

```
In [1]: !pip install efficientnet
!pip install iterative-stratification
!pip install gdown

import os
if not os.path.isfile('model_effnet_bo_087.h5'):
    !gdown https://drive.google.com/uc?id=1FXF1HymYbRf30lThMTXAa74TRup3AhD_
if not os.path.isfile('stage_1_train.csv.zip'):
    !gdown https://drive.google.com/uc?id=10t89rZpBlwzLG-SL7owQaqBU2tpvAfmM
    !unzip stage_1_train.csv.zip
```

Collecting efficientnet

Downloading <https://files.pythonhosted.org/packages/97/82/f3ae07316f0461417dc54affab6e86ab188a5a22f33176d35271628b96e0/efficientnet-1.0.0-py3-none-any.whl> (<https://files.pythonhosted.org/packages/97/82/f3ae07316f0461417dc54affab6e86ab188a5a22f33176d35271628b96e0/efficientnet-1.0.0-py3-none-any.whl>)

Requirement already satisfied: scikit-image in /opt/conda/lib/python3.6/site-packages (from efficientnet) (0.16.2)

Requirement already satisfied: keras-applications<=1.0.8,>=1.0.7 in /opt/conda/lib/python3.6/site-packages (from efficientnet) (1.0.8)

Requirement already satisfied: pillow>=4.3.0 in /opt/conda/lib/python3.6/site-packages (from scikit-image->efficientnet) (5.4.1)

Requirement already satisfied: PyWavelets>=0.4.0 in /opt/conda/lib/python3.6/site-packages (from scikit-image->efficientnet) (1.1.1)

Requirement already satisfied: networkx>=2.0 in /opt/conda/lib/python3.6/site-packages (from scikit-image->efficientnet) (2.4)

Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /opt/conda/lib/python3.6/site-packages (from scikit-image->efficientnet) (3.0.3)

Requirement already satisfied: imageio>=2.3.0 in /opt/conda/lib/python3.6/site-packages (from scikit-image->efficientnet) (2.6.1)

Requirement already satisfied: scipy>=0.19.0 in /opt/conda/lib/python3.6/site-packages (from scikit-image->efficientnet) (1.2.1)

Requirement already satisfied: numpy>=1.9.1 in /opt/conda/lib/python3.6/site-packages (from keras-applications<=1.0.8,>=1.0.7->efficientnet) (1.16.4)

Requirement already satisfied: h5py in /opt/conda/lib/python3.6/site-packages (from keras-applications<=1.0.8,>=1.0.7->efficientnet) (2.9.0)

Requirement already satisfied: decorator>=4.3.0 in /opt/conda/lib/python3.6/site-packages (from networkx>=2.0->scikit-image->efficientnet) (4.4.0)

Requirement already satisfied: cyclor>=0.10 in /opt/conda/lib/python3.6/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet) (0.10.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.6/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet) (1.1.0)

Requirement already satisfied: pyparsing!=2.0.4,!2.1.2,!2.1.6,>=2.0.1 in /opt/conda/lib/python3.6/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet) (2.4.2)

Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/lib/python3.6/site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet) (2.8.0)

Requirement already satisfied: six in /opt/conda/lib/python3.6/site-packages (from h5py->keras-applications<=1.0.8,>=1.0.7->efficientnet) (1.12.0)

Requirement already satisfied: setuptools in /opt/conda/lib/python3.6/site-packages (from kiwisolver>=1.0.1->matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet) (41.4.0)

Installing collected packages: efficientnet

Successfully installed efficientnet-1.0.0

Collecting iterative-stratification

Downloading [https://files.pythonhosted.org/packages/9d/79/9ba64c8c07b07b8b45d80725b2ebd7b7884701c1da34f70d4749f7b45f9a/iterative\\_stratification-0.1.6-py3-none-any.whl](https://files.pythonhosted.org/packages/9d/79/9ba64c8c07b07b8b45d80725b2ebd7b7884701c1da34f70d4749f7b45f9a/iterative_stratification-0.1.6-py3-none-any.whl) ([https://files.pythonhosted.org/packages/9d/79/9ba64c8c07b07b8b45d80725b2ebd7b7884701c1da34f70d4749f7b45f9a/iterative\\_stratification-0.1.6-py3-none-any.whl](https://files.pythonhosted.org/packages/9d/79/9ba64c8c07b07b8b45d80725b2ebd7b7884701c1da34f70d4749f7b45f9a/iterative_stratification-0.1.6-py3-none-any.whl))

Requirement already satisfied: numpy in /opt/conda/lib/python3.6/site-packages (from iterative-stratification) (1.16.4)

Requirement already satisfied: scipy in /opt/conda/lib/python3.6/site-packages (from iterative-stratification) (1.2.1)

Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.6/site-packages (from iterative-stratification) (0.21.3)

Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.6/site-packages (from scikit-learn->iterative-stratification) (0.13.2)

Installing collected packages: iterative-stratification

Successfully installed iterative-stratification-0.1.6

Collecting gdown

Downloading <https://files.pythonhosted.org/packages/b0/b4/a8e9d0b02bca6aa53087001abf064cc9992bda11bd6840875b8098d93573/gdown-3.8.3.tar.gz> (<https://files.pythonhosted.org/packages/b0/b4/a8e9d0b02bca6aa53087001abf064cc9992bda11bd6840875b8098d93573/gdown-3.8.3.tar.gz>)

