sota_neural_network_architectures_erick

December 16, 2018

1 Homework- 26.11.2018:

1.1 State of the Art Neural Network Architectures

The purpose of this homework is to implement and evaluate the sota architectures presented in the lecture. However, you are encouraged to try your own layer module ideas. Feel free to consult the Keras source code:

1. Based on the CNN modules presented in the lecture e.g. VGG16, Inception, ResNet, Xception, DenseNet, come up with your own CNN module and write a small text discussing your idea and motivations behind the module.

2. Evaluate all your module using the Keras CIFAR10 dataset splits (The model with best test accuracy will present their solution to the class).

1.2 Original ResNet implementation for CIFAR10 dataset

In [0]:
```
"""Trains a ResNet on the CIFAR10 dataset.
ResNet v1
[a] Deep Residual Learning for Image Recognition
https://arxiv.org/pdf/1512.03385.pdf
ResNet v2
[b] Identity Mappings in Deep Residual Networks
https://arxiv.org/pdf/1603.05027.pdf
"""

from __future__ import print_function
import keras
from keras.layers import Dense, Conv2D, BatchNormalization, Activation
from keras.layers import AveragePooling2D, Input, Flatten
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from keras.regularizers import l2
from keras import backend as K
from keras.models import Model
from keras.datasets import cifar10
```

```python
import numpy as np
import os

# Training parameters
batch_size = 128  # orig paper trained all networks with batch_size=128
epochs = 50
data_augmentation = True
num_classes = 10

# Subtracting pixel mean improves accuracy
subtract_pixel_mean = True

n = 3

# Model version
# Orig paper: version = 1 (ResNet v1), Improved ResNet: version = 2 (ResNet v2)
version = 2

# Computed depth from supplied model parameter n
if version == 1:
    depth = n * 6 + 2
elif version == 2:
    depth = n * 9 + 2

# Model name, depth and version
model_type = 'ResNet%dv%d' % (depth, version)

# Load the CIFAR10 data.
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Input image dimensions.
input_shape = x_train.shape[1:]

# Normalize data.
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# If subtract pixel mean is enabled
if subtract_pixel_mean:
    x_train_mean = np.mean(x_train, axis=0)
    x_train -= x_train_mean
    x_test -= x_train_mean

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
print('y_train shape:', y_train.shape)
```

```python
# Convert class vectors to binary class matrices.
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)


def lr_schedule(epoch):
    """Learning Rate Schedule
    Learning rate is scheduled to be reduced after 80, 120, 160, 180 epochs.
    Called automatically every epoch as part of callbacks during training.
    # Arguments
        epoch (int): The number of epochs
    # Returns
        lr (float32): learning rate
    """
    lr = 1e-3
    if epoch > 180:
        lr *= 0.5e-3
    elif epoch > 160:
        lr *= 1e-3
    elif epoch > 120:
        lr *= 1e-2
    elif epoch > 80:
        lr *= 1e-1
    print('Learning rate: ', lr)
    return lr


def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):
    """2D Convolution-Batch Normalization-Activation stack builder
    # Arguments
        inputs (tensor): input tensor from input image or previous layer
        num_filters (int): Conv2D number of filters
        kernel_size (int): Conv2D square kernel dimensions
        strides (int): Conv2D square stride dimensions
        activation (string): activation name
        batch_normalization (bool): whether to include batch normalization
        conv_first (bool): conv-bn-activation (True) or
            bn-activation-conv (False)
    # Returns
        x (tensor): tensor as input to the next layer
    """
    conv = Conv2D(num_filters,
```

```
                  kernel_size=kernel_size,
                  strides=strides,
                  padding='same',
                  kernel_initializer='he_normal',
                  kernel_regularizer=l2(1e-4))

    x = inputs
    if conv_first:
        x = conv(x)
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
    else:
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
        x = conv(x)
    return x


def resnet_v1(input_shape, depth, num_classes=10):
    """ResNet Version 1 Model builder [a]
    Stacks of 2 x (3 x 3) Conv2D-BN-ReLU
    Last ReLU is after the shortcut connection.
    At the beginning of each stage, the feature map size is halved (downsampled)
    by a convolutional layer with strides=2, while the number of filters is
    doubled. Within each stage, the layers have the same number filters and the
    same number of filters.
    Features maps sizes:
    stage 0: 32x32, 16
    stage 1: 16x16, 32
    stage 2:  8x8,  64
    The Number of parameters is approx the same as Table 6 of [a]:
    ResNet20 0.27M
    ResNet32 0.46M
    ResNet44 0.66M
    ResNet56 0.85M
    ResNet110 1.7M
    # Arguments
        input_shape (tensor): shape of input image tensor
        depth (int): number of core convolutional layers
        num_classes (int): number of classes (CIFAR10 has 10)
    # Returns
        model (Model): Keras model instance
    """
    if (depth - 2) % 6 != 0:
```

```
        raise ValueError('depth should be 6n+2 (eg 20, 32, 44 in [a])')
    # Start model definition.
    num_filters = 16
    num_res_blocks = int((depth - 2) / 6)

    inputs = Input(shape=input_shape)
    x = resnet_layer(inputs=inputs)
    # Instantiate the stack of residual units
    for stack in range(3):
        for res_block in range(num_res_blocks):
            strides = 1
            if stack > 0 and res_block == 0:  # first layer but not first stack
                strides = 2  # downsample
            y = resnet_layer(inputs=x,
                             num_filters=num_filters,
                             strides=strides)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters,
                             activation=None)
            if stack > 0 and res_block == 0:  # first layer but not first stack
                # linear projection residual shortcut connection to match
                # changed dims
                x = resnet_layer(inputs=x,
                                 num_filters=num_filters,
                                 kernel_size=1,
                                 strides=strides,
                                 activation=None,
                                 batch_normalization=False)
            x = keras.layers.add([x, y])
            x = Activation('relu')(x)
        num_filters *= 2

    # Add classifier on top.
    # v1 does not use BN after last shortcut connection-ReLU
    x = AveragePooling2D(pool_size=8)(x)
    y = Flatten()(x)
    outputs = Dense(num_classes,
                    activation='softmax',
                    kernel_initializer='he_normal')(y)

    # Instantiate model.
    model = Model(inputs=inputs, outputs=outputs)
    return model


def resnet_v2(input_shape, depth, num_classes=10):
    """ResNet Version 2 Model builder [b]
    Stacks of (1 x 1)-(3 x 3)-(1 x 1) BN-ReLU-Conv2D or also known as
```

```
    bottleneck layer
    First shortcut connection per layer is 1 x 1 Conv2D.
    Second and onwards shortcut connection is identity.
    At the beginning of each stage, the feature map size is halved (downsampled)
    by a convolutional layer with strides=2, while the number of filter maps is
    doubled. Within each stage, the layers have the same number filters and the
    same filter map sizes.
    Features maps sizes:
    conv1  : 32x32,  16
    stage 0: 32x32,  64
    stage 1: 16x16, 128
    stage 2:  8x8,  256
    # Arguments
        input_shape (tensor): shape of input image tensor
        depth (int): number of core convolutional layers
        num_classes (int): number of classes (CIFAR10 has 10)
    # Returns
        model (Model): Keras model instance
    """
    if (depth - 2) % 9 != 0:
        raise ValueError('depth should be 9n+2 (eg 56 or 110 in [b])')
    # Start model definition.
    num_filters_in = 16
    num_res_blocks = int((depth - 2) / 9)

    inputs = Input(shape=input_shape)
    # v2 performs Conv2D with BN-ReLU on input before splitting into 2 paths
    x = resnet_layer(inputs=inputs,
                     num_filters=num_filters_in,
                     conv_first=True)

    # Instantiate the stack of residual units
    for stage in range(3):
        for res_block in range(num_res_blocks):
            activation = 'relu'
            batch_normalization = True
            strides = 1
            if stage == 0:
                num_filters_out = num_filters_in * 4
                if res_block == 0:  # first layer and first stage
                    activation = None
                    batch_normalization = False
            else:
                num_filters_out = num_filters_in * 2
                if res_block == 0:  # first layer but not first stage
                    strides = 2    # downsample

            # bottleneck residual unit
```

```python
            y = resnet_layer(inputs=x,
                             num_filters=num_filters_in,
                             kernel_size=1,
                             strides=strides,
                             activation=activation,
                             batch_normalization=batch_normalization,
                             conv_first=False)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters_in,
                             conv_first=False)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters_out,
                             kernel_size=1,
                             conv_first=False)
            if res_block == 0:
                # linear projection residual shortcut connection to match
                # changed dims
                x = resnet_layer(inputs=x,
                                 num_filters=num_filters_out,
                                 kernel_size=1,
                                 strides=strides,
                                 activation=None,
                                 batch_normalization=False)
            x = keras.layers.add([x, y])

        num_filters_in = num_filters_out

    # Add classifier on top.
    # v2 has BN-ReLU before Pooling
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = AveragePooling2D(pool_size=8)(x)
    y = Flatten()(x)
    outputs = Dense(num_classes,
                    activation='softmax',
                    kernel_initializer='he_normal')(y)

    # Instantiate model.
    model = Model(inputs=inputs, outputs=outputs)
    return model


if version == 2:
    model = resnet_v2(input_shape=input_shape, depth=depth)
else:
    model = resnet_v1(input_shape=input_shape, depth=depth)

model.compile(loss='categorical_crossentropy',
```

```python
              optimizer=Adam(lr=lr_schedule(0)),
              metrics=['accuracy'])
model.summary()
print(model_type)

# Prepare model model saving directory.
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'cifar10_%s_model.{epoch:03d}.h5' % model_type
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
filepath = os.path.join(save_dir, model_name)

# Prepare callbacks for model saving and for learning rate adjustment.
checkpoint = ModelCheckpoint(filepath=filepath,
                             monitor='val_acc',
                             verbose=1,
                             save_best_only=True)

lr_scheduler = LearningRateScheduler(lr_schedule)

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               min_lr=0.5e-6)

callbacks = [checkpoint, lr_reducer, lr_scheduler]

# Run training, with or without data augmentation.
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              validation_data=(x_test, y_test),
              shuffle=True,
              callbacks=callbacks)
else:
    print('Using real-time data augmentation.')
    # This will do preprocessing and realtime data augmentation:
    datagen = ImageDataGenerator(
        # set input mean to 0 over the dataset
        featurewise_center=False,
        # set each sample mean to 0
        samplewise_center=False,
        # divide inputs by std of dataset
        featurewise_std_normalization=False,
        # divide each input by its std
        samplewise_std_normalization=False,
```

```python
        # apply ZCA whitening
        zca_whitening=False,
        # epsilon for ZCA whitening
        zca_epsilon=1e-06,
        # randomly rotate images in the range (deg 0 to 180)
        rotation_range=0,
        # randomly shift images horizontally
        width_shift_range=0.1,
        # randomly shift images vertically
        height_shift_range=0.1,
        # set range for random shear
        shear_range=0.,
        # set range for random zoom
        zoom_range=0.,
        # set range for random channel shifts
        channel_shift_range=0.,
        # set mode for filling points outside the input boundaries
        fill_mode='nearest',
        # value used for fill_mode = "constant"
        cval=0.,
        # randomly flip images
        horizontal_flip=True,
        # randomly flip images
        vertical_flip=False,
        # set rescaling factor (applied before any other transformation)
        rescale=None,
        # set function that will be applied on each input
        preprocessing_function=None,
        # image data format, either "channels_first" or "channels_last"
        data_format=None,
        # fraction of images reserved for validation (strictly between 0 and 1)
        validation_split=0.0)

    # Compute quantities required for featurewise normalization
    # (std, mean, and principal components if ZCA whitening is applied).
    datagen.fit(x_train)

    # Fit the model on the batches generated by datagen.flow().
    model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_size),
                        validation_data=(x_test, y_test),
                        epochs=epochs, verbose=1, workers=4,steps_per_epoch=1000,
                        callbacks=callbacks)

# Score trained model.
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

