Hello, I am Fu-sung Kim-Benjamin Tang (唐甫頌) with the student ID 0845058 and this is my homework.

Introduction

I used Keras to implement my convolutional neural network and cv2 to read the image data from the dataset. I wrote this code with help of several tech blogs and the keras documentation. I constructed my CNN architecture mostly with help of this blog: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html (https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

The code below is written in accordance to the Python-PEP8 style convention as required: https://realpython.com/python-pep8/ (https://realpython.com/python-pep8/).

Methodology

Data Pre-processing

I used the module cv2 to read the images and then to resize them to a 255x255 format, since not all images had the same format, but a uniform format should help to have a higher performance during the classification process.

After that the images are being processed via data augmentation with the keras ImageDataGenerator. As hyperparameters I enabled horizontal flipping, featurewise_std_normalization, featurewise center, a small rotation degree and small width as well as height shifts. By using this data generator, the initial training set can be increased in size via data augmentation (flipping, shifting, rotating the images a bit). This helps the model to train with more data and makes it also more robust, because it learns to recognize images correctly, even if they are slightly distorted. So by doing this, overfitting is also reduced.

After that the dimensions of the data is adjusted and because the neural network needs values from 0 to 1 as input, the greyscale values from 0 to 255 are mapped to 0 to 1 values (for each pixel).

Furthermore because the CNN can also not use strings as class names, I hot encoded the labels instead.

As parameters for batch size and epochs I used different settings via CVgridsearch but the current one with 32 batch size and 20 epochs turned out to be the best one.

Model architecture

The CNN architecture consists of three Conv2D layers and the first one receives the input of 255x255 image data in form of arrays and with pixel values between 0 and 1 (mapped from greyscale). Each Conv layer is then followed by a rectified linear unit (relu) layer, a max pooling layer and then a dropout layer. This setup helps to downsample and then to avoid overfitting by dismissing some data (dropout). At the end there is flatten and dense layer followed by another relu and dropout and finally the last dense layer will reduce the output to thirteen nodes as specified, so that each class is represented by a node.

Hyperparameters

Keras Image data generator:

- featurewise_center=False
- featurewise_std_normalization=False
- rotation_range=10
- width_shift_range=0.1
- height_shift_range=0.1
- horizontal_flip=True
- vertical_flip=True

Convolutional neural network:

- batch_size=32
- epochs=20
- used activation=linear
- dropout=0.25
- otimizer=adam

Findings and summary

Throughout the process of working on this assignment and trying to optimize and improve the model, I learned that more learning time in form of epochs or more additional layers actually don't necessarily improve the performance. It turns out that there is a point until the model can improve with given architecture and hyperparameter tuning, but after that point continued learning will actually degrade the performance. This can also be seen in the literature and to avoid this, early stopping, so stopping the model from learning when the performance starts to decrease, is implemented.
Furthermore another finding is that ensembles, so several classifiers/networks that are coupled together for thec classification process, are not necessarily improving the performance either. I used three different networks at some point with different configurations with bagging, so votes that were averaged a the end and the overall performance was worse than the performance of a single network such as this one alone.

Gitlab link: https://github.com/kimbold/VRDL_2019 (https://github.com/kimbold/VRDL_2019)

Install git and clone the repository with the dataset

In [1]:

```
pip install python-git
```

```
Collecting python-git
  Downloading https://files.pythonhosted.org/packages/f0/6f/664d1dce126168f4fb91e74e8
f7bd26db72f0b2a49b0fa1c9f9742daf1ca/python_git-2018.2.1-py3-none-any.whl (https://fil
es.pythonhosted.org/packages/f0/6f/664d1dce126168f4fb91e74e8f7bd26db72f0b2a49b0fa1c9f
9742daf1ca/python_git-2018.2.1-py3-none-any.whl)
Requirement already satisfied: send2trash in /usr/local/lib/python3.6/dist-packages
 (from python-git) (1.5.0)
Installing collected packages: python-git
Successfully installed python-git-2018.2.1
```

