss: 1.2064 - val_acc: 0.6260
Epoch 4/30
3325/3325 [==============================] - 3s 993us/step - loss: 0.8161 - acc: 0.7209 - val_loss: 1.2102 - val_acc: 0.6456
Epoch 5/30
3325/3325 [==============================] - 3s 959us/step - loss: 0.6820 - acc: 0.7678 - val_loss: 0.9933 - val_acc: 0.7193
Epoch 6/30
3325/3325 [==============================] - 3s 953us/step - loss: 0.5312 - acc: 0.8223 - val_loss: 0.9784 - val_acc: 0.7116
Epoch 7/30
3325/3325 [==============================] - 3s 947us/step - loss: 0.4383 - acc: 0.8490 - val_loss: 0.9169 - val_acc: 0.7474
Epoch 8/30
3325/3325 [==============================] - 3s 957us/step - loss: 0.3242 - acc: 0.8911 - val_loss: 0.9440 - val_acc: 0.7481
Epoch 9/30
3325/3325 [==============================] - 3s 958us/step - loss: 0.2709 - acc: 0.9095 - val_loss: 0.8829 - val_acc: 0.7467
Epoch 10/30
3325/3325 [==============================] - 3s 977us/step - loss: 0.2590 - acc: 0.9146 - val_loss: 1.0198 - val_acc: 0.7502
Epoch 11/30
3325/3325 [==============================] - 3s 945us/step - loss: 0.2234 - acc: 0.9251 - val_loss: 1.0048 - val_acc: 0.7621
Epoch 12/30
3325/3325 [==============================] - 3s 956us/step - loss: 0.2093 - acc: 0.9260 - val_loss: 0.9975 - val_acc: 0.7502
Epoch 13/30
3325/3325 [==============================] - 3s 955us/step - loss: 0.1833 - acc: 0.9389 - val_loss: 1.0227 - val_acc: 0.7382
Epoch 14/30
3325/3325 [==============================] - 3s 954us/step - loss: 0.1614 - acc: 0.9498 - val_loss: 1.2466 - val_acc: 0.7495
Epoch 15/30
3325/3325 [==============================] - 3s 959us/step - loss: 0.1614 - acc: 0.9447 - val_loss: 1.1797 - val_acc: 0.7319
Epoch 16/30
3325/3325 [==============================] - 3s 967us/step - loss: 0.1702 - acc: 0.9471 - val_loss: 1.2534 - val_acc: 0.7130
Epoch 17/30
3325/3325 [==============================] - 3s 961us/step - loss: 0.1469 - acc: 0.9558 - val_loss: 1.0008 - val_acc: 0.7691
Epoch 18/30
3325/3325 [==============================] - 3s 951us/step - loss: 0.0863 - acc: 0.9714 - val_loss: 1.1710 - val_acc: 0.7530
Epoch 19/30
3325/3325 [==============================] - 3s 957us/step - loss: 0.0838 - acc: 0.9768 - val_loss: 1.1809 - val_acc: 0.7656
Epoch 20/30
3325/3325 [==============================] - 3s 949us/step - loss: 0.0881 - acc: 0.9702 - val_loss: 1.2597 - val_acc: 0.7607
```

```
Epoch 21/30
3325/3325 [==============================] - 3s 960us/step - loss: 0.0795 - acc: 0.9753 - val_loss: 1.0471 - val_acc: 0.7705
Epoch 22/30
3325/3325 [==============================] - 3s 962us/step - loss: 0.0640 - acc: 0.9808 - val_loss: 1.1731 - val_acc: 0.7649
Epoch 23/30
3325/3325 [==============================] - 3s 945us/step - loss: 0.0626 - acc: 0.9808 - val_loss: 1.3530 - val_acc: 0.7796
Epoch 24/30
3325/3325 [==============================] - 3s 957us/step - loss: 0.0634 - acc: 0.9832 - val_loss: 1.1650 - val_acc: 0.7923
Epoch 25/30
3325/3325 [==============================] - 3s 948us/step - loss: 0.0576 - acc: 0.9805 - val_loss: 1.3638 - val_acc: 0.7523
Epoch 26/30
3325/3325 [==============================] - 3s 963us/step - loss: 0.0825 - acc: 0.9720 - val_loss: 1.3562 - val_acc: 0.7495
Epoch 27/30
3325/3325 [==============================] - 3s 969us/step - loss: 0.0731 - acc: 0.9741 - val_loss: 1.4182 - val_acc: 0.7565
Epoch 28/30
3325/3325 [==============================] - 3s 953us/step - loss: 0.0606 - acc: 0.9808 - val_loss: 1.4430 - val_acc: 0.7425
Epoch 29/30
3325/3325 [==============================] - 3s 951us/step - loss: 0.0655 - acc: 0.9762 - val_loss: 1.4016 - val_acc: 0.7551
Epoch 30/30
3325/3325 [==============================] - 3s 945us/step - loss: 0.0743 - acc: 0.9768 - val_loss: 1.5287 - val_acc: 0.7326
<keras.callbacks.History at 0x7f6525e3cf60>
```

5. Predict the accuracy for both train and validation data

```
Y_predict1 = model.predict(X_val)
Y_predict2 = model.predict(X_train)
```

```
# Finding the accuracy:

accuracy1 = accuracy_score(Y_val.argmax(axis=1), Y_predict1.argmax(axis=1))
print("The accuracy of validation data is", round(accuracy1*100, 2))

accuracy2 = accuracy_score(Y_train.argmax(axis=1), Y_predict2.argmax(axis=1))
print("The accuracy of train data is", round(accuracy2*100, 2))
```

```
The accuracy of validation data is 73.26
```