Kaggle Project Code

Wenxin (Hugo) Xue

SID: 487011

Log in Google Colab

```
try:
    from google.colab import drive
    drive.mount('/content/drive', force_remount=True)
    COLAB = True
    print("Note: using Google CoLab")
    %tensorflow_version 2.x
except:
    print("Note: not using Google CoLab")
    COLAB = False
```

Deploy GPU

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)
```

```
Mon Nov 29 17:28:49 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   36C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
```

```
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
from psutil import virtual_memory
ram_gb = virtual_memory().total / 1e9
print('Your runtime has {:.1f} gigabytes of available RAM\n'.format(ram_gb))

if ram_gb < 20:
  print('Not using a high-RAM runtime')
else:
  print('You are using a high-RAM runtime!')
```

```
Your runtime has 27.3 gigabytes of available RAM

You are using a high-RAM runtime!
```

import required packages

```
import sys

import tensorflow.keras
import pandas as pd
import sklearn as sk
import tensorflow as tf
import numpy as np
import os
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.resnet_v2 import ResNet152V2
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications import NASNetMobile
from tensorflow.keras.applications import ResNet50V2, ResNet101V2
from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications import DenseNet121

from tensorflow.keras.layers import Input, Dropout, Reshape
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.layers import LeakyReLU, PReLU
from tensorflow.keras.layers import UpSampling2D, Conv2D, MaxPooling2D
from tensorflow.keras.layers import Flatten, BatchNormalization
from tensorflow.keras.layers import Activation, ZeroPadding2D
from tensorflow.keras.models import Sequential, Model, load_model, save_model
from tensorflow.keras.initializers import he_normal
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
```

Load dataset

```
# file='/content/drive/My Drive/Colab Notebooks/assignment_yourname_class1.ipynb'
PATH = "/content/drive/My Drive/Colab Notebooks/applications-of-deep-learning-wustlfall-20
PATH_TRAIN = os.path.join(PATH, "train.csv")
PATH_TEST = os.path.join(PATH, "test.csv")


df_train = pd.read_csv(PATH_TRAIN)
df_test = pd.read_csv(PATH_TEST)

df_train['filename'] = df_train.id.astype(str) + ".jpg"
df_test['filename'] = df_test.id.astype(str) + ".jpg"
```

1. use train_test_split to sampling dataset

```
WIDTH = 150
HEIGHT = 150

#·Split·into·train/test
x_train,·x_test,·y_train,·y_test·=·train_test_split(····
····df_train['filename'].values, df_train['sqft'].values, test_size=0.1)

df_train_cut = pd.DataFrame({"filename":x_train,"sqft":y_train})
df_validate_cut = pd.DataFrame({"filename":x_test,"sqft":y_test})
print(f"Training shape: {df_train_cut.shape}")
print(f"Validate shape: {df_validate_cut.shape}")

training_datagen = ImageDataGenerator(
  rescale = 1./255,
  horizontal_flip=True,
  # vertical_flip=True,
  fill_mode='nearest')

train_generator = training_datagen.flow_from_dataframe(
        dataframe=df_train_cut,
        directory=PATH,
        x_col="filename",
        y_col="sqft",
        target_size=(HEIGHT, WIDTH),
        batch_size=32, # Keeping the training batch size small USUALLY increases performan
        class_mode='raw')

validation_datagen = ImageDataGenerator(rescale = 1./255)

val_generator = validation_datagen.flow_from_dataframe(
        dataframe=df_validate_cut,
        directory=PATH,
```

```
        x_col="filename",
        y_col="sqft",
        target_size=(HEIGHT, WIDTH),
        batch_size=256, # Make the validation batch size as large as you have memory for
        class_mode='raw')
```

```
Training shape: (21598, 2)
Validate shape: (2400, 2)
Found 21598 validated image filenames.
Found 2400 validated image filenames.
```

2. simply split train and test dataset by 9:1

```
TRAIN_PCT = 0.9
TRAIN_CUT = int(len(df_train) * TRAIN_PCT)

df_train_cut = df_train[0:TRAIN_CUT]
df_validate_cut = df_train[TRAIN_CUT:]

print(f"Training size: {len(df_train_cut)}")
print(f"Validate size: {len(df_validate_cut)}")

