Multi Class Classification Using Keras:

In [0]:

```
# Mount Google Drive
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive", forc
e_remount=True).
/gdrive
```

In [0]:

```
# Directory Structure:

'''
Dataset/

        train/

              category 1/

                        img1.png
                        img2.png
                        ...

              category 2/

                        img1.png
                        img2.png
                        ...

              category 3/

                        img1.png
                        img2.png
                        ...

              category 4/

                        img1.png
                        img2.png
                        ...

         test/

              category 1/

                        img1.png
                        img2.png
                        ...

              category 2/

                        img1.png
                        img2.png
                        ...

              category 3/

                        img1.png
                        img2.png
                        ...

              category 4/

                        img1.png
                        img2.png
```

```
                    ...

'''

# Total samples in train set : 440 [Categories: 102, 107, 112, 119]
# Total samples in test set  : 30  [Categories: 5, 9, 11, 5]
```

Out[0]:

```
'\nDataset/\n\n        train/\n        \n              category 1/\n              \n
img1.png\n                      img2.png\n                          ...\n                       \n
category 2/\n              \n                      img1.png\n                          img2.png\n
...\n               \n              category 3/\n              \n                          img1.png\r
img2.png\n                      ...\n                      \n              category 4/\n
\n                      img1.png\n                      img2.png\n                          ...\n\n
test/\n         \n              category 1/\n              \n                      img1.png\n
img2.png\n                      ...\n                      \n              category 2/\n
\n                      img1.png\n                      img2.png\n                          ...\n
\n              category 3/\n              \n                      img1.png\n
img2.png\n                      ...\n                      \n              category 4/\n
\n                      img1.png\n                      img2.png\n                          ...   \r
\n'
```

In [0]:

```
# Import packages
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from keras.utils.np_utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np
import os
import warnings
warnings.filterwarnings('ignore')
```

```
Using TensorFlow backend.
```

In [0]:

```
# Print number of images present in Train and Test set
train_images_count = len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/train/category 1')
) + \
                     len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/train/category 2'))
+ \
                     len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/train/category 3'))
+ \
                     len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/train/category 4'))

test_images_count = len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/test/category 1'))
+ \
                    len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/test/category 2')) +
\
                    len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/test/category 3')) +
\
                    len(os.listdir('./My Drive/Busigence/Image_2/Input/Dataset/test/category 4'))


print(train_images_count)
print(test_images_count)
```

```
440
30
```

In [0]:

```
# dimensions of our images.
img_width, img_height = 800, 800

# Path to save the trained weights (model)
top_model_weights_path = './Images/Output/bottleneck_fc_model.h5'
```

```
# Path to train and test directories
train_data_dir = './Images/Input/Dataset/train'
validation_data_dir = './Images/Input/Dataset/test'

# Number of train and test images in the dataset
nb_train_samples = train_images_count
nb_validation_samples = test_images_count
epochs = 50
batch_size = 10

train_categories_sample_size = [102, 107, 112, 119]
test_categories_sample_size = [5, 9, 11, 5]
one_hot_val = [0,1,2,3]
```

In [0]:

```
# Function to convert and save train and test images to numpy arrays
def save_bottleneck_features():
    datagen = ImageDataGenerator(rescale=1. / 255)

    # build the VGG16 network
    model = applications.VGG16(include_top=False, weights=None)

    generator = datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

    bottleneck_features_train = model.predict_generator(
                                generator, nb_train_samples // batch_size)

    np.save(open('./Images/Output/bottleneck_features_train.npy', 'wb'), bottleneck_features_train
)
    print("Saved training images as .npy files")

    generator = datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)
    bottleneck_features_validation = model.predict_generator(
                                    generator, nb_validation_samples // batch_size)
    np.save(open('./Images/Output/bottleneck_features_validation.npy', 'wb'),
bottleneck_features_validation)
    print("Saved testing images as .npy files")

save_bottleneck_features()
```

```
WARNING: Logging before flag parsing goes to stderr.
W0818 11:44:54.149414 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:74: The name
tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0818 11:44:54.172227 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name
tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

W0818 11:44:54.175867 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name
tf.random_uniform is deprecated. Please use tf.random.uniform instead.

W0818 11:44:54.209740 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3976: The name
tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

W0818 11:44:54.487055 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:2741: The name
tf.Session is deprecated. Please use tf.compat.v1.Session instead.
```

```
Found 440 images belonging to 4 classes.
```

```
W0818 11:44:54.775104 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:174: The name
tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_session instead.
```

```
Saved training images as .npy files
Found 30 images belonging to 4 classes.
Saved testing images as .npy files
```