Requirement already satisfied: filelock in /opt/conda/lib/python3.6/site-packages (from gdown) (3.0.12)

Requirement already satisfied: requests in /opt/conda/lib/python3.6/site-packages (from gdown) (2.22.0)

Requirement already satisfied: six in /opt/conda/lib/python3.6/site-packages (from gdown) (1.12.0)

Requirement already satisfied: tqdm in /opt/conda/lib/python3.6/site-packages (from gdown) (4.36.1)

Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in /opt/conda/lib/python3.6/site-packages (from requests->gdown) (1.24.2)

Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.6/site-packages (from requests->gdown) (2019.9.11)

Requirement already satisfied: idna<2.9,>=2.5 in /opt/conda/lib/python3.6/site-packages (from requests->gdown) (2.8)

Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/lib/python3.6/site-packages (from requests->gdown) (3.0.4)

Building wheels for collected packages: gdown

Building wheel for gdown (setup.py) ... done

Created wheel for gdown: filename=gdown-3.8.3-cp36-none-any.whl size=8850 sha256=8074522793dc355a859935949558d718048dcd1dd53cb683f86216a7254e288a6

Stored in directory: /tmp/.cache/pip/wheels/a7/9d/16/9e0bda9a327ff2cddae8de48a27553fb1efce73133593d066

Successfully built gdown

```
Installing collected packages: gdown
Successfully installed gdown-3.8.3
Downloading...
From: https://drive.google.com/uc?id=1FXF1HymYbRf30lThMTXAa74TRup3AhD\_ (https://drive.google.com/uc?id=1FXF1HymYbRf30lThMTXAa74TRup3AhD\_)
To: /kaggle/working/model_effnet_bo_087.h5
16.7MB [00:00, 71.1MB/s]
Downloading...
From: https://drive.google.com/uc?id=10t89rZpBlwzLG-SL7owQaqBU2tpvAfmM (https://drive.google.com/uc?id=10t89rZpBlwzLG-SL7owQaqBU2tpvAfmM)
To: /kaggle/working/stage_1_train.csv.zip
15.2MB [00:00, 98.8MB/s]
Archive: stage_1_train.csv.zip
  inflating: stage_1_train.csv
```

In [2]: `!ls`

```
__notebook__.ipynb  model_effnet_bo_087.h5  stage_1_train.csv.zip
__output__.json    stage_1_train.csv
```

In [3]:

```
import numpy as np
import pandas as pd
import pydicom
import os
import glob
import random
import cv2
import tensorflow as tf
from math import ceil, floor
from tqdm import tqdm
from imgaug import augmenters as iaa
import matplotlib.pyplot as plt
from math import ceil, floor
import keras
import keras.backend as K
from keras.callbacks import Callback, ModelCheckpoint
from keras.layers import Dense, Flatten, Dropout
from keras.models import Model, load_model
from keras.utils import Sequence
from keras.losses import binary_crossentropy
from keras.optimizers import Adam
```

Using TensorFlow backend.

In [4]:

```
HEIGHT = 256
WIDTH = 256
CHANNELS = 3
SHAPE = (HEIGHT, WIDTH, CHANNELS)
input_folder = '../input/rsna-intracranial-hemorrhage-detection/rsna-intracranial-hemorrhage-detection/'
path_train_img = input_folder + 'stage_2_train/'
path_test_img = input_folder + 'stage_2_test/'
```

In [5]:

```
train_df = pd.read_csv('stage_1_train.csv')
train_df.head()
```

Out[5]:

	ID	Label
0	ID_63eb1e259_epidural	0
1	ID_63eb1e259_intraparenchymal	0
2	ID_63eb1e259_intraventricular	0
3	ID_63eb1e259_subarachnoid	0
4	ID_63eb1e259_subdural	0

In [6]:

```
# extract subtype
train_df['sub_type'] = train_df['ID'].apply(lambda x: x.split('_')[-1])
# extract filename
train_df['file_name'] = train_df['ID'].apply(lambda x: '_'.join(x.split('_')[:2]) + '.dcm')
train_df.head()
```

Out[6]:

	ID	Label	sub_type	file_name
0	ID_63eb1e259_epidural	0	epidural	ID_63eb1e259.dcm
1	ID_63eb1e259_intraparenchymal	0	intraparenchymal	ID_63eb1e259.dcm
2	ID_63eb1e259_intraventricular	0	intraventricular	ID_63eb1e259.dcm
3	ID_63eb1e259_subarachnoid	0	subarachnoid	ID_63eb1e259.dcm
4	ID_63eb1e259_subdural	0	subdural	ID_63eb1e259.dcm

In [7]: `train_df.shape`

Out[7]: (4045572, 4)

```
In [8]: # remove duplicates
train_df.drop_duplicates(['Label', 'sub_type', 'file_name'], inplace=True)
train_df.shape
```

Out[8]: (4045548, 4)

```
In [9]: print("Number of train images availabe:", len(os.listdir(path_train_img)))

Number of train images availabe: 752803
```

```
In [10]: train_final_df = pd.pivot_table(train_df.drop(columns='ID'), index="file_name", \
                                         columns="sub_type", values="Label")
train_final_df.head()
```

Out[10]:

	sub_type	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
file_name							
ID_000039fa0.dcm	0	0	0	0	0	0	0
ID_00005679d.dcm	0	0	0	0	0	0	0
ID_00008ce3c.dcm	0	0	0	0	0	0	0
ID_0000950d7.dcm	0	0	0	0	0	0	0
ID_0000aee4b.dcm	0	0	0	0	0	0	0

```
In [11]: train_final_df.shape
```

