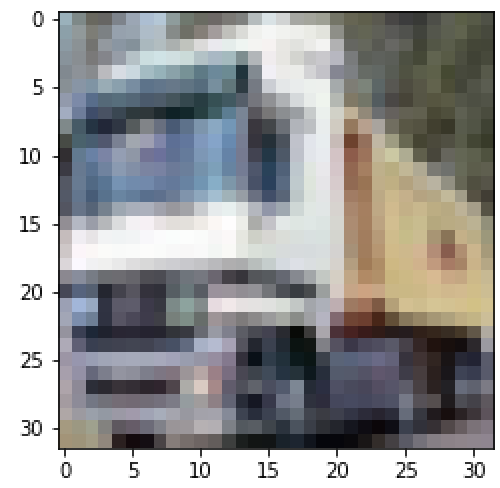
```
%tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt

np.random.seed(451)

from keras.datasets import cifar10

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

x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
Using TensorFlow backend.
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
```

```
#gray = 0.2989 * r + 0.5870 * g + 0.1140 * b
x_train_gray = np.dot(x_train[:,:,:,:3], [0.299, 0.587, 0.114])
x_test_gray = np.dot(x_test[:,:,:,:3], [0.299, 0.587, 0.114])

x_train_gray = x_train_gray.reshape(-1,32,32,1)
x_test_gray = x_test_gray.reshape(-1,32,32,1)

from keras.utils.np_utils import to_categorical

y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)

plt.imshow(x_train[1])
plt.show()

plt.imshow(x_train_gray[1,:,:,0], cmap='gray')
plt.show()
```



```
np.random.seed(451)

import datetime

from keras.layers import Flatten, Activation, Conv2D, MaxPool2D, AvgPool2D, Dense, Dropout
from keras.optimizers import Adam, SGD
from keras.models import Sequential
import keras.backend as K
from keras.regularizers import l1,l2
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, ReduceLROnPlateau
from keras.models import model_from_json, Model

def build_tower(input_layer, features_nr, shape, tower_nr,
                dropout=False, normalization=False, regularization="l2", dropout_ratio=0.2
    #3x3 kernel tower
    tower = Conv2D(features_nr, (1,1), padding='same', activation='relu',
                     kernel_regularizer=regularization, name='tower_%d_%dx%da'%(tower_nr,
    tower = Conv2D(features_nr*2, shape, padding='same', activation='relu',
                     kernel_regularizer=regularization, name='tower_%d_%dx%db'%(tower_nr,
    #condidional dropout/normalization
    if dropout:
        tower = Dropout(dropout_ratio, name='tower_%d_%dx%ddrop'%(tower_nr, shape[0], shap
    if normalization:
        tower = BatchNormalization(name='tower_%d_%dx%dnorm'%(tower_nr, shape[0], shape[1]

    return tower

def build_simple_tower(input_layer, features_nr, shape, tower_nr,
                dropout=False, normalization=False, regularization="l2", dropout_ratio=0.2
    #3x3 kernel tower
    tower = Conv2D(features_nr, shape, padding='same', activation='relu',
                     kernel_regularizer=regularization,
                   name='tower_simple_%d_%dx%db'%(tower_nr, shape[0], shape[1]))(input_lay
    #condidional dropout/normalization
    if dropout:
        tower = Dropout(dropout_ratio, name='tower_%d_%dx%ddrop'%(tower_nr, shape[0], shap
    if normalization:
        tower = BatchNormalization(name='tower_%d_%dx%dnorm'%(tower_nr, shape[0], shape[1]

    return tower
```

```
def build_tower_subsample(input_layer, features_nr, shape, tower_nr,
                          dropout=False, normalization=False, regularization='l2', dropout
    tower = build_tower(input_layer, features_nr, shape, tower_nr,
                        dropout, normalization, regularization, dropout_ratio)
    pool = MaxPooling2D((2,2), padding='same', name='tower_%d_2x2subsample'%(tower_nr))(to

    return pool

def build_simple_tower_subsample(input_layer, features_nr, shape, tower_nr,
                          dropout=False, normalization=False, regularization='l2', dropout
    tower = build_simple_tower(input_layer, features_nr, shape, tower_nr,
                        dropout, normalization, regularization, dropout_ratio)
    pool = MaxPooling2D((2,2), padding='same', name='tower_%d_2x2subsample'%(tower_nr))(to

    return pool

def build_dense(input_layer, neurons_nr, dense_nr,
                dropout=False, normalization=False, regularization='l2', dropout_ratio=0.5
    dense = Dense(neurons_nr, kernel_regularizer=regularization,
                  name='dense_%d_%d'%(dense_nr, neurons_nr))(input_layer)

    if dropout:
        dense = Dropout(dropout_ratio, name='dense_%d_%ddrop'%(dense_nr, neurons_nr))(dens