WIDTH = 150
HEIGHT = 150

training_datagen = ImageDataGenerator(
  rescale = 1./255,
  horizontal_flip=True,
  # vertical_flip=True,
  fill_mode='nearest')

train_generator = training_datagen.flow_from_dataframe(
        dataframe=df_train_cut,
        directory=PATH,
        x_col="filename",
        y_col="sqft",
        target_size=(HEIGHT, WIDTH),
        batch_size=32, # Keeping the training batch size small USUALLY increases performan
        class_mode='raw')

validation_datagen = ImageDataGenerator(rescale = 1./255)

val_generator = validation_datagen.flow_from_dataframe(
        dataframe=df_validate_cut,
        directory=PATH,
        x_col="filename",
        y_col="sqft",
        target_size=(HEIGHT, WIDTH),
        batch_size=256, # Make the validation batch size as large as you have memory for
        class_mode='raw')
```

```
Training size: 21598
Validate size: 2400
```

```
Found 21598 validated image filenames.
Found 2400 validated image filenames.
```

Try to build different types of model

```
# sequential model
# model = Sequential(
#     [
#      Conv2D(32, (3,3), activation='relu', input_shape=(HEIGHT, WIDTH, 3),padding="same")
#      MaxPooling2D(pool_size=(2,2)),
#      Conv2D(64, (3,3), activation='relu', padding="same"),
#      Conv2D(64, (3,3), activation='relu'),
#      MaxPooling2D(pool_size=(2,2)),
#      Conv2D(128, (3,3), activation='relu',padding="same"),
#      Conv2D(128, (3,3), activation='relu'),
#      MaxPooling2D(pool_size=(4,4)),
#      Flatten(),
#      Dense(1024, activation='relu'),
#      Dense(512, activation='relu'),
#      Dense(256, activation='relu'),
#      Dense(128, activation='relu'),
#      Dense(64, activation='relu'),
#      Dense(1, activation='linear')
#     ]
# )
# model.summary()
################################################################################
# Xception-1 rmse=>500
input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
base_model = Xception(
    include_top=False, weights='imagenet', input_tensor=input_tensor,
    classifier_activation='relu'
)

base_model.trainable=True
x=base_model.layers[-1].output
x=GlobalAveragePooling2D()(x)

x = Dense(1536,activation='relu')(x)
x = Dense(1536,activation='relu')(x)
x = Dense(728,activation='relu')(x)
x = Dense(728,activation='relu')(x)

output=Dense(1,activation='linear')(x)
model=Model(inputs=input_tensor,outputs=output)
model.summary()
################################################################################
# Xception-2
# input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
# base_model = Xception(
#     include_top=False, weights='imagenet', input_tensor=input_tensor,
#     classifier activation='relu'
```

```
# )

# base_model.trainable=True
# x=base_model.output
# x=GlobalAveragePooling2D()(x)

# x = Dense(1024,activation='relu')(x)
# x = Dense(1024,activation='relu')(x)
# x = Dense(512,activation='relu')(x)
# x = Dense(64,activation='relu')(x)

# output=Dense(1,activation='linear')(x)
# model=Model(inputs=input_tensor,outputs=output)
# model.summary()
################################################################################
# VGG16-2
# input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
# base_model = VGG16(
#     include_top=False, weights='imagenet', input_tensor=input_tensor,
#     classifier_activation='relu'
# )

# base_model.trainable=True
# x=base_model.output
# x=GlobalAveragePooling2D()(x)

# x = Dense(512,activation='relu')(x)
# x = Dense(512,activation='relu')(x)
# x = Dense(256,activation='relu')(x)
# x = Dense(256,activation='relu')(x)
# x = Dense(64,activation='relu')(x)

# output=Dense(1,activation='linear')(x)
# model=Model(inputs=input_tensor,outputs=output)
# model.summary()
#############################################################
# input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
# base_model = DenseNet121(
#     include_top=False, weights='imagenet', input_tensor=input_tensor
# )