Out[11]: (674258, 6)

```
In [12]: # Invalid image ID_6431af929.dcm
train_final_df.drop('ID_6431af929.dcm', inplace=True)
```

```
In [13]: import efficientnet.keras as efn
from iterstrat.ml_stratifiers import MultilabelStratifiedShuffleSplit
```

```
In [14]: def get_corrected_bsb_window(dcm, window_center, window_width):
#----- Correct Dicom Image -----#
if (dcm.BitsStored == 12) and (dcm.PixelRepresentation == 0) and (int(dcm.RescaleIntercept) > -100):
    x = dcm.pixel_array + 1000
    px_mode = 4096
    x[x>=px_mode] = x[x>=px_mode] - px_mode
    dcm.PixelData = x.tobytes()
    dcm.RescaleIntercept = -1000

#----- Windowing -----#
img = dcm.pixel_array * dcm.RescaleSlope + dcm.RescaleIntercept
img_min = window_center - window_width // 2
img_max = window_center + window_width // 2
img = np.clip(img, img_min, img_max)
return img

def get_rgb_image(img):
    brain_img = get_corrected_bsb_window(img, 40, 80)
    subdural_img = get_corrected_bsb_window(img, 80, 200)
    soft_img = get_corrected_bsb_window(img, 40, 380)

    brain_img = (brain_img - 0) / 80
    subdural_img = (subdural_img - (-20)) / 200
    soft_img = (soft_img - (-150)) / 380
    bsb_img = np.array([brain_img, subdural_img, soft_img]).transpose(1,2,0)

    return bsb_img

def _read(path, desired_size=(WIDTH, HEIGHT)):

    dcm = pydicom.dcmread(path)

    try:
        img = get_rgb_image(dcm)
    except:
        img = np.zeros(desired_size)

    img = cv2.resize(img, desired_size[:2], interpolation=cv2.INTER_LINEAR)

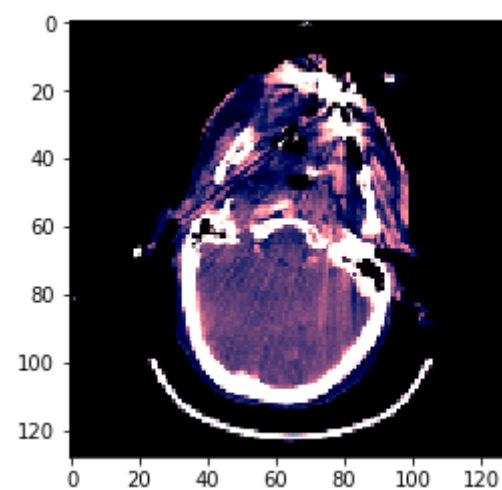
    return img
```

```
In [15]: _read(path_train_img + 'ID_ffff922b9.dcm', (128, 128)).shape
```

Out[15]: (128, 128, 3)

```
In [16]: plt.imshow(
          _read(path_train_img + 'ID_ffff922b9.dcm', (128, 128))
        )
```

```
Out[16]: <matplotlib.image.AxesImage at 0x7f7b19c2dba8>
```



```
In [17]: # Augmentations
# Flip Left Right
# Cropping
sometimes = lambda aug: iaa.Sometimes(0.25, aug)
augmentation = iaa.Sequential([
    iaa.Fliplr(0.25),
    sometimes(iaa.Crop(px=(0, 25), keep_size = True,
                        sample_independently = False))
], random_order = True)
```

```

In [18]: # Train Data Generator
class TrainDataGenerator(keras.utils.Sequence):

    def __init__(self, dataset, labels, batch_size=16, img_size=(512, 512), img_dir = path_train_img, \
        augment = False, *args, **kwargs):
        self.dataset = dataset
        self.ids = dataset.index
        self.labels = labels
        self.batch_size = batch_size
        self.img_size = img_size
        self.img_dir = img_dir
        self.augment = augment
        self.on_epoch_end()

    def __len__(self):
        return int(ceil(len(self.ids) / self.batch_size))

    def __getitem__(self, index):
        indices = self.indices[index*self.batch_size:(index+1)*self.batch_size]
        X, Y = self.__data_generation(indices)
        return X, Y

    def augmentor(self, image):
        augment_img = augmentation
        image_aug = augment_img.augment_image(image)
        return image_aug

    def on_epoch_end(self):
        self.indices = np.arange(len(self.ids))
        np.random.shuffle(self.indices)

    def __data_generation(self, indices):
        X = np.empty((self.batch_size, *self.img_size, 3))
        Y = np.empty((self.batch_size, 6), dtype=np.float32)

        for i, index in enumerate(indices):
            ID = self.ids[index]
            image = _read(self.img_dir + ID, self.img_size)
            if self.augment:
                X[i,] = self.augmentor(image)
            else:
                X[i,] = image
            Y[i,] = self.labels.iloc[index].values
        return X, Y

class TestDataGenerator(keras.utils.Sequence):

    def __init__(self, ids, labels, batch_size = 5, img_size = (512, 512), img_dir = path_test_img, \
        *args, **kwargs):
        self.ids = ids
        self.labels = labels
        self.batch_size = batch_size
        self.img_size = img_size
        self.img_dir = img_dir
        self.on_epoch_end()

    def __len__(self):
        return int(ceil(len(self.ids) / self.batch_size))

    def __getitem__(self, index):
        indices = self.indices[index*self.batch_size:(index+1)*self.batch_size]
        list_IDS_temp = [self.ids[k] for k in indices]
        X = self.__data_generation(list_IDS_temp)
        return X

    def on_epoch_end(self):
        self.indices = np.arange(len(self.ids))

    def __data_generation(self, list_IDS_temp):
        X = np.empty((self.batch_size, *self.img_size, 3))
        for i, ID in enumerate(list_IDS_temp):
            image = _read(self.img_dir + ID, self.img_size)
            X[i,] = image
        return X