    if normalization:
        dense = BatchNormalization(name='dense_%d_%dnorm'%(dense_nr, neurons_nr))(dense)

    return dense

def build_inception_module(input_layer, features_nr, module_nr,
                           dropout=False, normalization=False, regularization='l2', dropou
    #feature_nr is an array we'll use to build our layers
    #data is in the form: [1x1, 3x3 reduce, 3x3, 5x5 reduce, 5x5, pool proj]

    inception_1x1 = Conv2D(features_nr[0],1,1,border_mode='same',activation='relu',name='i

    inception_3x3_reduce = Conv2D(features_nr[1],1,1,border_mode='same',activation='relu',

    inception_3x3 = Conv2D(features_nr[2],3,3,border_mode='same',activation='relu',name='i

    inception_5x5_reduce = Conv2D(features_nr[3],1,1,border_mode='same',activation='relu',

    inception_5x5 = Conv2D(features_nr[4],5,5,border_mode='same',activation='relu',name='i

    inception_pool = MaxPooling2D(pool_size=(3,3),strides=(1,1),border_mode='same',name='i

    inception_pool_proj = Conv2D(features_nr[5],1,1,border_mode='same',activation='relu',n

    inception_output = concatenate([inception_1x1,inception_3x3,inception_5x5,inception_po

    if dropout:
        inception_output = Dropout(dropout_ratio, name='inception_%d_/output_drop'%(module
    if normalization:
        inception_output = BatchNormalization(name='inception_%d_/output_norm'%(module_nr)

    pooled = MaxPooling2D((2,2), padding='same', name='inception_%d_2x2subsample'%(module_
    pooled = MaxPooling2D((1,1), padding='same', name='inception_%d_1x1subsample'%(module_

    return pooled

i='cifar10-nrcrt7-'+datetime.datetime.now().strftime("%I:%M%p_%B-%d-%Y")

K.clear_session()

!mkdir -p models
!mkdir -p logs

a = EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=1, mode='auto')#wi
b = ModelCheckpoint(monitor='val_loss', filepath='./models/'+str(i)+'.hdf5', verbose=1, sa
c = TensorBoard(log_dir='./logs/'+str(i),
                write_grads=True,
                write_graph=True,
                write_images=True,
                batch_size=128)#saves a log file for tensorboard; remember to save differe

#we'll use this instead of decay
d = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, verbose=0, mode='auto',

callbacks=[a,b,c,d]

#------------model definition-------------------

use_norm = True
lrate = 0.001

input_img = Input(shape = (32, 32, 3), name='input')

#conv_1 = Conv2D(1, (1,1), padding='same', activation='relu',
               # kernel_regularizer = regularization, name='conv_64x64x1_inception_in')(ir

#hopefully this will learn a good internal representation of the image channels
#conv_1 = Conv2D(1, (1,1), padding='same', activation='relu',
                #kernel_regularizer = regularization, name='conv_64x64x1_inception_in')(ir

inception_1 = build_inception_module(input_img, [64,96,128,16,32,32], 1, False, use_norm)

inception_2 = build_inception_module(inception_1, [128,128,192,32,96,64], 2, False, use_nc

inception_3 = build_inception_module(inception_2, [192,96,208,16,48,64], 3, False, use_nor

inception_4 = build_inception_module(inception_3, [160, 112, 224, 24, 64, 64], 4, False, u
#tower_3 = build_simple_tower(inception_2, 144, (3,3),  3, False, use_norm)
#tower_4 = build_simple_tower_subsample(tower_3, 144, (3,3), 4, False, use_norm)

#tower_5 = build_simple_tower(tower_4, 288, (3,3),  5, False, use_norm)
#tower_6 = build_simple_tower_subsample(tower_5, 288, (3,3), 6, False, use_norm)

#model top

flat_pool = AveragePooling2D(pool_size=(2, 2), padding='valid')(inception_4)

flat = Flatten()(flat_pool)
```

```
dense_5 = build_dense(flat, 128, 1, True, use_norm)

dense_6 = build_dense(dense_5, 64, 2, True, use_norm)

out = Dense(10, activation='softmax')(dense_6)

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

#------------------------------------------------

model.compile(loss='binary_crossentropy',
              optimizer=Adam(lrate),