```
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
y_train shape: (50000, 1)
Learning rate:  0.001
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 32, 32, 3)    0
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 32, 32, 16)   448         input_2[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, 32, 32, 16)   64          conv2d_32[0][0]
__________________________________________________________________________________________________
activation_29 (Activation)      (None, 32, 32, 16)   0           batch_normalization_29[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 32, 32, 16)   272         activation_29[0][0]
__________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 32, 32, 16)   64          conv2d_33[0][0]
__________________________________________________________________________________________________
activation_30 (Activation)      (None, 32, 32, 16)   0           batch_normalization_30[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 32, 32, 16)   2320        activation_30[0][0]
__________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 32, 32, 16)   64          conv2d_34[0][0]
__________________________________________________________________________________________________
activation_31 (Activation)      (None, 32, 32, 16)   0           batch_normalization_31[0][0]
__________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 32, 32, 64)   1088        activation_29[0][0]
__________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 32, 32, 64)   1088        activation_31[0][0]
__________________________________________________________________________________________________
add_10 (Add)                    (None, 32, 32, 64)   0           conv2d_36[0][0]
                                                                 conv2d_35[0][0]
__________________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 32, 32, 64)   256         add_10[0][0]
__________________________________________________________________________________________________
activation_32 (Activation)      (None, 32, 32, 64)   0           batch_normalization_32[0][0]
__________________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, 32, 32, 16)   1040        activation_32[0][0]
__________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 32, 32, 16)   64          conv2d_37[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, 32, 32, 16)   0           batch_normalization_33[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 32, 32, 16)   2320        activation_33[0][0]
__________________________________________________________________________________________________
batch_normalization_34 (BatchNo (None, 32, 32, 16)   64          conv2d_38[0][0]
```

```
__________________________________________________________________________________________________
activation_34 (Activation)      (None, 32, 32, 16)   0           batch_normalization_34[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 32, 32, 64)   1088        activation_34[0][0]
__________________________________________________________________________________________________
add_11 (Add)                    (None, 32, 32, 64)   0           add_10[0][0]
                                                                 conv2d_39[0][0]
__________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, 32, 32, 64)   256         add_11[0][0]
__________________________________________________________________________________________________
activation_35 (Activation)      (None, 32, 32, 64)   0           batch_normalization_35[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 32, 32, 16)   1040        activation_35[0][0]
__________________________________________________________________________________________________
batch_normalization_36 (BatchNo (None, 32, 32, 16)   64          conv2d_40[0][0]
__________________________________________________________________________________________________
activation_36 (Activation)      (None, 32, 32, 16)   0           batch_normalization_36[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 32, 32, 16)   2320        activation_36[0][0]
__________________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 32, 32, 16)   64          conv2d_41[0][0]
__________________________________________________________________________________________________
activation_37 (Activation)      (None, 32, 32, 16)   0           batch_normalization_37[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, 32, 32, 64)   1088        activation_37[0][0]
__________________________________________________________________________________________________
add_12 (Add)                    (None, 32, 32, 64)   0           add_11[0][0]
                                                                 conv2d_42[0][0]
__________________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, 32, 32, 64)   256         add_12[0][0]
__________________________________________________________________________________________________
activation_38 (Activation)      (None, 32, 32, 64)   0           batch_normalization_38[0][0]
__________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, 16, 16, 64)   4160        activation_38[0][0]
__________________________________________________________________________________________________
batch_normalization_39 (BatchNo (None, 16, 16, 64)   256         conv2d_43[0][0]
__________________________________________________________________________________________________
activation_39 (Activation)      (None, 16, 16, 64)   0           batch_normalization_39[0][0]
__________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, 16, 16, 64)   36928       activation_39[0][0]
__________________________________________________________________________________________________
batch_normalization_40 (BatchNo (None, 16, 16, 64)   256         conv2d_44[0][0]
__________________________________________________________________________________________________
activation_40 (Activation)      (None, 16, 16, 64)   0           batch_normalization_40[0][0]
__________________________________________________________________________________________________
conv2d_46 (Conv2D)              (None, 16, 16, 128)  8320        add_12[0][0]
__________________________________________________________________________________________________
conv2d_45 (Conv2D)              (None, 16, 16, 128)  8320        activation_40[0][0]
```

```
__________________________________________________________________________________________________
add_13 (Add)                    (None, 16, 16, 128)  0           conv2d_46[0][0]
                                                                 conv2d_45[0][0]
__________________________________________________________________________________________________
batch_normalization_41 (BatchNo (None, 16, 16, 128)  512         add_13[0][0]
__________________________________________________________________________________________________
activation_41 (Activation)      (None, 16, 16, 128)  0           batch_normalization_41[0][0]
__________________________________________________________________________________________________
conv2d_47 (Conv2D)              (None, 16, 16, 64)   8256        activation_41[0][0]
__________________________________________________________________________________________________
batch_normalization_42 (BatchNo (None, 16, 16, 64)   256         conv2d_47[0][0]
__________________________________________________________________________________________________
activation_42 (Activation)      (None, 16, 16, 64)   0           batch_normalization_42[0][0]
__________________________________________________________________________________________________
conv2d_48 (Conv2D)              (None, 16, 16, 64)   36928       activation_42[0][0]
__________________________________________________________________________________________________
batch_normalization_43 (BatchNo (None, 16, 16, 64)   256         conv2d_48[0][0]
__________________________________________________________________________________________________
activation_43 (Activation)      (None, 16, 16, 64)   0           batch_normalization_43[0][0]
__________________________________________________________________________________________________
conv2d_49 (Conv2D)              (None, 16, 16, 128)  8320        activation_43[0][0]
__________________________________________________________________________________________________
add_14 (Add)                    (None, 16, 16, 128)  0           add_13[0][0]
                                                                 conv2d_49[0][0]
__________________________________________________________________________________________________
batch_normalization_44 (BatchNo (None, 16, 16, 128)  512         add_14[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, 16, 16, 128)  0           batch_normalization_44[0][0]
__________________________________________________________________________________________________
conv2d_50 (Conv2D)              (None, 16, 16, 64)   8256        activation_44[0][0]
__________________________________________________________________________________________________
batch_normalization_45 (BatchNo (None, 16, 16, 64)   256         conv2d_50[0][0]
__________________________________________________________________________________________________
activation_45 (Activation)      (None, 16, 16, 64)   0           batch_normalization_45[0][0]
__________________________________________________________________________________________________
conv2d_51 (Conv2D)              (None, 16, 16, 64)   36928       activation_45[0][0]
__________________________________________________________________________________________________
batch_normalization_46 (BatchNo (None, 16, 16, 64)   256         conv2d_51[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, 16, 16, 64)   0           batch_normalization_46[0][0]
__________________________________________________________________________________________________
conv2d_52 (Conv2D)              (None, 16, 16, 128)  8320        activation_46[0][0]
__________________________________________________________________________________________________
add_15 (Add)                    (None, 16, 16, 128)  0           add_14[0][0]
                                                                 conv2d_52[0][0]
__________________________________________________________________________________________________
batch_normalization_47 (BatchNo (None, 16, 16, 128)  512         add_15[0][0]
__________________________________________________________________________________________________
```

activation_47 (Activation)	(None, 16, 16, 128)	0	batch_normalization_47[0][0]
conv2d_53 (Conv2D)	(None, 8, 8, 128)	16512	activation_47[0][0]
batch_normalization_48 (BatchNo	(None, 8, 8, 128)	512	conv2d_53[0][0]
activation_48 (Activation)	(None, 8, 8, 128)	0	batch_normalization_48[0][0]
conv2d_54 (Conv2D)	(None, 8, 8, 128)	147584	activation_48[0][0]
batch_normalization_49 (BatchNo	(None, 8, 8, 128)	512	conv2d_54[0][0]
activation_49 (Activation)	(None, 8, 8, 128)	0	batch_normalization_49[0][0]
conv2d_56 (Conv2D)	(None, 8, 8, 256)	33024	add_15[0][0]
conv2d_55 (Conv2D)	(None, 8, 8, 256)	33024	activation_49[0][0]
add_16 (Add)	(None, 8, 8, 256)	0	conv2d_56[0][0] conv2d_55[0][0]
batch_normalization_50 (BatchNo	(None, 8, 8, 256)	1024	add_16[0][0]
activation_50 (Activation)	(None, 8, 8, 256)	0	batch_normalization_50[0][0]
conv2d_57 (Conv2D)	(None, 8, 8, 128)	32896	activation_50[0][0]
batch_normalization_51 (BatchNo	(None, 8, 8, 128)	512	conv2d_57[0][0]
activation_51 (Activation)	(None, 8, 8, 128)	0	batch_normalization_51[0][0]
conv2d_58 (Conv2D)	(None, 8, 8, 128)	147584	activation_51[0][0]
batch_normalization_52 (BatchNo	(None, 8, 8, 128)	512	conv2d_58[0][0]
activation_52 (Activation)	(None, 8, 8, 128)	0	batch_normalization_52[0][0]
conv2d_59 (Conv2D)	(None, 8, 8, 256)	33024	activation_52[0][0]
add_17 (Add)	(None, 8, 8, 256)	0	add_16[0][0] conv2d_59[0][0]
batch_normalization_53 (BatchNo	(None, 8, 8, 256)	1024	add_17[0][0]
activation_53 (Activation)	(None, 8, 8, 256)	0	batch_normalization_53[0][0]
conv2d_60 (Conv2D)	(None, 8, 8, 128)	32896	activation_53[0][0]

```
batch_normalization_54 (BatchNo (None, 8, 8, 128)    512         conv2d_60[0][0]
________________________________________________________________________________________________
activation_54 (Activation)      (None, 8, 8, 128)    0           batch_normalization_54[0][0]
________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, 8, 8, 128)    147584      activation_54[0][0]
________________________________________________________________________________________________
batch_normalization_55 (BatchNo (None, 8, 8, 128)    512         conv2d_61[0][0]
________________________________________________________________________________________________
activation_55 (Activation)      (None, 8, 8, 128)    0           batch_normalization_55[0][0]
________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, 8, 8, 256)    33024       activation_55[0][0]
________________________________________________________________________________________________
add_18 (Add)                    (None, 8, 8, 256)    0           add_17[0][0]
                                                                 conv2d_62[0][0]
________________________________________________________________________________________________
batch_normalization_56 (BatchNo (None, 8, 8, 256)    1024        add_18[0][0]
________________________________________________________________________________________________
activation_56 (Activation)      (None, 8, 8, 256)    0           batch_normalization_56[0][0]
________________________________________________________________________________________________
average_pooling2d_2 (AveragePoo (None, 1, 1, 256)    0           activation_56[0][0]
________________________________________________________________________________________________
flatten_2 (Flatten)             (None, 256)          0           average_pooling2d_2[0][0]
________________________________________________________________________________________________
dense_2 (Dense)                 (None, 10)           2570        flatten_2[0][0]
================================================================================================
Total params: 849,002
Trainable params: 843,786
Non-trainable params: 5,216
________________________________________________________________________________________________
ResNet29v2
Using real-time data augmentation.
Epoch 1/50
Learning rate:  0.001
1000/1000 [==============================] - 216s 216ms/step - loss: 1.6375 - acc: 0.5708 - val_

Epoch 00001: val_acc improved from -inf to 0.66120, saving model to /content/saved_models/cifar1
Epoch 2/50
Learning rate:  0.001
1000/1000 [==============================] - 204s 204ms/step - loss: 1.1090 - acc: 0.7302 - val_

Epoch 00002: val_acc did not improve from 0.66120
Epoch 3/50
Learning rate:  0.001
1000/1000 [==============================] - 204s 204ms/step - loss: 0.9232 - acc: 0.7851 - val_

Epoch 00003: val_acc improved from 0.66120 to 0.66380, saving model to /content/saved_models/cif
Epoch 4/50
Learning rate:  0.001
```

```
1000/1000 [==============================] - 204s 204ms/step - loss: 0.8154 - acc: 0.8156 - val_

Epoch 00004: val_acc improved from 0.66380 to 0.76920, saving model to /content/saved_models/cif
Epoch 5/50
Learning rate:  0.001
1000/1000 [==============================] - 204s 204ms/step - loss: 0.7485 - acc: 0.8344 - val_

Epoch 00005: val_acc did not improve from 0.76920
Epoch 6/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.6888 - acc: 0.8506 - val_