```

```
In [19]: # load test set
test_df = pd.read_csv(input_folder + 'stage_2_sample_submission.csv')
test_df.head()
```

```
Out[19]:
```

	ID	Label
0	ID_0fbf6a978_epidural	0.5
1	ID_0fbf6a978_intraparenchymal	0.5
2	ID_0fbf6a978_intraventricular	0.5
3	ID_0fbf6a978_subarachnoid	0.5
4	ID_0fbf6a978_subdural	0.5

```
In [20]: # extract subtype
test_df['sub_type'] = test_df['ID'].apply(lambda x: x.split('_')[-1])
# extract filename
test_df['file_name'] = test_df['ID'].apply(lambda x: '_'.join(x.split('_')[:2]) + '.dcm')

test_df = pd.pivot_table(test_df.drop(columns='ID'), index="file_name", \
                           columns="sub_type", values="Label")

test_df.head()

test_df.shape
```

```
Out[20]: (121232, 6)
```

```
In [21]: test_df.head()
```

```
Out[21]:
```

	sub_type	any	epidural	intraparenchymal	intraventricular	subarachnoid	subdural
file_name							
ID_000000e27.dcm	0.5	0.5	0.5	0.5	0.5	0.5	0.5
ID_000009146.dcm	0.5	0.5	0.5	0.5	0.5	0.5	0.5
ID_00007b8cb.dcm	0.5	0.5	0.5	0.5	0.5	0.5	0.5
ID_000134952.dcm	0.5	0.5	0.5	0.5	0.5	0.5	0.5
ID_000176f2a.dcm	0.5	0.5	0.5	0.5	0.5	0.5	0.5

```
In [22]: # https://github.com/trent-b/iterative-stratification
# Multilabel stratification
splits = MultilabelStratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=12345)
file_names = train_final_df.index
labels = train_final_df.values
# Lets take only the first split
split = next(splits.split(file_names, labels))
train_idx = split[0]
valid_idx = split[1]
submission_predictions = []
len(train_idx), len(valid_idx)
```

```
Out[22]: (539405, 134852)
```

```
In [23]: # train data generator
data_generator_train = TrainDataGenerator(train_final_df.iloc[train_idx],
                                           train_final_df.iloc[train_idx],
                                           128,
                                           (WIDTH, HEIGHT),
                                           augment=False)

# validation data generator
data_generator_val = TrainDataGenerator(train_final_df.iloc[valid_idx],
                                         train_final_df.iloc[valid_idx],
                                         128,
                                         (WIDTH, HEIGHT),
                                         augment=False)

data_generator_test = TestDataGenerator(test_df.index, None,
                                         128,
                                         (WIDTH, HEIGHT),
                                         augment=False)
```

```
In [24]: len(data_generator_train), len(data_generator_val), len(data_generator_test)
```

```
Out[24]: (4215, 1054, 948)
```



```

In [25]: from keras import backend as K

def weighted_log_loss(y_true, y_pred):
    """
    Can be used as the loss function in model.compile()
    -----
    """

    class_weights = np.array([2., 1., 1., 1., 1., 1.])

    eps = K.epsilon()

    y_pred = K.clip(y_pred, eps, 1.0-eps)

    out = -(
        y_true * K.log(y_pred) * class_weights
        + (1.0 - y_true) * K.log(1.0 - y_pred) * class_weights)

    return K.mean(out, axis=-1)

def _normalized_weighted_average(arr, weights=None):
    """
    A simple Keras implementation that mimics that of
    numpy.average(), specifically for this competition
    """

    if weights is not None:
        scl = K.sum(weights)
        weights = K.expand_dims(weights, axis=1)
        return K.sum(K.dot(arr, weights), axis=1) / scl
    return K.mean(arr, axis=1)

def weighted_loss(y_true, y_pred):
    """
    Will be used as the metric in model.compile()
    -----

    Similar to the custom loss function 'weighted_log_loss()' above
    but with normalized weights, which should be very similar
    to the official competition metric:
        https://www.kaggle.com/kambarakun/lb-probe-weights-n-of-positives-scoring
    and hence:
        sklearn.metrics.log_loss with sample weights
    """

    class_weights = K.variable([2., 1., 1., 1., 1., 1.])

    eps = K.epsilon()

    y_pred = K.clip(y_pred, eps, 1.0-eps)

    loss = -(
        y_true * K.log(y_pred)
        + (1.0 - y_true) * K.log(1.0 - y_pred))

    loss_samples = _normalized_weighted_average(loss, class_weights)

    return K.mean(loss_samples)

def weighted_log_loss_metric(trues, preds):
    """
    Will be used to calculate the log loss
    of the validation set in PredictionCheckpoint()
    -----
    """
    class_weights = [2., 1., 1., 1., 1., 1.]

    epsilon = 1e-7

    preds = np.clip(preds, epsilon, 1-epsilon)
    loss = trues * np.log(preds) + (1 - trues) * np.log(1 - preds)
    loss_samples = np.average(loss, axis=1, weights=class_weights)

    return - loss_samples.mean()