              metrics=['accuracy'])

model.summary()

model_json = model.to_json()
with open("./models/"+str(i)+".json", "w") as json_file:
    json_file.write(model_json)

print("Saved model to" + "../models/"+str(i)+".json")
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/resou
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:75: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:77: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:79: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:81: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:83: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:85: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:87: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:75: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:77: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:79: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:81: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:83: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:85: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:87: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:75: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:77: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:79: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:81: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:83: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:85: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:87: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:75: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:77: UserWarning: Update
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:79: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:81: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:83: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:85: UserWarning: Update
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:87: UserWarning: Update
Model: "model_1"
________________________________________________________________________________________
Layer (type)                     Output Shape         Param #     Connected to
========================================================================================
input (InputLayer)               (None, 32, 32, 3)    0
________________________________________________________________________________________
inception_1_/3x3_reduce (Conv2D (None, 32, 32, 96)   384         input[0][0]
________________________________________________________________________________________
inception_1_/5x5_reduce (Conv2D (None, 32, 32, 16)   64          input[0][0]
________________________________________________________________________________________
inception_1_/pool (MaxPooling2D (None, 32, 32, 3)    0           input[0][0]
________________________________________________________________________________________
inception_1_/1x1 (Conv2D)        (None, 32, 32, 64)   256         input[0][0]
________________________________________________________________________________________
inception_1_/3x3 (Conv2D)        (None, 32, 32, 128)  110720      inception_1_/3x3_rec
________________________________________________________________________________________
inception_1_/5x5 (Conv2D)        (None, 32, 32, 32)   12832       inception_1_/5x5_rec
________________________________________________________________________________________
inception_1_/pool_proj (Conv2D) (None, 32, 32, 32)   128         inception_1_/pool[0]
________________________________________________________________________________________
inception_1_/output (Concatenat (None, 32, 32, 256)  0           inception_1_/1x1[0][
                                                                  inception_1_/3x3[0][
                                                                  inception_1_/5x5[0][
                                                                  inception_1_/pool_pr