Epoch 00006: val_acc did not improve from 0.76920
Epoch 7/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.6496 - acc: 0.8625 - val_

Epoch 00007: val_acc did not improve from 0.76920
Epoch 8/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.6164 - acc: 0.8708 - val_

Epoch 00008: val_acc did not improve from 0.76920
Epoch 9/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.5948 - acc: 0.8769 - val_

Epoch 00009: val_acc improved from 0.76920 to 0.79250, saving model to /content/saved_models/cif
Epoch 10/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.5724 - acc: 0.8840 - val_

Epoch 00010: val_acc did not improve from 0.79250
Epoch 11/50
Learning rate:  0.001
1000/1000 [==============================] - 210s 210ms/step - loss: 0.5561 - acc: 0.8890 - val_

Epoch 00011: val_acc improved from 0.79250 to 0.83710, saving model to /content/saved_models/cif
Epoch 12/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.5356 - acc: 0.8952 - val_

Epoch 00012: val_acc did not improve from 0.83710
Epoch 13/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.5194 - acc: 0.8995 - val_

Epoch 00013: val_acc did not improve from 0.83710
```

```
Epoch 14/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.5119 - acc: 0.9011 - val_

Epoch 00014: val_acc improved from 0.83710 to 0.84890, saving model to /content/saved_models/cif
Epoch 15/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4971 - acc: 0.9064 - val_

Epoch 00015: val_acc did not improve from 0.84890
Epoch 16/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4897 - acc: 0.9079 - val_

Epoch 00016: val_acc did not improve from 0.84890
Epoch 17/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4774 - acc: 0.9123 - val_

Epoch 00017: val_acc did not improve from 0.84890
Epoch 18/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.4705 - acc: 0.9148 - val_

Epoch 00018: val_acc improved from 0.84890 to 0.85160, saving model to /content/saved_models/cif
Epoch 19/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4638 - acc: 0.9159 - val_

Epoch 00019: val_acc improved from 0.85160 to 0.86300, saving model to /content/saved_models/cif
Epoch 20/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4584 - acc: 0.9175 - val_

Epoch 00020: val_acc did not improve from 0.86300
Epoch 21/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.4517 - acc: 0.9200 - val_

Epoch 00021: val_acc did not improve from 0.86300
Epoch 22/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4475 - acc: 0.9207 - val_

Epoch 00022: val_acc did not improve from 0.86300
Epoch 23/50
Learning rate:  0.001
1000/1000 [==============================] - 205s 205ms/step - loss: 0.4413 - acc: 0.9231 - val_
```

```
Epoch 00023: val_acc did not improve from 0.86300
Epoch 24/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.4354 - acc: 0.9239 - val_

Epoch 00024: val_acc did not improve from 0.86300
Epoch 25/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.4333 - acc: 0.9252 - val_

Epoch 00025: val_acc did not improve from 0.86300
Epoch 26/50
Learning rate:  0.001
1000/1000 [==============================] - 206s 206ms/step - loss: 0.4228 - acc: 0.9288 - val_

Epoch 00026: val_acc did not improve from 0.86300
Epoch 27/50
Learning rate:  0.001
 145/1000 [===>...] - ETA: 2:52 - loss: 0.4307 - acc: 0.9241
```

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

    KeyboardInterrupt                         Traceback (most recent call last)

    <ipython-input-16-2fc70d9ece25> in <module>()
    411                         validation_data=(x_test, y_test),
    412                         epochs=epochs, verbose=1, workers=4,steps_per_epoch=1000,
--> 413                         callbacks=callbacks)
    414
    415 # Score trained model.


    /usr/local/lib/python3.6/dist-packages/keras/legacy/interfaces.py in wrapper(*args, **kw
     89                 warnings.warn('Update your `' + object_name + '` call to the ' +
     90                               'Keras 2 API: ' + signature, stacklevel=2)
---> 91             return func(*args, **kwargs)
     92         wrapper._original_function = func
     93         return wrapper


    /usr/local/lib/python3.6/dist-packages/keras/engine/training.py in fit_generator(self, g
   1416             use_multiprocessing=use_multiprocessing,
   1417             shuffle=shuffle,
-> 1418             initial_epoch=initial_epoch)
   1419
   1420     @interfaces.legacy_generator_methods_support
```

```
/usr/local/lib/python3.6/dist-packages/keras/engine/training_generator.py in fit_generat
    215                 outs = model.train_on_batch(x, y,
    216                                             sample_weight=sample_weight,
--> 217                                             class_weight=class_weight)
    218
    219                 outs = to_list(outs)


/usr/local/lib/python3.6/dist-packages/keras/engine/training.py in train_on_batch(self,
   1215             ins = x + y + sample_weights
   1216         self._make_train_function()
-> 1217         outputs = self.train_function(ins)
   1218         return unpack_singleton(outputs)
   1219


/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py in __call__(s
   2713                 return self._legacy_call(inputs)
   2714
-> 2715             return self._call(inputs)
   2716         else:
   2717             if py_any(is_tensor(x) for x in inputs):


/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py in _call(self
   2673             fetched = self._callable_fn(*array_vals, run_metadata=self.run_metadata)
   2674         else:
-> 2675             fetched = self._callable_fn(*array_vals)
   2676         return fetched[:len(self.outputs)]
   2677


/usr/local/lib/python3.6/dist-packages/tensorflow/python/client/session.py in __call__(s
   1437           ret = tf_session.TF_SessionRunCallable(
   1438               self._session._session, self._handle, args, status,
-> 1439               run_metadata_ptr)
   1440         if run_metadata:
   1441           proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)


KeyboardInterrupt:
```

1.2.1 Results:

The original implementation of ResNetV2 for CIFAR obtained **0.863** for the validation accuracy at the 19 epoch out of the 27 epochs it was executed. We decided to stop the training/validation of the model at that number, because we were not appreciating any improvement in the accuracy.