```

```
In [26]: base_model = efn.EfficientNetB0(weights = 'imagenet', include_top = False, \
                                             pooling = 'avg', input_shape = (HEIGHT, WIDTH, 3))

x = base_model.output
x = Dropout(0.125)(x)
output_layer = Dense(6, activation = 'sigmoid')(x)
model = Model(inputs=base_model.input, outputs=output_layer)
model.compile(optimizer = Adam(lr = 0.0001), loss = 'binary_crossentropy', metrics = ['acc'])
model.load_weights('model_effnet_bo_087.h5')
model.summary()
```

Downloading data from [https://github.com/Callidior/keras-applications/releases/download/efficientnet/efficientnet-b0\\_weights\\_tf\\_dim\\_ordering\\_tf\\_kernels\\_autoaugment\\_notop.h5](https://github.com/Callidior/keras-applications/releases/download/efficientnet/efficientnet-b0_weights_tf_dim_ordering_tf_kernels_autoaugment_notop.h5) ([https://github.com/Callidior/keras-applications/releases/download/efficientnet/efficientnet-b0\\_weights\\_tf\\_dim\\_ordering\\_tf\\_kernels\\_autoaugment\\_notop.h5](https://github.com/Callidior/keras-applications/releases/download/efficientnet/efficientnet-b0_weights_tf_dim_ordering_tf_kernels_autoaugment_notop.h5))

```
16809984/16804768 [=====] - 0s 0us/step
```