# base_model.trainable=True
# x=base_model.output
# x=GlobalAveragePooling2D()(x)

# x = Dense(512,activation='relu')(x)
# x = Dense(512,activation='relu')(x)
# x = Dense(256,activation='relu')(x)
# x = Dense(256,activation='relu')(x)
# x = Dense(64,activation='relu')(x)
# x = Dense(64,activation='relu')(x)

# output=Dense(1,activation='linear')(x)
# model=Model(inputs=input_tensor,outputs=output)
# model.summary()
```

```
##########################################################
# input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
# base_model = ResNet152V2(
#     include_top=False, weights='imagenet', input_tensor=input_tensor,
#     classifier_activation='relu'
# )

# x=base_model.output
# x=GlobalAveragePooling2D()(x)

# x = Dense(1024,activation='relu')(x)
# x = Dense(1024,activation='relu')(x)
# # x = Dense(64,activation='relu')(x)

# output=Dense(1,activation='linear')(x)
# model=Model(inputs=input_tensor,outputs=output)
# model.summary()
##########################################################
# input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
# base_model = Xception(
#     include_top=False, weights=None, input_tensor=input_tensor,
#     classifier_activation='relu'
# )

# x=base_model.output
# x=GlobalAveragePooling2D()(x)

# x = Dense(1024,activation='relu')(x)
# x = Dense(1024,activation='relu')(x)

# output=Dense(1)(x)
# model=Model(inputs=input_tensor,outputs=output)
# model.summary()
```

```
block3_sepconv2_bn (BatchNorma  (None, 36, 36, 256)  1024       ['block3_sepcon
lization)

conv2d_1 (Conv2D)               (None, 18, 18, 256)  32768      ['add[0][0]']

block3_pool (MaxPooling2D)      (None, 18, 18, 256)  0          ['block3_sepcon

batch_normalization_1 (BatchNo  (None, 18, 18, 256)  1024       ['conv2d_1[0][0
rmalization)

add_1 (Add)                     (None, 18, 18, 256)  0          ['block3_pool[0
                                                                 'batch_normali

block4_sepconv1_act (Activatio  (None, 18, 18, 256)  0          ['add_1[0][0]']
n)

block4_sepconv1 (SeparableConv  (None, 18, 18, 728)  188672     ['block4_sepcon
2D)

block4_sepconv1_bn (BatchNorma  (None, 18, 18, 728)  2912       ['block4_sepcon

lization)
```

```
block4_sepconv2_act (Activatio  (None, 18, 18, 728)  0          ['block4_sepcon
n)

block4_sepconv2 (SeparableConv  (None, 18, 18, 728)  536536     ['block4_sepcon
2D)

block4_sepconv2_bn (BatchNorma  (None, 18, 18, 728)  2912       ['block4_sepcon
lization)

conv2d_2 (Conv2D)              (None, 9, 9, 728)    186368      ['add_1[0][0]']

block4_pool (MaxPooling2D)     (None, 9, 9, 728)    0           ['block4_sepcon

batch_normalization_2 (BatchNo  (None, 9, 9, 728)   2912        ['conv2d_2[0][0
rmalization)

add_2 (Add)                    (None, 9, 9, 728)    0           ['block4_pool[0
                                                                 'batch_normali

block5_sepconv1_act (Activatio  (None, 9, 9, 728)   0           ['add_2[0][0]']
n)

block5_sepconv1 (SeparableConv  (None, 9, 9, 728)   536536      ['block5_sepcon
2D)

block5_sepconv1_bn (BatchNorma  (None, 9, 9, 728)   2912        ['block5_sepcon
lization)

block5_sepconv2_act (Activatio  (None, 9, 9, 728)   0           ['block5_sepcon
n)

block5_sepconv2 (SeparableConv  (None, 9, 9, 728)   536536      ['block5_sepcon
2D)
```

Model Training

- metrics: rmse
- learning rate: 1e-4 or 1e-5
- Adam
- model_checkpoint: save best model
- lr_reduce: 1e-7