1.3 Customized ResNetv2 implementation for CIFAR

In [0]:
```
import time
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers import Dropout
from keras.layers import AveragePooling2D, Input, Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau
from keras.regularizers import l2
from keras.preprocessing.image import ImageDataGenerator
from keras.layers.convolutional import MaxPooling2D
from keras.layers import Dense, Conv2D, BatchNormalization, Activation
from keras.utils import np_utils
# from keras_sequential_ascii import sequential_model_to_ascii_printout
from keras import backend as K
# if K.backend()=='tensorflow':
#     K.set_image_dim_ordering("th")
# Import Tensorflow with multiprocessing
import tensorflow as tf
import multiprocessing as mp

# Loading the CIFAR-10 datasets
from keras.datasets import cifar10

import seaborn as sn
import pandas  as pd

from sklearn.metrics import classification_report, confusion_matrix

# Training parameters
batch_size = 128  # orig paper trained all networks with batch_size=128
epochs = 100
data_augmentation = True
num_classes = 10
n = 3
depth = n * 9 + 2
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

#------------ Preprocessing ------------
# Input image dimensions.
input_shape = x_train.shape[1:]

# Normalize data.
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# Substract mean
x_train_mean = np.mean(x_train, axis=0)
x_train -= x_train_mean
x_test -= x_train_mean

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
print('y_train shape:', y_train.shape)

# Convert class vectors to binary class matrices.
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# ----- Callbacks functions -----
def lr_schedule(epoch):
    """Learning Rate Schedule
    Learning rate is scheduled to be reduced after 80, 120, 160, 180 epochs.
    Called automatically every epoch as part of callbacks during training.
    # Arguments
        epoch (int): The number of epochs
    # Returns
        lr (float32): learning rate
    """
    lr = 1e-3
    if epoch > 180:
        lr *= 0.5e-3
    elif epoch > 160:
        lr *= 1e-3
    elif epoch > 120:
        lr *= 1e-2
    elif epoch > 80:
        lr *= 1e-1
    print('Learning rate: ', lr)
    return lr

# # Prepare callbacks for model saving and for learning rate adjustment.
# checkpoint = ModelCheckpoint(filepath=filepath,
```

```python
#                               monitor='val_acc',
#                               verbose=1,
#                               save_best_only=True)

lr_scheduler = LearningRateScheduler(lr_schedule)

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               min_lr=0.5e-6)

callbacks = [lr_reducer, lr_scheduler]

# Create the model
def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):
    """2D Convolution-Batch Normalization-Activation stack builder
    # Arguments
        inputs (tensor): input tensor from input image or previous layer
        num_filters (int): Conv2D number of filters
        kernel_size (int): Conv2D square kernel dimensions
        strides (int): Conv2D square stride dimensions
        activation (string): activation name
        batch_normalization (bool): whether to include batch normalization
        conv_first (bool): conv-bn-activation (True) or
            bn-activation-conv (False)
    # Returns
        x (tensor): tensor as input to the next layer
    """
    conv = Conv2D(num_filters,
                  kernel_size=kernel_size,
                  strides=strides,
                  padding='same',
                  kernel_initializer='he_normal',
                  kernel_regularizer=l2(1e-4))

    x = inputs
    if conv_first:
        x = conv(x)
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
```

```python
    else:
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
        x = conv(x)
    return x


num_filters_in = 16
num_res_blocks = int((depth - 2) / 9)

inputs = Input(shape=input_shape)

x = resnet_layer(inputs=inputs,
                 num_filters=num_filters_in,
                 conv_first=True)

for stage in range(3):
    for res_block in range(num_res_blocks):
        activation = 'relu'
        batch_normalization = True
        strides = 1
        if stage == 0:
            num_filters_out = num_filters_in * 4
            if res_block == 0:  # first layer and first stage
                activation = None
                batch_normalization = False
        else:
            num_filters_out = num_filters_in * 2
            if res_block == 0:  # first layer but not first stage
                strides = 2    # downsample

        # bottleneck residual unit
        y = resnet_layer(inputs=x,
                         num_filters=num_filters_in,
                         kernel_size=1,
                         strides=strides,
                         activation=activation,
                         batch_normalization=batch_normalization,
                         conv_first=False)
        y = Dropout(0.2)(y)

        y = resnet_layer(inputs=y,
                         num_filters=num_filters_in,
                         conv_first=False)
        y = Dropout(0.2)(y)
```

```python
        y = resnet_layer(inputs=y,
                         num_filters=num_filters_out,
                         kernel_size=1,
                         conv_first=False)
        y = Dropout(0.2)(y)

        if res_block == 0:
            # linear projection residual shortcut connection to match
            # changed dims
            x = resnet_layer(inputs=x,
                             num_filters=num_filters_out,
                             kernel_size=1,
                             strides=strides,
                             activation=None,
                             batch_normalization=False)

        x = keras.layers.add([x, y])

    num_filters_in = num_filters_out
# Customizing last layers of the ResNet Model
# Add classifier on top
x = BatchNormalization()(x)
x = Activation('relu')(x)
# x = AveragePooling2D(pool_size=8)(x)
x = MaxPooling2D(pool_size=8)(x)
y = Flatten()(x)
y = Dense(1024, activation='relu', kernel_constraint=maxnorm(3))(y)
y = Dropout(0.2)(y)
y = Dense(512, activation='relu', kernel_constraint=maxnorm(3))(y)
y = Dropout(0.2)(y)

outputs = Dense(num_classes,
                activation ='softmax',
                kernel_initializer='he_normal')(y)

model = Model(inputs=inputs, outputs=outputs)

model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=lr_schedule(0)),
              metrics=['accuracy'])
model.summary()

# Run training, with or without data augmentation.
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
```

```
              validation_data=(x_test, y_test),
              shuffle=True,
              callbacks=callbacks)

else:
    print('Using real-time data augmentation.')
    # This will do preprocessing and realtime data augmentation:
    datagen = ImageDataGenerator(
    # set input mean to 0 over the dataset
    featurewise_center=False,
    # set each sample mean to 0
    samplewise_center=False,
    # divide inputs by std of dataset
    featurewise_std_normalization=False,
    # divide each input by its std
    samplewise_std_normalization=False,
    # apply ZCA whitening
    zca_whitening=False,
    # epsilon for ZCA whitening
    zca_epsilon=1e-06,
    # randomly rotate images in the range (deg 0 to 180)
    rotation_range=0,
    # randomly shift images horizontally
    width_shift_range=0.1,
    # randomly shift images vertically
    height_shift_range=0.1,
    # set range for random shear
    shear_range=0.,
    # set range for random zoom
    zoom_range=0.,
    # set range for random channel shifts
    channel_shift_range=0.,
    # set mode for filling points outside the input boundaries
    fill_mode='nearest',
    # value used for fill_mode = "constant"
    cval=0.,
    # randomly flip images
    horizontal_flip=True,
    # randomly flip images
    vertical_flip=False,
    # set rescaling factor (applied before any other transformation)
    rescale=None,
    # set function that will be applied on each input
    preprocessing_function=None,
    # image data format, either "channels_first" or "channels_last"
    data_format=None,
    # fraction of images reserved for validation (strictly between 0 and 1)
    validation_split=0.0)
```

```
            # Compute quantities required for featurewise normalization
            # (std, mean, and principal components if ZCA whitening is applied).
            datagen.fit(x_train)

            # Fit the model on the batches generated by datagen.flow().
            model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_size),
                                validation_data=(x_test, y_test),
                                epochs=epochs, verbose=1, workers=4,steps_per_epoch=1000,
                                callbacks=callbacks)

        # Final evaluation of the model
        scores = model.evaluate(X_test, y_test, verbose=0)
        print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
x_train shape: (50000, 3, 32, 32)
50000 train samples
10000 test samples
y_train shape: (50000, 1)
Learning rate:  0.001
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_7 (InputLayer)             (None, 3, 32, 32)     0
____________________________________________________________________________________________________
conv2d_177 (Conv2D)              (None, 16, 32, 32)    448         input_7[0][0]
____________________________________________________________________________________________________
batch_normalization_151 (BatchN (None, 16, 32, 32)    128         conv2d_177[0][0]
____________________________________________________________________________________________________
activation_151 (Activation)      (None, 16, 32, 32)    0           batch_normalization_151[0][0]
____________________________________________________________________________________________________
conv2d_178 (Conv2D)              (None, 16, 32, 32)    272         activation_151[0][0]
____________________________________________________________________________________________________
dropout_9 (Dropout)              (None, 16, 32, 32)    0           conv2d_178[0][0]
____________________________________________________________________________________________________
batch_normalization_152 (BatchN (None, 16, 32, 32)    128         dropout_9[0][0]
____________________________________________________________________________________________________
activation_152 (Activation)      (None, 16, 32, 32)    0           batch_normalization_152[0][0]
____________________________________________________________________________________________________
conv2d_179 (Conv2D)              (None, 16, 32, 32)    2320        activation_152[0][0]
____________________________________________________________________________________________________
dropout_10 (Dropout)             (None, 16, 32, 32)    0           conv2d_179[0][0]
____________________________________________________________________________________________________
batch_normalization_153 (BatchN (None, 16, 32, 32)    128         dropout_10[0][0]
____________________________________________________________________________________________________
activation_153 (Activation)      (None, 16, 32, 32)    0           batch_normalization_153[0][0]
____________________________________________________________________________________________________
conv2d_180 (Conv2D)              (None, 64, 32, 32)    1088        activation_153[0][0]
____________________________________________________________________________________________________
```

```
conv2d_181 (Conv2D)             (None, 64, 32, 32)   1088        activation_151[0][0]
__________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 64, 32, 32)   0           conv2d_180[0][0]
__________________________________________________________________________________________________
add_49 (Add)                    (None, 64, 32, 32)   0           conv2d_181[0][0]
                                                                 dropout_11[0][0]
__________________________________________________________________________________________________
batch_normalization_154 (BatchN (None, 64, 32, 32)   128         add_49[0][0]
__________________________________________________________________________________________________
activation_154 (Activation)     (None, 64, 32, 32)   0           batch_normalization_154[0][0]
__________________________________________________________________________________________________
conv2d_182 (Conv2D)             (None, 16, 32, 32)   1040        activation_154[0][0]
__________________________________________________________________________________________________
dropout_12 (Dropout)            (None, 16, 32, 32)   0           conv2d_182[0][0]
__________________________________________________________________________________________________
batch_normalization_155 (BatchN (None, 16, 32, 32)   128         dropout_12[0][0]
__________________________________________________________________________________________________
activation_155 (Activation)     (None, 16, 32, 32)   0           batch_normalization_155[0][0]
__________________________________________________________________________________________________
conv2d_183 (Conv2D)             (None, 16, 32, 32)   2320        activation_155[0][0]
__________________________________________________________________________________________________
dropout_13 (Dropout)            (None, 16, 32, 32)   0           conv2d_183[0][0]
__________________________________________________________________________________________________
batch_normalization_156 (BatchN (None, 16, 32, 32)   128         dropout_13[0][0]
__________________________________________________________________________________________________
activation_156 (Activation)     (None, 16, 32, 32)   0           batch_normalization_156[0][0]
__________________________________________________________________________________________________
conv2d_184 (Conv2D)             (None, 64, 32, 32)   1088        activation_156[0][0]
__________________________________________________________________________________________________
dropout_14 (Dropout)            (None, 64, 32, 32)   0           conv2d_184[0][0]
__________________________________________________________________________________________________
add_50 (Add)                    (None, 64, 32, 32)   0           add_49[0][0]
                                                                 dropout_14[0][0]
__________________________________________________________________________________________________
batch_normalization_157 (BatchN (None, 64, 32, 32)   128         add_50[0][0]
__________________________________________________________________________________________________
activation_157 (Activation)     (None, 64, 32, 32)   0           batch_normalization_157[0][0]
__________________________________________________________________________________________________
conv2d_185 (Conv2D)             (None, 16, 32, 32)   1040        activation_157[0][0]
__________________________________________________________________________________________________
dropout_15 (Dropout)            (None, 16, 32, 32)   0           conv2d_185[0][0]
__________________________________________________________________________________________________
batch_normalization_158 (BatchN (None, 16, 32, 32)   128         dropout_15[0][0]
__________________________________________________________________________________________________
activation_158 (Activation)     (None, 16, 32, 32)   0           batch_normalization_158[0][0]
__________________________________________________________________________________________________
conv2d_186 (Conv2D)             (None, 16, 32, 32)   2320        activation_158[0][0]
__________________________________________________________________________________________________
```

```
dropout_16 (Dropout)            (None, 16, 32, 32)   0           conv2d_186[0][0]
__________________________________________________________________________________________________
batch_normalization_159 (BatchN (None, 16, 32, 32)   128         dropout_16[0][0]
__________________________________________________________________________________________________
activation_159 (Activation)     (None, 16, 32, 32)   0           batch_normalization_159[0][0]
__________________________________________________________________________________________________
conv2d_187 (Conv2D)             (None, 64, 32, 32)   1088        activation_159[0][0]
__________________________________________________________________________________________________
dropout_17 (Dropout)            (None, 64, 32, 32)   0           conv2d_187[0][0]
__________________________________________________________________________________________________
add_51 (Add)                    (None, 64, 32, 32)   0           add_50[0][0]
                                                                 dropout_17[0][0]
__________________________________________________________________________________________________
batch_normalization_160 (BatchN (None, 64, 32, 32)   128         add_51[0][0]
__________________________________________________________________________________________________
activation_160 (Activation)     (None, 64, 32, 32)   0           batch_normalization_160[0][0]
__________________________________________________________________________________________________
conv2d_188 (Conv2D)             (None, 64, 16, 16)   4160        activation_160[0][0]
__________________________________________________________________________________________________
dropout_18 (Dropout)            (None, 64, 16, 16)   0           conv2d_188[0][0]
__________________________________________________________________________________________________
batch_normalization_161 (BatchN (None, 64, 16, 16)   64          dropout_18[0][0]
__________________________________________________________________________________________________
activation_161 (Activation)     (None, 64, 16, 16)   0           batch_normalization_161[0][0]
__________________________________________________________________________________________________
conv2d_189 (Conv2D)             (None, 64, 16, 16)   36928       activation_161[0][0]
__________________________________________________________________________________________________
dropout_19 (Dropout)            (None, 64, 16, 16)   0           conv2d_189[0][0]
__________________________________________________________________________________________________
batch_normalization_162 (BatchN (None, 64, 16, 16)   64          dropout_19[0][0]
__________________________________________________________________________________________________
activation_162 (Activation)     (None, 64, 16, 16)   0           batch_normalization_162[0][0]
__________________________________________________________________________________________________
conv2d_190 (Conv2D)             (None, 128, 16, 16)  8320        activation_162[0][0]
__________________________________________________________________________________________________
conv2d_191 (Conv2D)             (None, 128, 16, 16)  8320        add_51[0][0]
__________________________________________________________________________________________________
dropout_20 (Dropout)            (None, 128, 16, 16)  0           conv2d_190[0][0]
__________________________________________________________________________________________________
add_52 (Add)                    (None, 128, 16, 16)  0           conv2d_191[0][0]
                                                                 dropout_20[0][0]
__________________________________________________________________________________________________
batch_normalization_163 (BatchN (None, 128, 16, 16)  64          add_52[0][0]
__________________________________________________________________________________________________
activation_163 (Activation)     (None, 128, 16, 16)  0           batch_normalization_163[0][0]
__________________________________________________________________________________________________
conv2d_192 (Conv2D)             (None, 64, 16, 16)   8256        activation_163[0][0]
__________________________________________________________________________________________________
```

```
dropout_21 (Dropout)            (None, 64, 16, 16)   0           conv2d_192[0][0]
__________________________________________________________________________________________________
batch_normalization_164 (BatchN (None, 64, 16, 16)   64          dropout_21[0][0]
__________________________________________________________________________________________________
activation_164 (Activation)     (None, 64, 16, 16)   0           batch_normalization_164[0][0]
__________________________________________________________________________________________________
conv2d_193 (Conv2D)             (None, 64, 16, 16)   36928       activation_164[0][0]
__________________________________________________________________________________________________
dropout_22 (Dropout)            (None, 64, 16, 16)   0           conv2d_193[0][0]
__________________________________________________________________________________________________
batch_normalization_165 (BatchN (None, 64, 16, 16)   64          dropout_22[0][0]
__________________________________________________________________________________________________
activation_165 (Activation)     (None, 64, 16, 16)   0           batch_normalization_165[0][0]
__________________________________________________________________________________________________
conv2d_194 (Conv2D)             (None, 128, 16, 16)  8320        activation_165[0][0]
__________________________________________________________________________________________________
dropout_23 (Dropout)            (None, 128, 16, 16)  0           conv2d_194[0][0]
__________________________________________________________________________________________________
add_53 (Add)                    (None, 128, 16, 16)  0           add_52[0][0]
                                                                 dropout_23[0][0]
__________________________________________________________________________________________________
batch_normalization_166 (BatchN (None, 128, 16, 16)  64          add_53[0][0]
__________________________________________________________________________________________________
activation_166 (Activation)     (None, 128, 16, 16)  0           batch_normalization_166[0][0]
__________________________________________________________________________________________________
conv2d_195 (Conv2D)             (None, 64, 16, 16)   8256        activation_166[0][0]
__________________________________________________________________________________________________
dropout_24 (Dropout)            (None, 64, 16, 16)   0           conv2d_195[0][0]
__________________________________________________________________________________________________
batch_normalization_167 (BatchN (None, 64, 16, 16)   64          dropout_24[0][0]
__________________________________________________________________________________________________
activation_167 (Activation)     (None, 64, 16, 16)   0           batch_normalization_167[0][0]
__________________________________________________________________________________________________
conv2d_196 (Conv2D)             (None, 64, 16, 16)   36928       activation_167[0][0]
__________________________________________________________________________________________________
dropout_25 (Dropout)            (None, 64, 16, 16)   0           conv2d_196[0][0]
__________________________________________________________________________________________________
batch_normalization_168 (BatchN (None, 64, 16, 16)   64          dropout_25[0][0]
__________________________________________________________________________________________________
activation_168 (Activation)     (None, 64, 16, 16)   0           batch_normalization_168[0][0]
__________________________________________________________________________________________________
conv2d_197 (Conv2D)             (None, 128, 16, 16)  8320        activation_168[0][0]
__________________________________________________________________________________________________
dropout_26 (Dropout)            (None, 128, 16, 16)  0           conv2d_197[0][0]
__________________________________________________________________________________________________
add_54 (Add)                    (None, 128, 16, 16)  0           add_53[0][0]
                                                                 dropout_26[0][0]
__________________________________________________________________________________________________
```

```
batch_normalization_169 (BatchN (None, 128, 16, 16)  64          add_54[0][0]
__________________________________________________________________________________________________
activation_169 (Activation)     (None, 128, 16, 16)  0           batch_normalization_169[0][0]
__________________________________________________________________________________________________
conv2d_198 (Conv2D)             (None, 128, 8, 8)    16512       activation_169[0][0]
__________________________________________________________________________________________________
dropout_27 (Dropout)            (None, 128, 8, 8)    0           conv2d_198[0][0]
__________________________________________________________________________________________________
batch_normalization_170 (BatchN (None, 128, 8, 8)    32          dropout_27[0][0]
__________________________________________________________________________________________________
activation_170 (Activation)     (None, 128, 8, 8)    0           batch_normalization_170[0][0]
__________________________________________________________________________________________________
conv2d_199 (Conv2D)             (None, 128, 8, 8)    147584      activation_170[0][0]
__________________________________________________________________________________________________
dropout_28 (Dropout)            (None, 128, 8, 8)    0           conv2d_199[0][0]
__________________________________________________________________________________________________
batch_normalization_171 (BatchN (None, 128, 8, 8)    32          dropout_28[0][0]
__________________________________________________________________________________________________
activation_171 (Activation)     (None, 128, 8, 8)    0           batch_normalization_171[0][0]
__________________________________________________________________________________________________
conv2d_200 (Conv2D)             (None, 256, 8, 8)    33024       activation_171[0][0]
__________________________________________________________________________________________________
conv2d_201 (Conv2D)             (None, 256, 8, 8)    33024       add_54[0][0]
__________________________________________________________________________________________________
dropout_29 (Dropout)            (None, 256, 8, 8)    0           conv2d_200[0][0]
__________________________________________________________________________________________________
add_55 (Add)                    (None, 256, 8, 8)    0           conv2d_201[0][0]
                                                                 dropout_29[0][0]
__________________________________________________________________________________________________
batch_normalization_172 (BatchN (None, 256, 8, 8)    32          add_55[0][0]
__________________________________________________________________________________________________
activation_172 (Activation)     (None, 256, 8, 8)    0           batch_normalization_172[0][0]
__________________________________________________________________________________________________
conv2d_202 (Conv2D)             (None, 128, 8, 8)    32896       activation_172[0][0]
__________________________________________________________________________________________________
dropout_30 (Dropout)            (None, 128, 8, 8)    0           conv2d_202[0][0]
__________________________________________________________________________________________________
batch_normalization_173 (BatchN (None, 128, 8, 8)    32          dropout_30[0][0]
__________________________________________________________________________________________________
activation_173 (Activation)     (None, 128, 8, 8)    0           batch_normalization_173[0][0]
__________________________________________________________________________________________________
conv2d_203 (Conv2D)             (None, 128, 8, 8)    147584      activation_173[0][0]
__________________________________________________________________________________________________
dropout_31 (Dropout)            (None, 128, 8, 8)    0           conv2d_203[0][0]
__________________________________________________________________________________________________
batch_normalization_174 (BatchN (None, 128, 8, 8)    32          dropout_31[0][0]
__________________________________________________________________________________________________
activation_174 (Activation)     (None, 128, 8, 8)    0           batch_normalization_174[0][0]
```

```
____________________________________________________________________________________________________
conv2d_204 (Conv2D)             (None, 256, 8, 8)    33024       activation_174[0][0]
____________________________________________________________________________________________________
dropout_32 (Dropout)            (None, 256, 8, 8)    0           conv2d_204[0][0]
____________________________________________________________________________________________________
add_56 (Add)                    (None, 256, 8, 8)    0           add_55[0][0]
                                                                 dropout_32[0][0]
____________________________________________________________________________________________________
batch_normalization_175 (BatchN (None, 256, 8, 8)    32          add_56[0][0]
____________________________________________________________________________________________________
activation_175 (Activation)     (None, 256, 8, 8)    0           batch_normalization_175[0][0]
____________________________________________________________________________________________________
conv2d_205 (Conv2D)             (None, 128, 8, 8)    32896       activation_175[0][0]
____________________________________________________________________________________________________
dropout_33 (Dropout)            (None, 128, 8, 8)    0           conv2d_205[0][0]
____________________________________________________________________________________________________
batch_normalization_176 (BatchN (None, 128, 8, 8)    32          dropout_33[0][0]
____________________________________________________________________________________________________
activation_176 (Activation)     (None, 128, 8, 8)    0           batch_normalization_176[0][0]
____________________________________________________________________________________________________
conv2d_206 (Conv2D)             (None, 128, 8, 8)    147584      activation_176[0][0]
____________________________________________________________________________________________________
dropout_34 (Dropout)            (None, 128, 8, 8)    0           conv2d_206[0][0]
____________________________________________________________________________________________________
batch_normalization_177 (BatchN (None, 128, 8, 8)    32          dropout_34[0][0]
____________________________________________________________________________________________________
activation_177 (Activation)     (None, 128, 8, 8)    0           batch_normalization_177[0][0]
____________________________________________________________________________________________________
conv2d_207 (Conv2D)             (None, 256, 8, 8)    33024       activation_177[0][0]
____________________________________________________________________________________________________
dropout_35 (Dropout)            (None, 256, 8, 8)    0           conv2d_207[0][0]
____________________________________________________________________________________________________
add_57 (Add)                    (None, 256, 8, 8)    0           add_56[0][0]
                                                                 dropout_35[0][0]
____________________________________________________________________________________________________
batch_normalization_178 (BatchN (None, 256, 8, 8)    32          add_57[0][0]
____________________________________________________________________________________________________
activation_178 (Activation)     (None, 256, 8, 8)    0           batch_normalization_178[0][0]
____________________________________________________________________________________________________
max_pooling2d_5 (MaxPooling2D)  (None, 256, 1, 1)    0           activation_178[0][0]
____________________________________________________________________________________________________
flatten_8 (Flatten)             (None, 256)          0           max_pooling2d_5[0][0]
____________________________________________________________________________________________________
dense_8 (Dense)                 (None, 1024)         263168      flatten_8[0][0]
____________________________________________________________________________________________________
dropout_36 (Dropout)            (None, 1024)         0           dense_8[0][0]
____________________________________________________________________________________________________
dense_9 (Dense)                 (None, 512)          524800      dropout_36[0][0]
```

```
--------------------------------------------------------------------------------------------------------
dropout_37 (Dropout)            (None, 512)          0           dense_9[0][0]
--------------------------------------------------------------------------------------------------------
dense_10 (Dense)                (None, 10)           5130        dropout_37[0][0]
========================================================================================================
Total params: 1,631,242
Trainable params: 1,630,170
Non-trainable params: 1,072
--------------------------------------------------------------------------------------------------------
Using real-time data augmentation.
Epoch 1/100
Learning rate:  0.001
1000/1000 [==============================] - 336s 336ms/step - loss: 2.1289 - acc: 0.3178 - val_
Epoch 2/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 1.5934 - acc: 0.4890 - val_
Epoch 3/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 1.3653 - acc: 0.5788 - val_
Epoch 4/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 1.2336 - acc: 0.6293 - val_
Epoch 5/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 1.1421 - acc: 0.6629 - val_
Epoch 6/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 1.0715 - acc: 0.6902 - val_
Epoch 7/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 1.0253 - acc: 0.7090 - val_
Epoch 8/100
Learning rate:  0.001
1000/1000 [==============================] - 317s 317ms/step - loss: 0.9716 - acc: 0.7282 - val_
Epoch 9/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.9310 - acc: 0.7435 - val_
Epoch 10/100
Learning rate:  0.001
1000/1000 [==============================] - 317s 317ms/step - loss: 0.8991 - acc: 0.7570 - val_
Epoch 11/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.8690 - acc: 0.7673 - val_
Epoch 12/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.8423 - acc: 0.7779 - val_
Epoch 13/100
Learning rate:  0.001
```

```
1000/1000 [==============================] - 316s 316ms/step - loss: 0.8247 - acc: 0.7843 - val_
Epoch 14/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.8071 - acc: 0.7907 - val_
Epoch 15/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7902 - acc: 0.7966 - val_
Epoch 16/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7768 - acc: 0.8015 - val_
Epoch 17/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7651 - acc: 0.8066 - val_
Epoch 18/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7568 - acc: 0.8089 - val_
Epoch 19/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7447 - acc: 0.8137 - val_
Epoch 20/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7363 - acc: 0.8176 - val_
Epoch 21/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7275 - acc: 0.8205 - val_
Epoch 22/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7205 - acc: 0.8228 - val_
Epoch 23/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7162 - acc: 0.8244 - val_
Epoch 24/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.7111 - acc: 0.8263 - val_
Epoch 25/100
Learning rate:  0.001
1000/1000 [==============================] - 315s 315ms/step - loss: 0.7067 - acc: 0.8276 - val_
Epoch 26/100
Learning rate:  0.001
1000/1000 [==============================] - 314s 314ms/step - loss: 0.6935 - acc: 0.8329 - val_
Epoch 27/100
Learning rate:  0.001
1000/1000 [==============================] - 315s 315ms/step - loss: 0.6893 - acc: 0.8335 - val_
Epoch 28/100
Learning rate:  0.001
1000/1000 [==============================] - 315s 315ms/step - loss: 0.6863 - acc: 0.8355 - val_
Epoch 29/100
Learning rate:  0.001
```

```
1000/1000 [==============================] - 317s 317ms/step - loss: 0.6802 - acc: 0.8359 - val_
Epoch 30/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.6757 - acc: 0.8386 - val_
Epoch 31/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.6772 - acc: 0.8380 - val_
Epoch 32/100
Learning rate:  0.001
1000/1000 [==============================] - 316s 316ms/step - loss: 0.6656 - acc: 0.8407 - val_
Epoch 33/100
Learning rate:  0.001
 690/1000 [===================>...] - ETA: 1:35 - loss: 0.6633 - acc: 0.8456
```