In [0]:

```
# Encoding train and test labels
train_labels_encode, test_labels_encode = [],  []
for i in range(len(one_hot_val)):
    train_labels_encode.extend([one_hot_val[i]] * train_categories_sample_size[i])
    test_labels_encode.extend([one_hot_val[i]] * test_categories_sample_size[i])
train_labels_encode = to_categorical(train_labels_encode)
test_labels_encode = to_categorical(test_labels_encode)

# Load numpy values of train set
train_data = np.load(open('./Images/Output/bottleneck_features_train.npy', 'rb'))
train_labels = train_labels_encode

# Load numpy values of test set
validation_data = np.load(open('./Images/Output/bottleneck_features_validation.npy', 'rb'))
validation_labels = test_labels_encode

# Declare a sequential model
model = Sequential()
model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='sigmoid'))

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

# Summary of the model
model.summary()
```

```
W0818 11:47:00.952584 139929989732224 deprecation.py:506] From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future
version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
W0818 11:47:00.985032 139929989732224 deprecation_wrapper.py:119] From
/usr/local/lib/python3.6/dist-packages/keras/optimizers.py:790: The name tf.train.Optimizer is dep
recated. Please use tf.compat.v1.train.Optimizer instead.

W0818 11:47:01.079891 139929989732224 deprecation.py:323] From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 320000)            0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               81920256
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
=================================================================
Total params: 81,921,284
Trainable params: 81,921,284
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
# Model training
history = model.fit(train_data, train_labels,
          epochs=epochs,
          batch_size=batch_size,
          validation_data=(validation_data, validation_labels))
model.save_weights(top_model_weights_path)
```

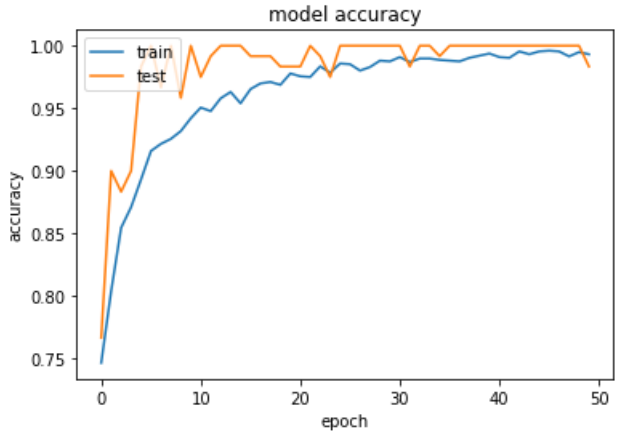
```
Train on 440 samples, validate on 30 samples
Epoch 1/50
440/440 [==============================] - 5s 12ms/step - loss: 0.5338 - acc: 0.7466 - val_loss: 0
.4350 - val_acc: 0.7667
Epoch 2/50
440/440 [==============================] - 4s 9ms/step - loss: 0.4118 - acc: 0.8040 - val_loss: 0.
3382 - val_acc: 0.9000
Epoch 3/50
440/440 [==============================] - 4s 9ms/step - loss: 0.3464 - acc: 0.8545 - val_loss: 0.
2893 - val_acc: 0.8833
Epoch 4/50
440/440 [==============================] - 4s 9ms/step - loss: 0.3147 - acc: 0.8710 - val_loss: 0.
2411 - val_acc: 0.9000
Epoch 5/50
440/440 [==============================] - 4s 9ms/step - loss: 0.2725 - acc: 0.8932 - val_loss: 0.
1786 - val_acc: 0.9833
Epoch 6/50
440/440 [==============================] - 4s 9ms/step - loss: 0.2314 - acc: 0.9159 - val_loss: 0.
1417 - val_acc: 1.0000
Epoch 7/50
440/440 [==============================] - 4s 9ms/step - loss: 0.2138 - acc: 0.9216 - val_loss: 0.
1374 - val_acc: 0.9667
Epoch 8/50
440/440 [==============================] - 4s 9ms/step - loss: 0.2067 - acc: 0.9256 - val_loss: 0.
0961 - val_acc: 1.0000
Epoch 9/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1877 - acc: 0.9318 - val_loss: 0.
1275 - val_acc: 0.9583
Epoch 10/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1751 - acc: 0.9420 - val_loss: 0.
0895 - val_acc: 1.0000
Epoch 11/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1525 - acc: 0.9506 - val_loss: 0.
1130 - val_acc: 0.9750
Epoch 12/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1458 - acc: 0.9477 - val_loss: 0.
0900 - val_acc: 0.9917
Epoch 13/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1320 - acc: 0.9580 - val_loss: 0.
0520 - val_acc: 1.0000
Epoch 14/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1181 - acc: 0.9631 - val_loss: 0.
0496 - val_acc: 1.0000
Epoch 15/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1224 - acc: 0.9540 - val_loss: 0.
0501 - val_acc: 1.0000
Epoch 16/50
440/440 [==============================] - 4s 9ms/step - loss: 0.1028 - acc: 0.9653 - val_loss: 0.
0519 - val_acc: 0.9917
Epoch 17/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0942 - acc: 0.9699 - val_loss: 0.
0410 - val_acc: 0.9917
Epoch 18/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0906 - acc: 0.9710 - val_loss: 0.
0396 - val_acc: 0.9917
Epoch 19/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0897 - acc: 0.9687 - val_loss: 0.
0471 - val_acc: 0.9833
Epoch 20/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0794 - acc: 0.9778 - val_loss: 0.
0438 - val_acc: 0.9833
Epoch 21/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0772 - acc: 0.9756 - val_loss: 0.
0339 - val_acc: 0.9833
Epoch 22/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0686 - acc: 0.9750 - val_loss: 0.
0283 - val_acc: 1.0000
```