Get the dataset by cloning my repository from git (this is how Google Colab gets the data)

In [2]:

```
!git clone https://github.com/kimbold/VRDL_2019 #https://github.com/fastai/courses.git
```

```
Cloning into 'VRDL_2019'...
remote: Enumerating objects: 3862, done.
remote: Counting objects: 100% (3862/3862), done.
remote: Compressing objects: 100% (3860/3860), done.
remote: Total 3862 (delta 0), reused 3859 (delta 0), pack-reused 0
Receiving objects: 100% (3862/3862), 51.62 MiB | 57.90 MiB/s, done.
```

Import modules

In [3]:

```
import os
import cv2
import numpy as np
from numpy import array
import pandas as pd
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.resnet50 import preprocess_input
from keras.utils import to_categorical
from matplotlib.pyplot import imshow
import matplotlib.pyplot as plt
from PIL import Image

# Import modules for Convolutional Neural Network
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
```

```
Using TensorFlow backend.
```

Load the training and testing data from the folders for each class respectively

In [4]:

```
x_train = []
x_test = []
testing_data = []
training_data = []
training_labels = []
training_data_per_class = {}
flipped_training = []
train_path = 'VRDL_2019/dataset/train/'
string_class_names = os.listdir(train_path)
class_names = [x for x in range(13)]


for folder in class_names:
    print("Accessing: "+string_class_names[folder])
    training_data_per_class[folder] = os.listdir('VRDL_2019/dataset/train/'+string_clas


# Configure path for testing data
test_path = 'VRDL_2019/dataset/test/'
test_batch = os.listdir(test_path)


# Training data and labels through all classes
array_counter=0
for class_name in class_names:
    print("Loading train:"+string_class_names[class_name])
    for item in training_data_per_class[class_name]:
        training_labels.append(class_name)
        img_path = 'VRDL_2019/dataset/train/'+string_class_names[class_name]+"/"+item
        x = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        x = cv2.resize(x,(256,256))
        x = preprocess_input(x)
        training_data.append(x)


for pic in training_data:
    mirror=np.fliplr(pic)
    flipped_training.append(mirror)

training_data.extend(flipped_training)
training_labels.extend(training_labels)


# Testing data
for sample in test_batch:
    img_path = test_path+sample
    x = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    x = cv2.resize(x,(256,256))
    x = preprocess_input(x)
    x_test.append(x)


# Finally converting list into numpy array
for item in x_test:
    testing_data.append(np.array(item))
```

```
Accessing: highway
Accessing: forest
Accessing: office
Accessing: kitchen
Accessing: livingroom
```

```
Accessing: opencountry
Accessing: street
Accessing: bedroom
Accessing: insidecity
Accessing: mountain
Accessing: coast
Accessing: tallbuilding
Accessing: suburb
Loading train:highway
Loading train:forest
Loading train:office
Loading train:kitchen
Loading train:livingroom
Loading train:opencountry
Loading train:street
Loading train:bedroom
Loading train:insidecity
Loading train:mountain
Loading train:coast
Loading train:tallbuilding
Loading train:suburb
```

Check how a stored image looks like

In [5]:

```python
id = 0
print(training_data[id].shape)
plt.imshow(training_data[id], cmap="gray")
plt.show()
```

```
(256, 256)
```