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

        KeyboardInterrupt                         Traceback (most recent call last)

        <ipython-input-23-34e4b60880fa> in <module>()
        302                             validation_data=(x_test, y_test),
        303                             epochs=epochs, verbose=1, workers=4,steps_per_epoch=1000,
    --> 304                             callbacks=callbacks)
        305
        306 # Final evaluation of the model


        /usr/local/lib/python3.6/dist-packages/keras/legacy/interfaces.py in wrapper(*args, **kw
         89                 warnings.warn('Update your `' + object_name + '` call to the ' +
         90                               'Keras 2 API: ' + signature, stacklevel=2)
    ---> 91             return func(*args, **kwargs)
         92         wrapper._original_function = func
         93         return wrapper


        /usr/local/lib/python3.6/dist-packages/keras/engine/training.py in fit_generator(self, g
       1416             use_multiprocessing=use_multiprocessing,
       1417             shuffle=shuffle,
    -> 1418             initial_epoch=initial_epoch)
       1419
       1420     @interfaces.legacy_generator_methods_support


        /usr/local/lib/python3.6/dist-packages/keras/engine/training_generator.py in fit_generat
        215                 outs = model.train_on_batch(x, y,
        216                                             sample_weight=sample_weight,
    --> 217                                             class_weight=class_weight)
        218
```

```
    219                 outs = to_list(outs)


    /usr/local/lib/python3.6/dist-packages/keras/engine/training.py in train_on_batch(self,
   1215             ins = x + y + sample_weights
   1216         self._make_train_function()
-> 1217         outputs = self.train_function(ins)
   1218         return unpack_singleton(outputs)
   1219


    /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py in __call__(s
   2713                 return self._legacy_call(inputs)
   2714
-> 2715             return self._call(inputs)
   2716         else:
   2717             if py_any(is_tensor(x) for x in inputs):


    /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py in _call(self
   2673             fetched = self._callable_fn(*array_vals, run_metadata=self.run_metadata)
   2674         else:
-> 2675             fetched = self._callable_fn(*array_vals)
   2676         return fetched[:len(self.outputs)]
   2677


