Scene Preidictor in Autonomous Vehicle

Objective : Predict scene based on vision sensor data.
Classify whether vehicle is driving in “highway”, “city” or “residential”.

Labeled data from Berkeley Dataset : **https://bdd-data.berkeley.edu/**

```
"labels": [
          {
             "category": "traffic light",
             "attributes": {
                "occluded": false,
                "truncated": false,
                "trafficLightColor": "green"
             },

         {
             "category": "car",
             "attributes": {
                "occluded": false,
                "truncated": false,
                "trafficLightColor": "none"
             },

        "category": "drivable area",
             "attributes": {
                "areaType": "direct"
             },
```

Word Embedding:
Tried different embedding algorithm (count, freq, tfids)
Went with tfidf token vectorizer. Need to try **Glove, FastText & Word2Vec.**

```
# define Tokenizer with Vocab Size
tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(x_train)
#x_train = tokenizer.texts_to_matrix(x_train, mode='count') #got low accuracy
with mode='count'
#x_test = tokenizer.texts_to_matrix(x_test, mode='count')
#x_train = tokenizer.texts_to_matrix(x_train, mode='freq')
#x_test = tokenizer.texts_to_matrix(x_test, mode='freq')
x_train = tokenizer.texts_to_matrix(x_train, mode='tfidf')
x_test = tokenizer.texts_to_matrix(x_test, mode='tfidf')
```

Model:
DNN for classification (Need to try LSTM, CNN)

```
model = Sequential()
model.add(Dense(512, input_shape=(vocab_size,)))
model.add(Activation('relu')) #relu better than sigmoid
model.add(Dropout(0.3))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
#model.compile(loss='Poisson', optimizer='adam', metrics=['accuracy'])
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy']) #categorical_crossentropy slightly better performance
history = model.fit(x_train, y_train,
batch_size=batch_size,
epochs=epochs,
verbose=1,
validation_split=0.1)

score = model.evaluate(x_test, y_test, batch_size=batch_size, verbose=1)
```

Accuracy:

```
Epoch 1/5 1/1 [==============================] - 0s 156ms/step - loss: 1.0898
- accuracy: 0.3333 - val_loss: 1.0745 - val_accuracy: 0.3333 Epoch 2/5 1/1
[==============================] - 0s 22ms/step - loss: 0.9850 - accuracy:
0.7143 - val_loss: 1.0607 - val_accuracy: 0.3333 Epoch 3/5 1/1
[==============================] - 0s 22ms/step - loss: 0.9130 - accuracy:
0.7143 - val_loss: 1.0643 - val_accuracy: 0.3333 Epoch 4/5 1/1
[==============================] - 0s 23ms/step - loss: 0.8369 - accuracy:
0.7143 - val_loss: 1.0928 - val_accuracy: 0.3333 Epoch 5/5 1/1
[==============================] - 0s 24ms/step - loss: 0.7191 - accuracy:
0.7143 - val_loss: 1.1539 - val_accuracy: 0.3333 1/1
[==============================] - 0s 1ms/step - loss: 0.7328 - accuracy:
0.7143 Test loss: 0.7327927350997925 Test accuracy: 0.7142857313156128
```

Obstacles:

Dataset was in JSON with size > 1.5GB. Couldn’t load the data in python.

Unexpected Problems:
Accuracy never went above 70%.

Interesting thing about this Project:

Autonomous vehicle can detect objects and their location but they are unaware of the scene and overall understanding of the surroundings.

Another aspect of the project was to bridge the Perception AI with a conversational AI.

Uses cases: Conversational AI, Text mining, Prediction.