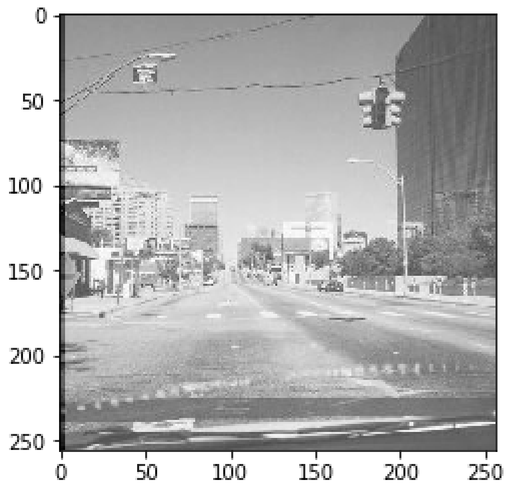


Image data generator from Keras

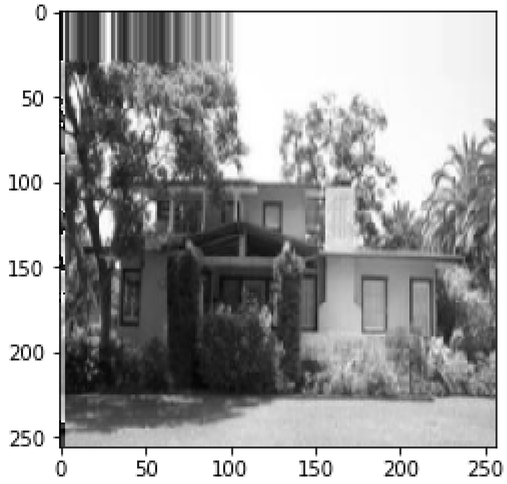
In [7]:

```
limit = int(len(training_data)/2)

datagen = ImageDataGenerator(
    featurewise_center=False,
    featurewise_std_normalization=False,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True)


for i in range(limit):
    expanded = np.expand_dims(training_data[i],0)
    expanded = np.expand_dims(expanded,0)
    aug_iter = datagen.flow(expanded)
    aug_images = [next(aug_iter)[0].astype(np.uint8) for i in range(5)]


    for x in range(len(aug_images)):
        aug_images[x] = aug_images[x][0, :, :]
        training_data.append(aug_images[x])
        training_labels.append(training_labels[i])
```

```
/usr/local/lib/python3.6/dist-packages/keras_preprocessing/image/numpy_array_iterato
r.py:127: UserWarning: NumpyArrayIterator is set to use the data format convention "c
hannels_last" (channels on axis 3), i.e. expected either 1, 3, or 4 channels on axis
3. However, it was passed an array with shape (1, 1, 256, 256) (256 channels).
  str(self.x.shape[channels_axis]) + ' channels).')
/usr/local/lib/python3.6/dist-packages/keras_preprocessing/image/image_data_generato
r.py:716: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but it
hasn't been fit on any training data. Fit it first by calling `.fit(numpy_data)`.
  warnings.warn('This ImageDataGenerator specifies '
/usr/local/lib/python3.6/dist-packages/keras_preprocessing/image/image_data_generato
r.py:724: UserWarning: This ImageDataGenerator specifies `featurewise_std_normalizati
on`, but it hasn't been fit on any training data. Fit it first by calling `.fit(numpy
_data)`.
  warnings.warn('This ImageDataGenerator specifies '
```

In [8]:

```
plt.imshow(aug_images[4], cmap="gray")
plt.show()
```



Inspect image dimensions

In [9]:

```
training_data_array = array(training_data)
testing_data_array = array(testing_data)
training_labels_array = array(training_labels)

print('Training data shape : ', training_data_array.shape, training_labels_array.shape)
print('Testing data shape : ', testing_data_array.shape)
```

```
Training data shape :  (19733, 256, 256) (19733,)
Testing data shape :  (1040, 256, 256)
```

Data preprocessing Reshape data so that it can be used as input for the CNN (e.g. mapping from greyscale to 0-1 ranged values)

In [10]:

```
reshaped_training_data_array = training_data_array.reshape(-1, 256,256, 1)
reshaped_testing_data_array = testing_data_array.reshape(-1,256,256,1)

print(reshaped_training_data_array.shape, reshaped_testing_data_array.shape)
```

Out[10]:

```
((19733, 256, 256, 1), (1040, 256, 256, 1))
```

In [0]:

```
training_data_processed = reshaped_training_data_array.astype('float32')
testing_data_processed = reshaped_testing_data_array.astype('float32')
training_data_processed = reshaped_training_data_array / 255.
testing_data_processed = reshaped_testing_data_array / 255.
```