    /usr/local/lib/python3.6/dist-packages/tensorflow/python/client/session.py in __call__(s
   1437           ret = tf_session.TF_SessionRunCallable(
   1438               self._session._session, self._handle, args, status,
-> 1439               run_metadata_ptr)
   1440         if run_metadata:
   1441           proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)


    KeyboardInterrupt:
```

1.3.1 Results:

We decided to customize ResNetV2 by adding 2 fully-connected layers combined with a dropout layer. Unfortunately, this customization did not add any benefit to the original implementation. In fact, we found that the performance was worse than the original one. At the 19 epoch the custom model had a validation accuracy of **0.833** while the original one had **0.863** validation accuracy.

3. Evaluate your module using the FERPlus dataset (The model with the best test accuracy will present their solution to the class).

   3.1 Download the FER2013 dataset (images_path).

3.2 Download the FERPlus labels (labels_path).

3.3 Use the following code snippet to load the dataset giving the appropiate paths to the csv files downloaded in 3.1 and 3.2:

1.4 FERPlus dataset collection

```
In [0]: class FERPlus(object):
            """Class for loading FER2013 [1] emotion classification dataset with
            the FERPlus labels [2]:
            [1] kaggle.com/c/challenges-in-representation-learning-facial-\
                    expression-recognition-challenge
            [2] github.com/Microsoft/FERPlu://github.com/Microsoft/FERPlus"""

            def __init__(self, images_path, labels_path, split='train', image_size=(48, 48),
                         dataset_name='FERPlus'):

                self.split = split
                self.image_size = image_size
                self.dataset_name = dataset_name
                self.images_path = images_path
                self.labels_path = labels_path
                self.class_names = ['neutral', 'happiness', 'surprise', 'sadness',
                                    'anger', 'disgust', 'fear', 'contempt']
                self.num_classes = len(self.class_names)
                self.arg_to_name = dict(zip(range(self.num_classes), self.class_names))
                self.name_to_arg = dict(zip(self.class_names, range(self.num_classes)))
                self._split_to_filter = {
                    'train': 'Training', 'val': 'PublicTest', 'test': 'PrivateTest'}

            def load_data(self):
                filter_name = self._split_to_filter[self.split]
                pixel_sequences = pd.read_csv(self.images_path)
                pixel_sequences = pixel_sequences[pixel_sequences.Usage == filter_name]
                pixel_sequences = pixel_sequences['pixels'].tolist()
                faces = []
                for pixel_sequence in pixel_sequences:
                    face = [float(pixel) for pixel in pixel_sequence.split(' ')]
                    face = np.asarray(face).reshape(48, 48)
                    faces.append(cv2.resize(face, self.image_size))
                faces = np.asarray(faces)
                faces = np.expand_dims(faces, -1)

                emotions = pd.read_csv(self.labels_path)
                emotions = emotions[emotions.Usage == filter_name]
                emotions = emotions.iloc[:, 2:10].values
                N = np.sum(emotions, axis=1)
                mask = N != 0
                N, faces, emotions = N[mask], faces[mask], emotions[mask]
```

```
        emotions = emotions / np.expand_dims(N, 1)
        return faces, emotions
```

1.5 Customized ResNetv2 implementation for FERPlus dataset

```
In [0]: import time
        import matplotlib.pyplot as plt
        import numpy as np
        from keras.models import Sequential
        from keras.layers import Dropout
        from keras.layers import AveragePooling2D, Input, Flatten
        from keras.layers import Convolution2D
        from keras.constraints import maxnorm
        from keras.optimizers import SGD
        from keras.optimizers import Adam
        from keras.callbacks import ModelCheckpoint, LearningRateScheduler
        from keras.callbacks import ReduceLROnPlateau
        from keras.regularizers import l2
        from keras.preprocessing.image import ImageDataGenerator
        from keras.layers.convolutional import MaxPooling2D
        from keras.layers import GlobalAveragePooling2D
        from keras.layers import Dense, Conv2D, BatchNormalization, Activation
        import keras.utils
        import cv2
        from keras.models import Model
        from keras import backend as K
        import tensorflow as tf
        import multiprocessing as mp
        import seaborn as sn
        import pandas  as pd
        from sklearn.metrics import classification_report, confusion_matrix

        # Training parameters
        batch_size = 128  # orig paper trained all networks with batch_size=128
        epochs = 100
        data_augmentation = True

        n = 3
        depth = n * 9 + 2

        FER_Data = FERPlus('drive/My Drive/Colab Notebooks/FERPlus_data/fer2013/fer2013.csv', \
                           'drive/My Drive/Colab Notebooks/FERPlus_data/FERPlus/fer2013new.csv',
        x_train, y_train = FER_Data.load_data()
        FER_Data = FERPlus('drive/My Drive/Colab Notebooks/FERPlus_data/fer2013/fer2013.csv', \
                           'drive/My Drive/Colab Notebooks/FERPlus_data/FERPlus/fer2013new.csv',
        x_test, y_test = FER_Data.load_data()
        print(x_train.shape)
        print(x_test.shape)
```

```
print(FER_Data.num_classes)
num_classes = FER_Data.num_classes

#------------ Preprocessing ------------
# Input image dimensions.
input_shape = x_train.shape[1:]

# Normalize data.
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# Substract mean
x_train_mean = np.mean(x_train, axis=0)
x_train -= x_train_mean
x_test -= x_train_mean

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
print('y_train shape:', y_train.shape)

# ----- Callbacks functions -----
def lr_schedule(epoch):
    """Learning Rate Schedule
    Learning rate is scheduled to be reduced after 80, 120, 160, 180 epochs.
    Called automatically every epoch as part of callbacks during training.
    # Arguments
        epoch (int): The number of epochs
    # Returns
        lr (float32): learning rate
    """
    lr = 1e-3
    if epoch > 180:
        lr *= 0.5e-3
    elif epoch > 160:
        lr *= 1e-3
    elif epoch > 120:
        lr *= 1e-2
    elif epoch > 80:
        lr *= 1e-1
    print('Learning rate: ', lr)
    return lr

lr_scheduler = LearningRateScheduler(lr_schedule)

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
```

```
                                       min_lr=0.5e-6)

callbacks = [lr_reducer, lr_scheduler]

# Create the model
def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):
    """2D Convolution-Batch Normalization-Activation stack builder
    # Arguments
        inputs (tensor): input tensor from input image or previous layer
        num_filters (int): Conv2D number of filters
        kernel_size (int): Conv2D square kernel dimensions
        strides (int): Conv2D square stride dimensions
        activation (string): activation name
        batch_normalization (bool): whether to include batch normalization
        conv_first (bool): conv-bn-activation (True) or
            bn-activation-conv (False)
    # Returns
        x (tensor): tensor as input to the next layer
    """
    conv = Conv2D(num_filters,
                  kernel_size=kernel_size,
                  strides=strides,
                  padding='same',
                  kernel_initializer='he_normal',
                  kernel_regularizer=l2(1e-4))

    x = inputs
    if conv_first:
        x = conv(x)
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
    else:
        if batch_normalization:
            x = BatchNormalization()(x)
        if activation is not None:
            x = Activation(activation)(x)
        x = conv(x)
    return x
```

```python
num_filters_in = 16
num_res_blocks = int((depth - 2) / 9)

inputs = Input(shape=input_shape)

x = resnet_layer(inputs=inputs,
                 num_filters=num_filters_in,
                 conv_first=True)

for stage in range(3):
    for res_block in range(num_res_blocks):
        activation = 'relu'
        batch_normalization = True
        strides = 1
        if stage == 0:
            num_filters_out = num_filters_in * 4
            if res_block == 0:  # first layer and first stage
                activation = None
                batch_normalization = False
        else:
            num_filters_out = num_filters_in * 2
            if res_block == 0:  # first layer but not first stage
                strides = 2    # downsample

        # bottleneck residual unit
        y = resnet_layer(inputs=x,
                         num_filters=num_filters_in,
                         kernel_size=1,
                         strides=strides,
                         activation=activation,
                         batch_normalization=batch_normalization,
                         conv_first=False)
        y = Dropout(0.2)(y)

        y = resnet_layer(inputs=y,
                         num_filters=num_filters_in,
                         conv_first=False)
        y = Dropout(0.2)(y)

        y = resnet_layer(inputs=y,
                         num_filters=num_filters_out,
                         kernel_size=1,
                         conv_first=False)
        y = Dropout(0.2)(y)

        if res_block == 0:
            # linear projection residual shortcut connection to match
            # changed dims
```

```
            x = resnet_layer(inputs=x,
                             num_filters=num_filters_out,
                             kernel_size=1,
                             strides=strides,
                             activation=None,
                             batch_normalization=False)

        x = keras.layers.add([x, y])

    num_filters_in = num_filters_out

# Add classifier on top
x = BatchNormalization()(x)
x = Activation('relu')(x)
# x = AveragePooling2D(pool_size=8)(x)
x = MaxPooling2D(pool_size=8)(x)
# y = Flatten()(x)
y = Dense(1024, activation='relu', kernel_constraint=maxnorm(3))(y)
y = Dropout(0.2)(y)
y = Dense(512, activation='relu', kernel_constraint=maxnorm(3))(y)
y = Dropout(0.2)(y)
y = Convolution2D(filters=256, kernel_size=(3, 3), padding='same') (y)
y = AveragePooling2D(pool_size=(2, 2), padding='same')(y)
y = Dropout(0.5)(y)
y = BatchNormalization() (y)
y = Convolution2D(filters=num_classes,
                  kernel_size=(3, 3),
                  padding='same')(y)
y = GlobalAveragePooling2D()(y)
outputs = Activation('softmax', name='predictions')(y)

# outputs = Dense(num_classes,
#                 activation ='softmax',
#                 kernel_initializer='he_normal')(y)

model = Model(inputs=inputs, outputs=outputs)

model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=lr_schedule(0)),
              metrics=['accuracy'])
model.summary()

# Run training, with or without data augmentation.
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
```

```
              validation_data=(x_test, y_test),
              shuffle=True,
              callbacks=callbacks)

else:
    print('Using real-time data augmentation.')
    # This will do preprocessing and realtime data augmentation:
    datagen = ImageDataGenerator(
    # set input mean to 0 over the dataset
    featurewise_center=False,
    # set each sample mean to 0
    samplewise_center=False,
    # divide inputs by std of dataset
    featurewise_std_normalization=False,
    # divide each input by its std
    samplewise_std_normalization=False,
    # apply ZCA whitening
    zca_whitening=False,
    # epsilon for ZCA whitening
    zca_epsilon=1e-06,
    # randomly rotate images in the range (deg 0 to 180)
    rotation_range=0,
    # randomly shift images horizontally
    width_shift_range=0.1,
    # randomly shift images vertically
    height_shift_range=0.1,
    # set range for random shear
    shear_range=0.,
    # set range for random zoom
    zoom_range=0.,
    # set range for random channel shifts
    channel_shift_range=0.,
    # set mode for filling points outside the input boundaries
    fill_mode='nearest',
    # value used for fill_mode = "constant"
    cval=0.,
    # randomly flip images
    horizontal_flip=True,
    # randomly flip images
    vertical_flip=False,
    # set rescaling factor (applied before any other transformation)
    rescale=None,
    # set function that will be applied on each input
    preprocessing_function=None,
    # image data format, either "channels_first" or "channels_last"
    data_format=None,
    # fraction of images reserved for validation (strictly between 0 and 1)
    validation_split=0.0)
```

```
        # Compute quantities required for featurewise normalization
        # (std, mean, and principal components if ZCA whitening is applied).
        datagen.fit(x_train)

        # Fit the model on the batches generated by datagen.flow().
        model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_size),
                            validation_data=(x_test, y_test),
                            epochs=epochs, verbose=1, workers=4,steps_per_epoch=1000,
                            callbacks=callbacks)