Model: "model 1"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 256, 256, 3)	0	
stem_conv (Conv2D)	(None, 128, 128, 32)	864	input_1[0][0]
stem_bn (BatchNormalization)	(None, 128, 128, 32)	128	stem_conv[0][0]
stem_activation (Activation)	(None, 128, 128, 32)	0	stem_bn[0][0]
block1a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	stem_activation[0][0]
block1a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block1a_dwconv[0][0]
block1a_activation (Activation)	(None, 128, 128, 32)	0	block1a_bn[0][0]
block1a_conv (Conv2D)	(None, 128, 128, 32)	864	block1a_activation[0][0]
block1a (Sequential)	(None, 128, 128, 32)	1280	block1a_conv[0][0]
block2a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block1a[0][0]
block2a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block2a_dwconv[0][0]
block2a_activation (Activation)	(None, 128, 128, 32)	0	block2a_bn[0][0]
block2a_conv (Conv2D)	(None, 128, 128, 32)	864	block2a_activation[0][0]
block2a (Sequential)	(None, 128, 128, 32)	1280	block2a_conv[0][0]
block3a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block2a[0][0]
block3a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block3a_dwconv[0][0]
block3a_activation (Activation)	(None, 128, 128, 32)	0	block3a_bn[0][0]
block3a_conv (Conv2D)	(None, 128, 128, 32)	864	block3a_activation[0][0]
block3a (Sequential)	(None, 128, 128, 32)	1280	block3a_conv[0][0]
block3b_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block3a[0][0]
block3b_bn (BatchNormalization)	(None, 128, 128, 32)	128	block3b_dwconv[0][0]
block3b_activation (Activation)	(None, 128, 128, 32)	0	block3b_bn[0][0]
block3b_conv (Conv2D)	(None, 128, 128, 32)	864	block3b_activation[0][0]
block3b (Sequential)	(None, 128, 128, 32)	1280	block3b_conv[0][0]
block4a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block3b[0][0]
block4a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block4a_dwconv[0][0]
block4a_activation (Activation)	(None, 128, 128, 32)	0	block4a_bn[0][0]
block4a_conv (Conv2D)	(None, 128, 128, 32)	864	block4a_activation[0][0]
block4a (Sequential)	(None, 128, 128, 32)	1280	block4a_conv[0][0]
block4b_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block4a[0][0]
block4b_bn (BatchNormalization)	(None, 128, 128, 32)	128	block4b_dwconv[0][0]
block4b_activation (Activation)	(None, 128, 128, 32)	0	block4b_bn[0][0]
block4b_conv (Conv2D)	(None, 128, 128, 32)	864	block4b_activation[0][0]
block4b (Sequential)	(None, 128, 128, 32)	1280	block4b_conv[0][0]
block5a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block4b[0][0]
block5a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block5a_dwconv[0][0]
block5a_activation (Activation)	(None, 128, 128, 32)	0	block5a_bn[0][0]
block5a_conv (Conv2D)	(None, 128, 128, 32)	864	block5a_activation[0][0]
block5a (Sequential)	(None, 128, 128, 32)	1280	block5a_conv[0][0]
block5b_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block5a[0][0]
block5b_bn (BatchNormalization)	(None, 128, 128, 32)	128	block5b_dwconv[0][0]
block5b_activation (Activation)	(None, 128, 128, 32)	0	block5b_bn[0][0]
block5b_conv (Conv2D)	(None, 128, 128, 32)	864	block5b_activation[0][0]
block5b (Sequential)	(None, 128, 128, 32)	1280	block5b_conv[0][0]
block5c_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block5b[0][0]
block5c_bn (BatchNormalization)	(None, 128, 128, 32)	128	block5c_dwconv[0][0]
block5c_activation (Activation)	(None, 128, 128, 32)	0	block5c_bn[0][0]
block5c_conv (Conv2D)	(None, 128, 128, 32)	864	block5c_activation[0][0]
block5c (Sequential)	(None, 128, 128, 32)	1280	block5c_conv[0][0]
block5d_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block5c[0][0]
block5d_bn (BatchNormalization)	(None, 128, 128, 32)	128	block5d_dwconv[0][0]
block5d_activation (Activation)	(None, 128, 128, 32)	0	block5d_bn[0][0]
block5d_conv (Conv2D)	(None, 128, 128, 32)	864	block5d_activation[0][0]
block5d (Sequential)	(None, 128, 128, 32)	1280	block5d_conv[0][0]
block5e_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block5d[0][0]
block5e_bn (BatchNormalization)	(None, 128, 128, 32)	128	block5e_dwconv[0][0]
block5e_activation (Activation)	(None, 128, 128, 32)	0	block5e_bn[0][0]
block5e_conv (Conv2D)	(None, 128, 128, 32)	864	block5e_activation[0][0]
block5e (Sequential)	(None, 128, 128, 32)	1280	block5e_conv[0][0]
block5f_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block5e[0][0]
block5f_bn (BatchNormalization)	(None, 128, 128, 32)	128	block5f_dwconv[0][0]
block5f_activation (Activation)	(None, 128, 128, 32)	0	block5f_bn[0][0]
block5f_conv (Conv2D)	(None, 128, 128, 32)	864	block5f_activation[0][0]
block5f (Sequential)	(None, 128, 128, 32)	1280	block5f_conv[0][0]
block6a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block5f[0][0]
block6a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6a_dwconv[0][0]
block6a_activation (Activation)	(None, 128, 128, 32)	0	block6a_bn[0][0]
block6a_conv (Conv2D)	(None, 128, 128, 32)	864	block6a_activation[0][0]
block6a (Sequential)	(None, 128, 128, 32)	1280	block6a_conv[0][0]
block6b_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6a[0][0]
block6b_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6b_dwconv[0][0]
block6b_activation (Activation)	(None, 128, 128, 32)	0	block6b_bn[0][0]
block6b_conv (Conv2D)	(None, 128, 128, 32)	864	block6b_activation[0][0]
block6b (Sequential)	(None, 128, 128, 32)	1280	block6b_conv[0][0]
block6c_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6b[0][0]
block6c_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6c_dwconv[0][0]
block6c_activation (Activation)	(None, 128, 128, 32)	0	block6c_bn[0][0]
block6c_conv (Conv2D)	(None, 128, 128, 32)	864	block6c_activation[0][0]
block6c (Sequential)	(None, 128, 128, 32)	1280	block6c_conv[0][0]
block6d_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6c[0][0]
block6d_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6d_dwconv[0][0]
block6d_activation (Activation)	(None, 128, 128, 32)	0	block6d_bn[0][0]
block6d_conv (Conv2D)	(None, 128, 128, 32)	864	block6d_activation[0][0]
block6d (Sequential)	(None, 128, 128, 32)	1280	block6d_conv[0][0]
block6e_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6d[0][0]
block6e_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6e_dwconv[0][0]
block6e_activation (Activation)	(None, 128, 128, 32)	0	block6e_bn[0][0]
block6e_conv (Conv2D)	(None, 128, 128, 32)	864	block6e_activation[0][0]
block6e (Sequential)	(None, 128, 128, 32)	1280	block6e_conv[0][0]
block6f_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6e[0][0]
block6f_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6f_dwconv[0][0]
block6f_activation (Activation)	(None, 128, 128, 32)	0	block6f_bn[0][0]
block6f_conv (Conv2D)	(None, 128, 128, 32)	864	block6f_activation[0][0]
block6f (Sequential)	(None, 128, 128, 32)	1280	block6f_conv[0][0]
block6g_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6f[0][0]
block6g_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6g_dwconv[0][0]
block6g_activation (Activation)	(None, 128, 128, 32)	0	block6g_bn[0][0]
block6g_conv (Conv2D)	(None, 128, 128, 32)	864	block6g_activation[0][0]
block6g (Sequential)	(None, 128, 128, 32)	1280	block6g_conv[0][0]
block6h_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6g[0][0]
block6h_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6h_dwconv[0][0]
block6h_activation (Activation)	(None, 128, 128, 32)	0	block6h_bn[0][0]
block6h_conv (Conv2D)	(None, 128, 128, 32)	864	block6h_activation[0][0]
block6h (Sequential)	(None, 128, 128, 32)	1280	block6h_conv[0][0]
block6i_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6h[0][0]
block6i_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6i_dwconv[0][0]
block6i_activation (Activation)	(None, 128, 128, 32)	0	block6i_bn[0][0]
block6i_conv (Conv2D)	(None, 128, 128, 32)	864	block6i_activation[0][0]
block6i (Sequential)	(None, 128, 128, 32)	1280	block6i_conv[0][0]
block6j_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6i[0][0]