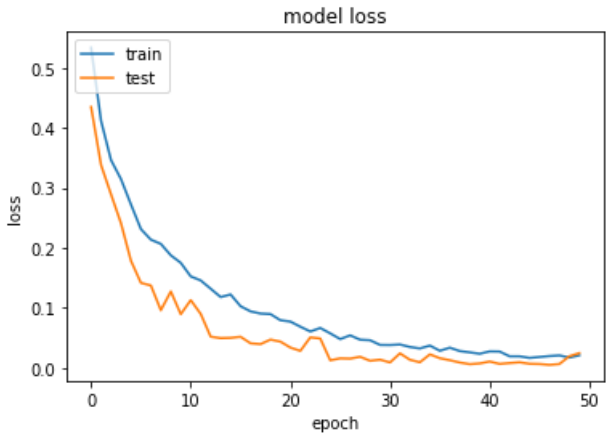
```
Epoch 23/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0607 - acc: 0.9835 - val_loss: 0.
0509 - val_acc: 0.9917
Epoch 24/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0665 - acc: 0.9784 - val_loss: 0.
0490 - val_acc: 0.9750
Epoch 25/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0575 - acc: 0.9858 - val_loss: 0.
0126 - val_acc: 1.0000
Epoch 26/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0480 - acc: 0.9852 - val_loss: 0.
0159 - val_acc: 1.0000
Epoch 27/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0542 - acc: 0.9801 - val_loss: 0.
0155 - val_acc: 1.0000
Epoch 28/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0470 - acc: 0.9830 - val_loss: 0.
0185 - val_acc: 1.0000
Epoch 29/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0461 - acc: 0.9881 - val_loss: 0.
0118 - val_acc: 1.0000
Epoch 30/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0383 - acc: 0.9875 - val_loss: 0.
0137 - val_acc: 1.0000
Epoch 31/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0381 - acc: 0.9909 - val_loss: 0.
0091 - val_acc: 1.0000
Epoch 32/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0390 - acc: 0.9869 - val_loss: 0.
0242 - val_acc: 0.9833
Epoch 33/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0348 - acc: 0.9898 - val_loss: 0.
0135 - val_acc: 1.0000
Epoch 34/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0324 - acc: 0.9898 - val_loss: 0.
0093 - val_acc: 1.0000
Epoch 35/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0372 - acc: 0.9886 - val_loss: 0.
0226 - val_acc: 0.9917
Epoch 36/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0285 - acc: 0.9881 - val_loss: 0.
0161 - val_acc: 1.0000
Epoch 37/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0337 - acc: 0.9875 - val_loss: 0.
0128 - val_acc: 1.0000
Epoch 38/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0280 - acc: 0.9903 - val_loss: 0.
0090 - val_acc: 1.0000
Epoch 39/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0259 - acc: 0.9920 - val_loss: 0.
0062 - val_acc: 1.0000
Epoch 40/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0236 - acc: 0.9938 - val_loss: 0.
0074 - val_acc: 1.0000
Epoch 41/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0275 - acc: 0.9909 - val_loss: 0.
0106 - val_acc: 1.0000
Epoch 42/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0273 - acc: 0.9903 - val_loss: 0.
0066 - val_acc: 1.0000
Epoch 43/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0193 - acc: 0.9955 - val_loss: 0.
0081 - val_acc: 1.0000
Epoch 44/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0193 - acc: 0.9932 - val_loss: 0.
0093 - val_acc: 1.0000
Epoch 45/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0167 - acc: 0.9955 - val_loss: 0.
0068 - val_acc: 1.0000
Epoch 46/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0182 - acc: 0.9960 - val_loss: 0.
0064 - val_acc: 1.0000
Epoch 47/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0197 - acc: 0.9955 - val_loss: 0.
0050 - val_acc: 1.0000
Epoch 48/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0207 - acc: 0.9915 - val_loss: 0.
```

```
0064 - val_acc: 1.0000
Epoch 49/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0172 - acc: 0.9949 - val_loss: 0.
0194 - val_acc: 1.0000
Epoch 50/50
440/440 [==============================] - 4s 9ms/step - loss: 0.0210 - acc: 0.9932 - val_loss: 0.
0246 - val_acc: 0.9833
```

In [0]:

```
#  "Accuracy"
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Summary

- A dataset was provided in the **Input** directory where there are 440 images provided for training and 30 images for testing.
- For building a CNN image classifier on top of the dataset, transfer learning was applied using **VGG16** architecture as its was proven to yeild high accuracies in image classification.
- Perfomance of the model was found to be impressive on both train and test sets.
- Model performance on test set was found to be more when compared to training set, which is a pretty impressive case.