    # Final evaluation of the model
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
(28559, 48, 48, 1)
(3573, 48, 48, 1)
8
x_train shape: (28559, 48, 48, 1)
28559 train samples
3573 test samples
y_train shape: (28559, 8)
Learning rate:  0.001
________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================================
input_2 (InputLayer)            (None, 48, 48, 1)    0
________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 48, 48, 16)   160         input_2[0][0]
________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 48, 48, 16)   64          conv2d_34[0][0]
________________________________________________________________________________________________
activation_29 (Activation)      (None, 48, 48, 16)   0           batch_normalization_30[0][0]
________________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, 48, 48, 16)   272         activation_29[0][0]
________________________________________________________________________________________________
dropout_31 (Dropout)            (None, 48, 48, 16)   0           conv2d_35[0][0]
________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 48, 48, 16)   64          dropout_31[0][0]
________________________________________________________________________________________________
activation_30 (Activation)      (None, 48, 48, 16)   0           batch_normalization_31[0][0]
________________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, 48, 48, 16)   2320        activation_30[0][0]
________________________________________________________________________________________________
dropout_32 (Dropout)            (None, 48, 48, 16)   0           conv2d_36[0][0]
________________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 48, 48, 16)   64          dropout_32[0][0]
________________________________________________________________________________________________
activation_31 (Activation)      (None, 48, 48, 16)   0           batch_normalization_32[0][0]
```

```
__________________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, 48, 48, 64)   1088        activation_31[0][0]
__________________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, 48, 48, 64)   1088        activation_29[0][0]
__________________________________________________________________________________________________
dropout_33 (Dropout)            (None, 48, 48, 64)   0           conv2d_37[0][0]
__________________________________________________________________________________________________
add_10 (Add)                    (None, 48, 48, 64)   0           conv2d_38[0][0]
                                                                 dropout_33[0][0]
__________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 48, 48, 64)   256         add_10[0][0]
__________________________________________________________________________________________________
activation_32 (Activation)      (None, 48, 48, 64)   0           batch_normalization_33[0][0]
__________________________________________________________________________________________________
conv2d_39 (Conv2D)              (None, 48, 48, 16)   1040        activation_32[0][0]
__________________________________________________________________________________________________
dropout_34 (Dropout)            (None, 48, 48, 16)   0           conv2d_39[0][0]
__________________________________________________________________________________________________
batch_normalization_34 (BatchNo (None, 48, 48, 16)   64          dropout_34[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, 48, 48, 16)   0           batch_normalization_34[0][0]
__________________________________________________________________________________________________
conv2d_40 (Conv2D)              (None, 48, 48, 16)   2320        activation_33[0][0]
__________________________________________________________________________________________________
dropout_35 (Dropout)            (None, 48, 48, 16)   0           conv2d_40[0][0]
__________________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, 48, 48, 16)   64          dropout_35[0][0]
__________________________________________________________________________________________________
activation_34 (Activation)      (None, 48, 48, 16)   0           batch_normalization_35[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, 48, 48, 64)   1088        activation_34[0][0]
__________________________________________________________________________________________________
dropout_36 (Dropout)            (None, 48, 48, 64)   0           conv2d_41[0][0]
__________________________________________________________________________________________________
add_11 (Add)                    (None, 48, 48, 64)   0           add_10[0][0]
                                                                 dropout_36[0][0]
__________________________________________________________________________________________________
batch_normalization_36 (BatchNo (None, 48, 48, 64)   256         add_11[0][0]
__________________________________________________________________________________________________
activation_35 (Activation)      (None, 48, 48, 64)   0           batch_normalization_36[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, 48, 48, 16)   1040        activation_35[0][0]
__________________________________________________________________________________________________
dropout_37 (Dropout)            (None, 48, 48, 16)   0           conv2d_42[0][0]
__________________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, 48, 48, 16)   64          dropout_37[0][0]
__________________________________________________________________________________________________
activation_36 (Activation)      (None, 48, 48, 16)   0           batch_normalization_37[0][0]
```

```
____________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, 48, 48, 16)   2320        activation_36[0][0]
____________________________________________________________________________________________________
dropout_38 (Dropout)            (None, 48, 48, 16)   0           conv2d_43[0][0]
____________________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, 48, 48, 16)   64          dropout_38[0][0]
____________________________________________________________________________________________________
activation_37 (Activation)      (None, 48, 48, 16)   0           batch_normalization_38[0][0]
____________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, 48, 48, 64)   1088        activation_37[0][0]
____________________________________________________________________________________________________
dropout_39 (Dropout)            (None, 48, 48, 64)   0           conv2d_44[0][0]
____________________________________________________________________________________________________
add_12 (Add)                    (None, 48, 48, 64)   0           add_11[0][0]
                                                                 dropout_39[0][0]
____________________________________________________________________________________________________
batch_normalization_39 (BatchNo (None, 48, 48, 64)   256         add_12[0][0]
____________________________________________________________________________________________________
activation_38 (Activation)      (None, 48, 48, 64)   0           batch_normalization_39[0][0]
____________________________________________________________________________________________________
conv2d_45 (Conv2D)              (None, 24, 24, 64)   4160        activation_38[0][0]
____________________________________________________________________________________________________
dropout_40 (Dropout)            (None, 24, 24, 64)   0           conv2d_45[0][0]
____________________________________________________________________________________________________
batch_normalization_40 (BatchNo (None, 24, 24, 64)   256         dropout_40[0][0]
____________________________________________________________________________________________________
activation_39 (Activation)      (None, 24, 24, 64)   0           batch_normalization_40[0][0]
____________________________________________________________________________________________________
conv2d_46 (Conv2D)              (None, 24, 24, 64)   36928       activation_39[0][0]
____________________________________________________________________________________________________
dropout_41 (Dropout)            (None, 24, 24, 64)   0           conv2d_46[0][0]
____________________________________________________________________________________________________
batch_normalization_41 (BatchNo (None, 24, 24, 64)   256         dropout_41[0][0]
____________________________________________________________________________________________________
activation_40 (Activation)      (None, 24, 24, 64)   0           batch_normalization_41[0][0]
____________________________________________________________________________________________________
conv2d_47 (Conv2D)              (None, 24, 24, 128)  8320        activation_40[0][0]
____________________________________________________________________________________________________
conv2d_48 (Conv2D)              (None, 24, 24, 128)  8320        add_12[0][0]
____________________________________________________________________________________________________
dropout_42 (Dropout)            (None, 24, 24, 128)  0           conv2d_47[0][0]
____________________________________________________________________________________________________
add_13 (Add)                    (None, 24, 24, 128)  0           conv2d_48[0][0]
                                                                 dropout_42[0][0]
____________________________________________________________________________________________________
batch_normalization_42 (BatchNo (None, 24, 24, 128)  512         add_13[0][0]
____________________________________________________________________________________________________
activation_41 (Activation)      (None, 24, 24, 128)  0           batch_normalization_42[0][0]
```

```
__________________________________________________________________________________________________
conv2d_49 (Conv2D)              (None, 24, 24, 64)   8256        activation_41[0][0]
__________________________________________________________________________________________________
dropout_43 (Dropout)            (None, 24, 24, 64)   0           conv2d_49[0][0]
__________________________________________________________________________________________________
batch_normalization_43 (BatchNo (None, 24, 24, 64)   256         dropout_43[0][0]
__________________________________________________________________________________________________
activation_42 (Activation)      (None, 24, 24, 64)   0           batch_normalization_43[0][0]
__________________________________________________________________________________________________
conv2d_50 (Conv2D)              (None, 24, 24, 64)   36928       activation_42[0][0]
__________________________________________________________________________________________________
dropout_44 (Dropout)            (None, 24, 24, 64)   0           conv2d_50[0][0]
__________________________________________________________________________________________________
batch_normalization_44 (BatchNo (None, 24, 24, 64)   256         dropout_44[0][0]
__________________________________________________________________________________________________
activation_43 (Activation)      (None, 24, 24, 64)   0           batch_normalization_44[0][0]
__________________________________________________________________________________________________
conv2d_51 (Conv2D)              (None, 24, 24, 128)  8320        activation_43[0][0]
__________________________________________________________________________________________________
dropout_45 (Dropout)            (None, 24, 24, 128)  0           conv2d_51[0][0]
__________________________________________________________________________________________________
add_14 (Add)                    (None, 24, 24, 128)  0           add_13[0][0]
                                                                 dropout_45[0][0]
__________________________________________________________________________________________________
batch_normalization_45 (BatchNo (None, 24, 24, 128)  512         add_14[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, 24, 24, 128)  0           batch_normalization_45[0][0]
__________________________________________________________________________________________________
conv2d_52 (Conv2D)              (None, 24, 24, 64)   8256        activation_44[0][0]
__________________________________________________________________________________________________
dropout_46 (Dropout)            (None, 24, 24, 64)   0           conv2d_52[0][0]
__________________________________________________________________________________________________
batch_normalization_46 (BatchNo (None, 24, 24, 64)   256         dropout_46[0][0]
__________________________________________________________________________________________________
activation_45 (Activation)      (None, 24, 24, 64)   0           batch_normalization_46[0][0]
__________________________________________________________________________________________________
conv2d_53 (Conv2D)              (None, 24, 24, 64)   36928       activation_45[0][0]
__________________________________________________________________________________________________
dropout_47 (Dropout)            (None, 24, 24, 64)   0           conv2d_53[0][0]
__________________________________________________________________________________________________
batch_normalization_47 (BatchNo (None, 24, 24, 64)   256         dropout_47[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, 24, 24, 64)   0           batch_normalization_47[0][0]
__________________________________________________________________________________________________
conv2d_54 (Conv2D)              (None, 24, 24, 128)  8320        activation_46[0][0]
__________________________________________________________________________________________________
dropout_48 (Dropout)            (None, 24, 24, 128)  0           conv2d_54[0][0]
__________________________________________________________________________________________________
```

```
add_15 (Add)                    (None, 24, 24, 128)  0           add_14[0][0]
                                                                 dropout_48[0][0]
__________________________________________________________________________________________________
batch_normalization_48 (BatchNo (None, 24, 24, 128)  512         add_15[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)      (None, 24, 24, 128)  0           batch_normalization_48[0][0]
__________________________________________________________________________________________________
conv2d_55 (Conv2D)              (None, 12, 12, 128)  16512       activation_47[0][0]
__________________________________________________________________________________________________
dropout_49 (Dropout)            (None, 12, 12, 128)  0           conv2d_55[0][0]
__________________________________________________________________________________________________
batch_normalization_49 (BatchNo (None, 12, 12, 128)  512         dropout_49[0][0]
__________________________________________________________________________________________________
activation_48 (Activation)      (None, 12, 12, 128)  0           batch_normalization_49[0][0]
__________________________________________________________________________________________________
conv2d_56 (Conv2D)              (None, 12, 12, 128)  147584      activation_48[0][0]
__________________________________________________________________________________________________
dropout_50 (Dropout)            (None, 12, 12, 128)  0           conv2d_56[0][0]
__________________________________________________________________________________________________
batch_normalization_50 (BatchNo (None, 12, 12, 128)  512         dropout_50[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 12, 12, 128)  0           batch_normalization_50[0][0]
__________________________________________________________________________________________________
conv2d_57 (Conv2D)              (None, 12, 12, 256)  33024       activation_49[0][0]
__________________________________________________________________________________________________
conv2d_58 (Conv2D)              (None, 12, 12, 256)  33024       add_15[0][0]
__________________________________________________________________________________________________
dropout_51 (Dropout)            (None, 12, 12, 256)  0           conv2d_57[0][0]
__________________________________________________________________________________________________
add_16 (Add)                    (None, 12, 12, 256)  0           conv2d_58[0][0]
                                                                 dropout_51[0][0]
__________________________________________________________________________________________________
batch_normalization_51 (BatchNo (None, 12, 12, 256)  1024        add_16[0][0]
__________________________________________________________________________________________________
activation_50 (Activation)      (None, 12, 12, 256)  0           batch_normalization_51[0][0]
__________________________________________________________________________________________________
conv2d_59 (Conv2D)              (None, 12, 12, 128)  32896       activation_50[0][0]
__________________________________________________________________________________________________
dropout_52 (Dropout)            (None, 12, 12, 128)  0           conv2d_59[0][0]
__________________________________________________________________________________________________
batch_normalization_52 (BatchNo (None, 12, 12, 128)  512         dropout_52[0][0]
__________________________________________________________________________________________________
activation_51 (Activation)      (None, 12, 12, 128)  0           batch_normalization_52[0][0]
__________________________________________________________________________________________________
conv2d_60 (Conv2D)              (None, 12, 12, 128)  147584      activation_51[0][0]
__________________________________________________________________________________________________
dropout_53 (Dropout)            (None, 12, 12, 128)  0           conv2d_60[0][0]
__________________________________________________________________________________________________
```