block6j_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6j_dwconv[0][0]
block6j_activation (Activation)	(None, 128, 128, 32)	0	block6j_bn[0][0]
block6j_conv (Conv2D)	(None, 128, 128, 32)	864	block6j_activation[0][0]
block6j (Sequential)	(None, 128, 128, 32)	1280	block6j_conv[0][0]
block6k_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6j[0][0]
block6k_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6k_dwconv[0][0]
block6k_activation (Activation)	(None, 128, 128, 32)	0	block6k_bn[0][0]
block6k_conv (Conv2D)	(None, 128, 128, 32)	864	block6k_activation[0][0]
block6k (Sequential)	(None, 128, 128, 32)	1280	block6k_conv[0][0]
block6l_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6k[0][0]
block6l_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6l_dwconv[0][0]
block6l_activation (Activation)	(None, 128, 128, 32)	0	block6l_bn[0][0]
block6l_conv (Conv2D)	(None, 128, 128, 32)	864	block6l_activation[0][0]
block6l (Sequential)	(None, 128, 128, 32)	1280	block6l_conv[0][0]
block6m_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6l[0][0]
block6m_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6m_dwconv[0][0]
block6m_activation (Activation)	(None, 128, 128, 32)	0	block6m_bn[0][0]
block6m_conv (Conv2D)	(None, 128, 128, 32)	864	block6m_activation[0][0]
block6m (Sequential)	(None, 128, 128, 32)	1280	block6m_conv[0][0]
block6n_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6m[0][0]
block6n_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6n_dwconv[0][0]
block6n_activation (Activation)	(None, 128, 128, 32)	0	block6n_bn[0][0]
block6n_conv (Conv2D)	(None, 128, 128, 32)	864	block6n_activation[0][0]
block6n (Sequential)	(None, 128, 128, 32)	1280	block6n_conv[0][0]
block6o_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6n[0][0]
block6o_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6o_dwconv[0][0]
block6o_activation (Activation)	(None, 128, 128, 32)	0	block6o_bn[0][0]
block6o_conv (Conv2D)	(None, 128, 128, 32)	864	block6o_activation[0][0]
block6o (Sequential)	(None, 128, 128, 32)	1280	block6o_conv[0][0]
block6p_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6o[0][0]
block6p_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6p_dwconv[0][0]
block6p_activation (Activation)	(None, 128, 128, 32)	0	block6p_bn[0][0]
block6p_conv (Conv2D)	(None, 128, 128, 32)	864	block6p_activation[0][0]
block6p (Sequential)	(None, 128, 128, 32)	1280	block6p_conv[0][0]
block6q_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6p[0][0]
block6q_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6q_dwconv[0][0]
block6q_activation (Activation)	(None, 128, 128, 32)	0	block6q_bn[0][0]
block6q_conv (Conv2D)	(None, 128, 128, 32)	864	block6q_activation[0][0]
block6q (Sequential)	(None, 128, 128, 32)	1280	block6q_conv[0][0]
block6r_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6q[0][0]
block6r_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6r_dwconv[0][0]
block6r_activation (Activation)	(None, 128, 128, 32)	0	block6r_bn[0][0]
block6r_conv (Conv2D)	(None, 128, 128, 32)	864	block6r_activation[0][0]
block6r (Sequential)	(None, 128, 128, 32)	1280	block6r_conv[0][0]
block6s_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6r[0][0]
block6s_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6s_dwconv[0][0]
block6s_activation (Activation)	(None, 128, 128, 32)	0	block6s_bn[0][0]
block6s_conv (Conv2D)	(None, 128, 128, 32)	864	block6s_activation[0][0]
block6s (Sequential)	(None, 128, 128, 32)	1280	block6s_conv[0][0]
block6t_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6s[0][0]
block6t_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6t_dwconv[0][0]
block6t_activation (Activation)	(None, 128, 128, 32)	0	block6t_bn[0][0]
block6t_conv (Conv2D)	(None, 128, 128, 32)	864	block6t_activation[0][0]
block6t (Sequential)	(None, 128, 128, 32)	1280	block6t_conv[0][0]
block6u_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6t[0][0]
block6u_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6u_dwconv[0][0]
block6u_activation (Activation)	(None, 128, 128, 32)	0	block6u_bn[0][0]
block6u_conv (Conv2D)	(None, 128, 128, 32)	864	block6u_activation[0][0]
block6u (Sequential)	(None, 128, 128, 32)	1280	block6u_conv[0][0]
block6v_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6u[0][0]
block6v_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6v_dwconv[0][0]
block6v_activation (Activation)	(None, 128, 128, 32)	0	block6v_bn[0][0]
block6v_conv (Conv2D)	(None, 128, 128, 32)	864	block6v_activation[0][0]
block6v (Sequential)	(None, 128, 128, 32)	1280	block6v_conv[0][0]
block6w_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6v[0][0]
block6w_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6w_dwconv[0][0]
block6w_activation (Activation)	(None, 128, 128, 32)	0	block6w_bn[0][0]
block6w_conv (Conv2D)	(None, 128, 128, 32)	864	block6w_activation[0][0]
block6w (Sequential)	(None, 128, 128, 32)	1280	block6w_conv[0][0]
block6x_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6w[0][0]
block6x_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6x_dwconv[0][0]
block6x_activation (Activation)	(None, 128, 128, 32)	0	block6x_bn[0][0]
block6x_conv (Conv2D)	(None, 128, 128, 32)	864	block6x_activation[0][0]
block6x (Sequential)	(None, 128, 128, 32)	1280	block6x_conv[0][0]
block6y_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6x[0][0]
block6y_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6y_dwconv[0][0]
block6y_activation (Activation)	(None, 128, 128, 32)	0	block6y_bn[0][0]
block6y_conv (Conv2D)	(None, 128, 128, 32)	864	block6y_activation[0][0]
block6y (Sequential)	(None, 128, 128, 32)	1280	block6y_conv[0][0]
block6z_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6y[0][0]
block6z_bn (BatchNormalization)	(None, 128, 128, 32)	128	block6z_dwconv[0][0]
block6z_activation (Activation)	(None, 128, 128, 32)	0	block6z_bn[0][0]
block6z_conv (Conv2D)	(None, 128, 128, 32)	864	block6z_activation[0][0]
block6z (Sequential)	(None, 128, 128, 32)	1280	block6z_conv[0][0]
block7a_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block6z[0][0]
block7a_bn (BatchNormalization)	(None, 128, 128, 32)	128	block7a_dwconv[0][0]
block7a_activation (Activation)	(None, 128, 128, 32)	0	block7a_bn[0][0]
block7a_conv (Conv2D)	(None, 128, 128, 32)	864	block7a_activation[0][0]
block7a (Sequential)	(None, 128, 128, 32)	1280	block7a_conv[0][0]
block7b_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block7a[0][0]
block7b_bn (BatchNormalization)	(None, 128, 128, 32)	128	block7b_dwconv[0][0]
block7b_activation (Activation)	(None, 128, 128, 32)	0	block7b_bn[0][0]
block7b_conv (Conv2D)	(None, 128, 128, 32)	864	block7b_activation[0][0]
block7b (Sequential)	(None, 128, 128, 32)	1280	block7b_conv[0][0]
block7c_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block7b[0][0]
block7c_bn (BatchNormalization)	(None, 128, 128, 32)	128	block7c_dwconv[0][0]
block7c_activation (Activation)	(None, 128, 128, 32)	0	block7c_bn[0][0]
block7c_conv (Conv2D)	(None, 128, 128, 32)	864	block7c_activation[0][0]
block7c (Sequential)	(None, 128, 128, 32)	1280	block7c_conv[0][0]
block7d_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block7c[0][0]
block7d_bn (BatchNormalization)	(None, 128, 128, 32)	128	block7d_dwconv[0][0]
block7d_activation (Activation)	(None, 128, 128, 32)	0	block7d_bn[0][0]
block7d_conv (Conv2D)	(None, 128, 128, 32)	864	block7d_activation[0][0]
block7d (Sequential)	(None, 128, 128, 32)	1280	block7d_conv[0][0]
block7e_dwconv (DepthwiseConv2D)	(None, 128, 128, 32)	288	block7d[0][0]
block7e_bn (BatchNormalization)	(None, 128, 128, 32)	128	block7e_dwconv[0][0]
block7e_activation (Activation)	(None,		

```
In [27]: import joblib
```

```
In [28]: def get_scores(data_gen, file_name='scores.pkl'):
          scores = model.evaluate_generator(data_gen, verbose=1)
          joblib.dump(scores, file_name)
          print(f"Loss: {scores[0]} and Accuracy: {scores[1]*100}")
```

```
In [29]: get_scores(data_gen=data generator train, file name='train scores.pkl')
```