Hot encoding for labels to change labels from categorical to one-hot encoding

In [0]:

```
train_Y_one_hot = to_categorical(training_labels)
```

Construct model architecture with layers and hyperparameters

In [0]:

```
batch_size = 64
epochs = 20
num_classes = 13
```

In [15]:

```
landscape_model = Sequential()

landscape_model.add(Conv2D(32, kernel_size=(5, 5),activation='linear',input_shape=(256,
landscape_model.add(LeakyReLU(alpha=0.1))
landscape_model.add(MaxPooling2D((2, 2),padding='same'))
landscape_model.add(Dropout(0.25))

landscape_model.add(Conv2D(64, (5, 5), activation='linear',padding='same'))
landscape_model.add(LeakyReLU(alpha=0.1))
landscape_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
landscape_model.add(Dropout(0.25))

landscape_model.add(Conv2D(128, (5, 5), activation='linear',padding='same'))
landscape_model.add(LeakyReLU(alpha=0.1))
landscape_model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
landscape_model.add(Dropout(0.25))

landscape_model.add(Flatten())
landscape_model.add(Dense(128, activation='linear'))
landscape_model.add(LeakyReLU(alpha=0.1))
landscape_model.add(Dropout(0.25))

landscape_model.add(Dense(num_classes, activation='softmax'))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:66: The name tf.get_default_graph is deprecated. Please use tf.compat.v
1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:541: The name tf.placeholder is deprecated. Please use tf.compat.v1.pla
ceholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:4432: The name tf.random_uniform is deprecated. Please use tf.random.un
iform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:4267: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2
d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:148: The name tf.placeholder_with_default is deprecated. Please use tf.
compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:3733: calling dropout (from tensorflow.python.ops.nn_ops) with keep_pro
b 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
`.
```

In [16]:

```
landscape_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.opt
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:79
3: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer
instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl
ow_backend.py:3576: The name tf.log is deprecated. Please use tf.math.log instead.
```

In [17]:

```
landscape_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      832       
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 256, 256, 32)      0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 128, 128, 32)      0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 64)      51264     
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128, 128, 64)      0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 64)        0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 64, 64, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 128)       204928    
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 64, 64, 128)       0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 32, 32, 128)       0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 32, 32, 128)       0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 131072)            0         
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16777344  
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 128)               0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 13)                1677      
=================================================================
Total params: 17,036,045
Trainable params: 17,036,045
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
landscape_model.fit(training_data_processed, train_Y_one_hot, batch_size=batch_size,epo
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/
math_grad.py:1250: 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
Train on 15786 samples, validate on 3947 samples
Epoch 1/20
15786/15786 [==============================] - 114s 7ms/step - loss: 1.7143 - acc: 0.
4100 - val_loss: 3.6396 - val_acc: 0.1112
Epoch 2/20
15786/15786 [==============================] - 105s 7ms/step - loss: 0.8231 - acc: 0.
6997 - val_loss: 4.2947 - val_acc: 0.1467
Epoch 3/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.5037 - acc: 0.
8235 - val_loss: 5.1242 - val_acc: 0.1999
Epoch 4/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.2710 - acc: 0.
9062 - val_loss: 6.3092 - val_acc: 0.1819
Epoch 5/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.1690 - acc: 0.
9398 - val_loss: 6.2353 - val_acc: 0.2567
Epoch 6/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.1113 - acc: 0.
9606 - val_loss: 7.0222 - val_acc: 0.2341
Epoch 7/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.1238 - acc: 0.
9583 - val_loss: 6.8336 - val_acc: 0.2589
Epoch 8/20
15786/15786 [==============================] - 104s 7ms/step - loss: 0.1582 - acc: 0.
9490 - val_loss: 9.1892 - val_acc: 0.1834
Epoch 9/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.1246 - acc: 0.
9580 - val_loss: 7.0604 - val_acc: 0.2653
Epoch 10/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0799 - acc: 0.
9752 - val_loss: 8.5180 - val_acc: 0.2280
Epoch 11/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0618 - acc: 0.
9796 - val_loss: 7.2064 - val_acc: 0.2787
Epoch 12/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0605 - acc: 0.
9821 - val_loss: 8.4003 - val_acc: 0.2174
Epoch 13/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.1784 - acc: 0.
9481 - val_loss: 9.2713 - val_acc: 0.2100
Epoch 14/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0755 - acc: 0.
9769 - val_loss: 7.8594 - val_acc: 0.2792
Epoch 15/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0509 - acc: 0.
9845 - val_loss: 8.5678 - val_acc: 0.2359
Epoch 16/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0635 - acc: 0.
9809 - val_loss: 9.8236 - val_acc: 0.2265
Epoch 17/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0821 - acc: 0.
9769 - val_loss: 10.3927 - val_acc: 0.2085
Epoch 18/20
```

```
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0829 - acc: 0.
9788 - val_loss: 8.6891 - val_acc: 0.2437
Epoch 19/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0454 - acc: 0.
9867 - val_loss: 10.4618 - val_acc: 0.2118
Epoch 20/20
15786/15786 [==============================] - 103s 7ms/step - loss: 0.0512 - acc: 0.
9857 - val_loss: 10.0442 - val_acc: 0.2080
```