```
batch_normalization_53 (BatchNo (None, 12, 12, 128)  512         dropout_53[0][0]
__________________________________________________________________________________________________
activation_52 (Activation)      (None, 12, 12, 128)  0           batch_normalization_53[0][0]
__________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, 12, 12, 256)  33024       activation_52[0][0]
__________________________________________________________________________________________________
dropout_54 (Dropout)            (None, 12, 12, 256)  0           conv2d_61[0][0]
__________________________________________________________________________________________________
add_17 (Add)                    (None, 12, 12, 256)  0           add_16[0][0]
                                                                 dropout_54[0][0]
__________________________________________________________________________________________________
batch_normalization_54 (BatchNo (None, 12, 12, 256)  1024        add_17[0][0]
__________________________________________________________________________________________________
activation_53 (Activation)      (None, 12, 12, 256)  0           batch_normalization_54[0][0]
__________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, 12, 12, 128)  32896       activation_53[0][0]
__________________________________________________________________________________________________
dropout_55 (Dropout)            (None, 12, 12, 128)  0           conv2d_62[0][0]
__________________________________________________________________________________________________
batch_normalization_55 (BatchNo (None, 12, 12, 128)  512         dropout_55[0][0]
__________________________________________________________________________________________________
activation_54 (Activation)      (None, 12, 12, 128)  0           batch_normalization_55[0][0]
__________________________________________________________________________________________________
conv2d_63 (Conv2D)              (None, 12, 12, 128)  147584      activation_54[0][0]
__________________________________________________________________________________________________
dropout_56 (Dropout)            (None, 12, 12, 128)  0           conv2d_63[0][0]
__________________________________________________________________________________________________
batch_normalization_56 (BatchNo (None, 12, 12, 128)  512         dropout_56[0][0]
__________________________________________________________________________________________________
activation_55 (Activation)      (None, 12, 12, 128)  0           batch_normalization_56[0][0]
__________________________________________________________________________________________________
conv2d_64 (Conv2D)              (None, 12, 12, 256)  33024       activation_55[0][0]
__________________________________________________________________________________________________
dropout_57 (Dropout)            (None, 12, 12, 256)  0           conv2d_64[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 12, 12, 1024) 263168      dropout_57[0][0]
__________________________________________________________________________________________________
dropout_58 (Dropout)            (None, 12, 12, 1024) 0           dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 12, 12, 512)  524800      dropout_58[0][0]
__________________________________________________________________________________________________
dropout_59 (Dropout)            (None, 12, 12, 512)  0           dense_4[0][0]
__________________________________________________________________________________________________
conv2d_65 (Conv2D)              (None, 12, 12, 256)  1179904     dropout_59[0][0]
__________________________________________________________________________________________________
average_pooling2d_2 (AveragePoo (None, 6, 6, 256)    0           conv2d_65[0][0]
__________________________________________________________________________________________________
dropout_60 (Dropout)            (None, 6, 6, 256)    0           average_pooling2d_2[0][0]
```

```
________________________________________________________________________________________________
batch_normalization_58 (BatchNo (None, 6, 6, 256)    1024        dropout_60[0][0]
________________________________________________________________________________________________
conv2d_66 (Conv2D)              (None, 6, 6, 8)      18440       batch_normalization_58[0][0]
________________________________________________________________________________________________
global_average_pooling2d_2 (Glo (None, 8)            0           conv2d_66[0][0]
________________________________________________________________________________________________
predictions (Activation)        (None, 8)            0           global_average_pooling2d_2[0][0
================================================================================================
Total params: 2,832,456
Trainable params: 2,827,240
Non-trainable params: 5,216
________________________________________________________________________________________________
Using real-time data augmentation.
Epoch 1/100
Learning rate:  0.001
1000/1000 [==============================] - 611s 611ms/step - loss: 1.7608 - acc: 0.5300 - val_
Epoch 2/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 1.3438 - acc: 0.6810 - val_
Epoch 3/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 1.2275 - acc: 0.7197 - val_
Epoch 4/100
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.1718 - acc: 0.7379 - val_
Epoch 5/100
Learning rate:  0.001
1000/1000 [==============================] - 588s 588ms/step - loss: 1.1333 - acc: 0.7525 - val_
Epoch 6/100
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.1130 - acc: 0.7596 - val_
Epoch 7/100
Learning rate:  0.001
1000/1000 [==============================] - 591s 591ms/step - loss: 1.0920 - acc: 0.7663 - val_
Epoch 8/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 1.0797 - acc: 0.7706 - val_
Epoch 9/100
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.0689 - acc: 0.7753 - val_
Epoch 10/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 1.0608 - acc: 0.7792 - val_
Epoch 11/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 1.0532 - acc: 0.7817 - val_
Epoch 12/100
```

```
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.0514 - acc: 0.7818 - val_
Epoch 13/100
Learning rate:  0.001
1000/1000 [==============================] - 587s 587ms/step - loss: 1.0457 - acc: 0.7844 - val_
Epoch 14/100
Learning rate:  0.001
1000/1000 [==============================] - 594s 594ms/step - loss: 1.0402 - acc: 0.7888 - val_
Epoch 15/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0362 - acc: 0.7889 - val_
Epoch 16/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0344 - acc: 0.7906 - val_
Epoch 17/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0296 - acc: 0.7926 - val_
Epoch 18/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0276 - acc: 0.7929 - val_
Epoch 19/100
Learning rate:  0.001
1000/1000 [==============================] - 591s 591ms/step - loss: 1.0237 - acc: 0.7955 - val_
Epoch 20/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0238 - acc: 0.7960 - val_
Epoch 21/100
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.0193 - acc: 0.7967 - val_
Epoch 22/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0200 - acc: 0.7969 - val_
Epoch 23/100
Learning rate:  0.001
1000/1000 [==============================] - 591s 591ms/step - loss: 1.0173 - acc: 0.7984 - val_
Epoch 24/100
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0148 - acc: 0.7996 - val_
Epoch 25/100
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.0129 - acc: 0.8000 - val_
Epoch 26/100
Learning rate:  0.001
1000/1000 [==============================] - 592s 592ms/step - loss: 1.0136 - acc: 0.7983 - val_
Epoch 27/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 1.0105 - acc: 0.8000 - val_
Epoch 28/100
```

```
Learning rate:  0.001
1000/1000 [==============================] - 593s 593ms/step - loss: 1.0083 - acc: 0.8018 - val_
Epoch 29/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 1.0066 - acc: 0.8030 - val_
Epoch 30/100
Learning rate:  0.001
1000/1000 [==============================] - 591s 591ms/step - loss: 1.0071 - acc: 0.8006 - val_
Epoch 31/100
Learning rate:  0.001
1000/1000 [==============================] - 588s 588ms/step - loss: 1.0066 - acc: 0.8028 - val_
Epoch 32/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 1.0035 - acc: 0.8036 - val_
Epoch 33/100
Learning rate:  0.001
1000/1000 [==============================] - 588s 588ms/step - loss: 1.0012 - acc: 0.8051 - val_
Epoch 34/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 0.9998 - acc: 0.8041 - val_
Epoch 35/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 0.9979 - acc: 0.8052 - val_
Epoch 36/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 0.9974 - acc: 0.8067 - val_
Epoch 37/100
Learning rate:  0.001
1000/1000 [==============================] - 590s 590ms/step - loss: 0.9967 - acc: 0.8059 - val_
Epoch 38/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 0.9978 - acc: 0.8059 - val_
Epoch 39/100
Learning rate:  0.001
1000/1000 [==============================] - 589s 589ms/step - loss: 0.9951 - acc: 0.8068 - val_
Epoch 40/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9938 - acc: 0.8076 - val_
Epoch 41/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9953 - acc: 0.8051 - val_
Epoch 42/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9919 - acc: 0.8084 - val_
Epoch 43/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9909 - acc: 0.8082 - val_
Epoch 44/100
```

```
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9900 - acc: 0.8090 - val_
Epoch 45/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9896 - acc: 0.8088 - val_
Epoch 46/100
Learning rate:  0.001
1000/1000 [==============================] - 585s 585ms/step - loss: 0.9906 - acc: 0.8088 - val_
Epoch 47/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9886 - acc: 0.8080 - val_
Epoch 48/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9864 - acc: 0.8105 - val_
Epoch 49/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9875 - acc: 0.8103 - val_
Epoch 50/100
Learning rate:  0.001
1000/1000 [==============================] - 585s 585ms/step - loss: 0.9827 - acc: 0.8108 - val_
Epoch 51/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9837 - acc: 0.8108 - val_
Epoch 52/100
Learning rate:  0.001
1000/1000 [==============================] - 585s 585ms/step - loss: 0.9837 - acc: 0.8093 - val_
Epoch 53/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9824 - acc: 0.8121 - val_
Epoch 54/100
Learning rate:  0.001
1000/1000 [==============================] - 585s 585ms/step - loss: 0.9812 - acc: 0.8108 - val_
Epoch 55/100
Learning rate:  0.001
1000/1000 [==============================] - 585s 585ms/step - loss: 0.9824 - acc: 0.8109 - val_
Epoch 56/100
Learning rate:  0.001
1000/1000 [==============================] - 587s 587ms/step - loss: 0.9796 - acc: 0.8133 - val_
Epoch 57/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9798 - acc: 0.8113 - val_
Epoch 58/100
Learning rate:  0.001
1000/1000 [==============================] - 586s 586ms/step - loss: 0.9814 - acc: 0.8126 - val_
Epoch 59/100
Learning rate:  0.001
 665/1000 [===================>..........] - ETA: 3:15 - loss: 0.9764 - acc: 0.8147
```

1.6 Results:

For the FERPlus dataset we wanted to explore a completely new and different architecture. Therefore, we added a sequence of the following layers: * Fully-connected * Dropout * Fully-connected * Dropout * Convolution * AveragePooling * Dropout * Batch normalization * Convolution * GlobalAveragePooling

The results obtained using this "frankenstein" architecture were better than what we were expecting. We obtained a validation accuracy of **0.803** at the 48 epoch.

We are not fully-aware of the effect of having convolutional layers after fully-connected layers had during the learning process. We were just trying something different that what it is normally done.

2 Conclusion

2.1 Models

- InceptionV3
- ResNetV1
- ResNetV2

2.2 Hyperparameters

- Dense layer size
- Dropout probability
- Number of fixed layers
- Learning rate
- decay schedule
- batch size
- epochs

2.3 Network architecture description:

We have used ResNetv2 as our base model. The number of convolutional and batch normalization layers and also the number of filters in each of the convolutional layers has been taken the resnet paper [1][5].

We have convolutional filter of size 3×3 as described in the VGG paper [2]. Since stacking two 3×3 convolutional gives the same receptive field as a single 5×5 with lesser number of parameters to train.

We have also used Global average pooling layer instead of the fully connected layer as described in the "Network inside Network"[3] as it enforces correspondance between feature maps and categories and is less prone to overfitting and hence does not require aggressive dropout mechanism. For the global average pooling to work we need to have as many filters as the number of classes. Hence the final convolutional layer has 10 filters.

This worked well for FER dataset. For CIFAR, we got better results when using the fully connected layer rather than the global pooling. We have used three fully connected layer to avoid the computational bottleneck which would be the case if only layer were used. This has been followed from Inception v2 paper [4].

2.4 Reference:

[1] He, Kaiming et al. “Deep Residual Learning for Image Recognition.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 770-778.

[2] Min Lin et al. "Network In Network", arxiv 2013

[3] Simonyan et al. "Very Deep Convolutional Networks for Large-Scale Image Recognition ", International Conference on Learning Representations 2015

[4] Rethinking the Inception Architecture for Computer Vision

[5] ResNet implementation acquired from https://github.com/keras-team/keras/blob/master/examples/cifar10_resnet.py