________________________________________________________________________________________
inception_1_/output_norm (Batch (None, 32, 32, 256)  1024        inception_1_/output[
inception_1_2x2subsample (MaxPo (None, 16, 16, 256)  0           inception_1_/output_

inception_2_/3x3_reduce (Conv2D (None, 16, 16, 128)  32896       inception_1_2x2subsa

inception_2_/5x5_reduce (Conv2D (None, 16, 16, 32)   8224        inception_1_2x2subsa

inception_2_/pool (MaxPooling2D (None, 16, 16, 256)  0           inception_1_2x2subsa

inception_2_/1x1 (Conv2D)       (None, 16, 16, 128)  32896       inception_1_2x2subsa

inception_2_/3x3 (Conv2D)       (None, 16, 16, 192)  221376      inception_2_/3x3_rec

inception_2_/5x5 (Conv2D)       (None, 16, 16, 96)   76896       inception_2_/5x5_rec

inception_2_/pool_proj (Conv2D) (None, 16, 16, 64)   16448       inception_2_/pool[0]

inception_2_/output (Concatenat (None, 16, 16, 480)  0           inception_2_/1x1[0][
                                                                 inception_2_/3x3[0][
                                                                 inception_2_/5x5[0][
                                                                 inception_2_/pool_pr

inception_2_/output_norm (Batch (None, 16, 16, 480)  1920        inception_2_/output[

inception_2_2x2subsample (MaxPo (None, 8, 8, 480)    0           inception_2_/output_

inception_3_/3x3_reduce (Conv2D (None, 8, 8, 96)     46176       inception_2_2x2subsa

inception_3_/5x5_reduce (Conv2D (None, 8, 8, 16)     7696        inception_2_2x2subsa

inception_3_/pool (MaxPooling2D (None, 8, 8, 480)    0           inception_2_2x2subsa

inception_3_/1x1 (Conv2D)       (None, 8, 8, 192)    92352       inception_2_2x2subsa

inception_3_/3x3 (Conv2D)       (None, 8, 8, 208)    179920      inception_3_/3x3_rec

inception_3_/5x5 (Conv2D)       (None, 8, 8, 48)     19248       inception_3_/5x5_rec

inception_3_/pool_proj (Conv2D) (None, 8, 8, 64)     30784       inception_3_/pool[0]

inception_3_/output (Concatenat (None, 8, 8, 512)    0           inception_3_/1x1[0][
                                                                 inception_3_/3x3[0][
                                                                 inception_3_/5x5[0][
                                                                 inception_3_/pool_pr

inception_3_/output_norm (Batch (None, 8, 8, 512)    2048        inception_3_/output[

inception_3_2x2subsample (MaxPo (None, 4, 4, 512)    0           inception_3_/output_

inception_4_/3x3_reduce (Conv2D (None, 4, 4, 112)    57456       inception_3_2x2subsa

inception_4_/5x5_reduce (Conv2D (None, 4, 4, 24)     12312       inception_3_2x2subsa

inception_4_/pool (MaxPooling2D (None, 4, 4, 512)    0           inception_3_2x2subsa

inception_4_/1x1 (Conv2D)       (None, 4, 4, 160)    82080       inception_3_2x2subsa

inception_4_/3x3 (Conv2D)       (None, 4, 4, 224)    226016      inception_4_/3x3_rec

inception_4_/5x5 (Conv2D)       (None, 4, 4, 64)     38464       inception_4_/5x5_rec

inception_4_/pool_proj (Conv2D) (None, 4, 4, 64)     32832       inception_4_/pool[0]
```

```
 inception_4_/pool_proj (Conv2D) (None, 4, 4, 64)     32832       inception_4_/pool[0]
 ________________________________________________________________________________________
 inception_4_/output (Concatenat (None, 4, 4, 512)    0           inception_4_/1x1[0][
```

```
import tensorflow as tf

with tf.device('/gpu:0'):
  model.fit(x_train, y_train_cat, batch_size=128, epochs=100, validation_split=0.2,verbose

result = model.evaluate(x_test, y_test_cat)

print("Accuracy on test set: ",result[1]*100,"%")
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/math_
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

Train on 40000 samples, validate on 10000 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks/tensor

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks/tensor