Out[18]:

```
<keras.callbacks.History at 0x7fa394e16898>
```

Predict labels from the test dataset

In [0]:

```python
predicted_classes = landscape_model.predict(testing_data_processed)
predicted_class_max = np.argmax(np.round(predicted_classes),axis=1)
test_filenames = []
predictions_as_labels = []


for item in predicted_class_max:
    predictions_as_labels.append(string_class_names[item])


for item in test_batch:
    x = item
    x=os.path.splitext(item)[0]
    test_filenames.append(x)
```

In [20]:

```python
columns = ['id']
data = ['label']

columns.extend(test_filenames)
data.extend(predictions_as_labels)

df = pd.DataFrame(index=['id','label'], data=[test_filenames, predictions_as_labels]).
df1 = df[['id', 'label']]

df1.to_csv('Predictions.csv', index=False)
display(df1)
```

	id	label
0	image_0385	coast
1	image_0629	suburb
2	image_1024	tallbuilding
3	image_0691	office
4	image_0328	bedroom
5	image_0182	coast
6	image_0002	suburb
7	image_0718	coast
8	image_0380	coast
9	image_0760	office
10	image_0975	insidecity
11	image_0129	forest
12	image_0800	tallbuilding
13	image_0805	highway
14	image_0603	opencountry
15	image_0038	forest
16	image_0885	suburb
17	image_0533	mountain
18	image_0748	opencountry
19	image_0768	tallbuilding
20	image_0737	insidecity
21	image_0344	suburb
22	image_0366	tallbuilding
23	image_1003	bedroom
24	image_0598	opencountry
25	image_0510	street
26	image_0813	street
27	image_0868	office
28	image_0701	street
29	image_0693	coast
...	...	...

	id	label
1010	image_0495	opencountry
1011	image_1017	insidecity
1012	image_0605	insidecity
1013	image_0238	livingroom
1014	image_0563	highway
1015	image_0232	forest
1016	image_0706	opencountry
1017	image_0635	livingroom
1018	image_0229	suburb
1019	image_0134	insidecity
1020	image_0450	forest
1021	image_0372	livingroom
1022	image_0213	livingroom
1023	image_0825	street
1024	image_0878	mountain
1025	image_0936	livingroom
1026	image_0455	office
1027	image_0680	kitchen
1028	image_0139	forest
1029	image_0757	opencountry
1030	image_0850	highway
1031	image_0872	tallbuilding
1032	image_0310	tallbuilding
1033	image_0442	bedroom
1034	image_0008	opencountry
1035	image_0319	highway
1036	image_0539	highway
1037	image_0255	highway
1038	image_0512	opencountry
1039	image_0207	livingroom

1040 rows × 2 columns