4215/4215 [=====] - 9113s 2s/step

Loss: 0.03487127274274826 and Accuracy: 94.0958321094513

```
In [30]: get_scores(data_gen=data generator val, file name='val_scores.pkl')
```

1054/1054 [=====] - 2257s 2s/step

Loss: 0.18287695944309235 and Accuracy: 94.08394694328308

```
In [31]: test_preds = model.predict_generator(data_generator_test, verbose=1)
```

```
948/948 [=====] - 2091s 2s/step
```

```
In [32]: test_preds[:5]
```

```
Out[32]: array([[6.1310172e-02, 8.7240338e-04, 2.7319193e-03, 4.8130751e-04,
                1.6119182e-03, 3.6428809e-02],
                [2.5629997e-06, 4.7683716e-07, 5.9604645e-08, 0.0000000e+00,
                3.2186508e-06, 1.4901161e-06],
                [3.1926930e-03, 2.0837784e-04, 8.2939863e-05, 1.0672212e-04,
                5.0994754e-04, 2.1046698e-03],
                [1.7616540e-02, 9.5963478e-05, 1.5140772e-03, 8.2284212e-04,
                2.4288595e-03, 1.0744154e-02],
                [1.5471071e-02, 6.6035986e-04, 2.0802021e-04, 5.2440166e-04,
                1.3437271e-03, 2.4896264e-02]], dtype=float32)
```

```
In [33]: test_preds.shape
```















```
Out[33]: (121344, 6)
```

As test labels are not disclosed as part of competition and we only get the score after submitting the file.

### Private leaderboard score





Overview	Data	Notebooks	Discussion	Leaderboard	Rules	Team	My Submissions	Late Submission	
272	▲13	kagglebaggledaggle				    	0.07081	6	2d
273	▲109	TrollFactory					0.07107	5	3d
274	▲109	luckylu					0.07108	2	3d
275	▼24	YaGana Sheriff-Hussaini					0.07124	16	2d
276	▼74	Incnas					0.07156	1	3d
277	▲117	Mike Kim					0.07213	4	5d
278	▼91	Cedric Damien					0.07267	2	3d
279	▲91	Surya Parsa					0.07267	2	6d
280	▲91	Mukesh					0.07267	1	3d
281	▲91	student					0.07267	1	5d