Epoch 1/100
40000/40000 [==============================] - 82s 2ms/step - loss: 0.9786 - accuracy

Epoch 00001: val_loss improved from inf to 0.54750, saving model to ./models/cifar10-
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks/tensor

Epoch 2/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.3377 - accuracy

Epoch 00002: val_loss improved from 0.54750 to 0.33929, saving model to ./models/cifa
Epoch 3/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.2591 - accuracy

Epoch 00003: val_loss improved from 0.33929 to 0.31674, saving model to ./models/cifa
Epoch 4/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.2296 - accuracy

Epoch 00004: val_loss improved from 0.31674 to 0.24687, saving model to ./models/cifa
Epoch 5/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.2144 - accuracy

Epoch 00005: val_loss did not improve from 0.24687
Epoch 6/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.2049 - accuracy

Epoch 00006: val_loss improved from 0.24687 to 0.22281, saving model to ./models/cifa
Epoch 7/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.1969 - accuracy

Epoch 00007: val_loss did not improve from 0.22281
Epoch 8/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.1887 - accuracy

Epoch 00008: val_loss did not improve from 0.22281
Epoch 9/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.1846 - accuracy

Epoch 00009: val_loss improved from 0.22281 to 0.18743, saving model to ./models/cifa
Epoch 10/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.1789 - accuracy

Epoch 00010: val_loss did not improve from 0.18743
Epoch 11/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.1765 - accuracy

Epoch 00011: val_loss did not improve from 0.18743
Epoch 12/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.1687 - accuracy

Epoch 00012: val_loss did not improve from 0.18743
Epoch 13/100
```

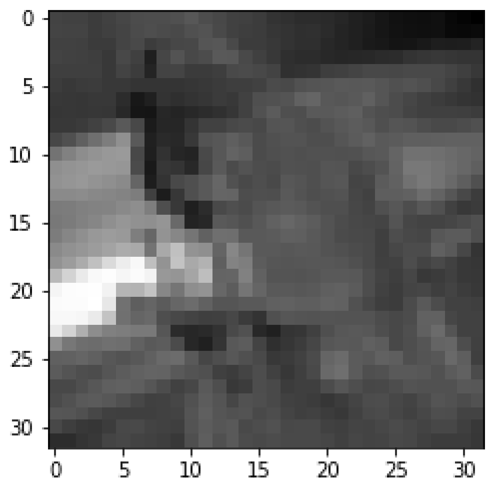
```
40000/40000 [==============================] - 72s 2ms/step - loss: 0.1663 - accuracy

Epoch 00013: val_loss did not improve from 0.18743
Epoch 14/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.1645 - accuracy

Epoch 00014: val_loss did not improve from 0.18743
Epoch 15/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.1309 - accuracy

Epoch 00015: val_loss improved from 0.18743 to 0.14383, saving model to ./models/cifa
Epoch 16/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.1107 - accuracy

Epoch 00016: val_loss improved from 0.14383 to 0.13995, saving model to ./models/cifa
Epoch 17/100
40000/40000 [==============================] - 72s 2ms/step - loss: 0.0990 - accuracy

Epoch 00017: val_loss improved from 0.13995 to 0.13503, saving model to ./models/cifa
Epoch 18/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.0906 - accuracy

Epoch 00018: val_loss improved from 0.13503 to 0.13293, saving model to ./models/cifa
Epoch 19/100
40000/40000 [==============================] - 71s 2ms/step - loss: 0.0834 - accuracy
```

```
predict = model.predict(x_test)
m = max(predict[7])
index = [i for i,j in enumerate(predict[7]) if j == m]
print("The value of the prediction of test sample with index 7 is :")
print(index)
```

```
The value of the prediction of test sample with index 7 is :
[6]
```

```
print("The actual class of test sample with index 7 is: ")
plt.imshow(x_test[7,:,:,0], cmap='gray')
plt.show()
```

```
The actual class of test sample with index 7 is:
```



```
40000/40000 [==============================] - 72s 2ms/step - loss: 0.0507 - accuracy
```

```
model.load_weights('./models/cifar10-nrcrt7-04:40PM_May-08-2020.hdf5')

result = model.evaluate(x_test, y_test_cat)
```

```
print(result)
```

```
10000/10000 [==============================] - 8s 794us/step
[0.13478883649110793, 0.9691197276115417]
Epoch 00029: early stopping
10000/10000 [==============================] - 8s 815us/step
Accuracy on test set:  96.87296748161316 %
```