```
STEP_SIZE_VALID=val_generator.n//val_generator.batch_size

model.compile(loss = 'mean_squared_error', optimizer=Adam(learning_rate=1e-5), metrics=[Ro
model_checkpoint=ModelCheckpoint(os.path.join(PATH, 'xception_bestmodel.h5'),monitor="val_
lr_reduce = ReduceLROnPlateau(monitor='val_loss',factor=0.6,patience=50,verbose=1,mode='au
callback = [model_checkpoint,lr_reduce]
model.fit(train_generator, epochs=500, steps_per_epoch=250, validation_data = val_generato
          callbacks=callback, verbose = 1, validation_steps=STEP_SIZE_VALID)

model.load_weights(os.path.join(PATH, 'xception_bestmodel.h5'))
```

```
submit_datagen = ImageDataGenerator(rescale = 1./255)

submit_generator = submit_datagen.flow_from_dataframe(
        dataframe=df_test,
        directory=PATH,
        x_col="filename",
        batch_size = 1,
        shuffle = False,
        target_size=(HEIGHT, WIDTH),
        class_mode=None)

submit_generator.reset()
best_pred = model.predict(submit_generator,steps=len(df_test))
best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':best_pred[:,0].flatten()})
best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
```

```
250/250 [==============================] - 58s 231ms/step - loss: 256860.9219 -
Epoch 31/500
250/250 [==============================] - ETA: 0s - loss: 231863.5938 - rmse: 4
Epoch 00031: val_loss improved from 634265.75000 to 624533.43750, saving model t
250/250 [==============================] - 62s 246ms/step - loss: 231863.5938 -
Epoch 32/500
250/250 [==============================] - ETA: 0s - loss: 227243.1094 - rmse: 4
Epoch 00032: val_loss did not improve from 624533.43750
250/250 [==============================] - 58s 232ms/step - loss: 227243.1094 -
Epoch 33/500
250/250 [==============================] - ETA: 0s - loss: 221624.6406 - rmse: 4
Epoch 00033: val_loss improved from 624533.43750 to 616740.43750, saving model t
250/250 [==============================] - 62s 246ms/step - loss: 221624.6406 -
Epoch 34/500
250/250 [==============================] - ETA: 0s - loss: 217948.1562 - rmse: 4
Epoch 00034: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 231ms/step - loss: 217948.1562 -
Epoch 35/500
250/250 [==============================] - ETA: 0s - loss: 203022.8906 - rmse: 4
Epoch 00035: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 230ms/step - loss: 203022.8906 -
Epoch 36/500
250/250 [==============================] - ETA: 0s - loss: 201615.6406 - rmse: 4
Epoch 00036: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 230ms/step - loss: 201615.6406 -
Epoch 37/500
250/250 [==============================] - ETA: 0s - loss: 171739.6562 - rmse: 4
Epoch 00037: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 230ms/step - loss: 171739.6562 -
Epoch 38/500
250/250 [==============================] - ETA: 0s - loss: 193667.8594 - rmse: 4
Epoch 00038: val_loss did not improve from 616740.43750
250/250 [==============================] - 68s 270ms/step - loss: 193667.8594 -
Epoch 39/500
250/250 [==============================] - ETA: 0s - loss: 174699.3750 - rmse: 4
Epoch 00039: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 230ms/step - loss: 174699.3750 -
Epoch 40/500
250/250 [==============================] - ETA: 0s - loss: 161006.5000 - rmse: 4
Epoch 00040: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 230ms/step - loss: 161006.5000 -
Epoch 41/500
250/250 [==============================] - ETA: 0s - loss: 164168.9375 - rmse: 4
Epoch 00041: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 231ms/step - loss: 164168.9375 -
```

```
250/250 [==============================] - 98s 391ms/step - loss: 164108.9375
Epoch 42/500
250/250 [==============================] - ETA: 0s - loss: 159476.7500 - rmse: 3

Epoch 00042: val_loss did not improve from 616740.43750
250/250 [==============================] - 58s 230ms/step - loss: 159476.7500 -
Epoch 43/500
250/250 [==============================] - ETA: 0s - loss: 165272.0938 - rmse: 4
Epoch 00043: val_loss improved from 616740.43750 to 614250.12500, saving model t
250/250 [==============================] - 62s 247ms/step - loss: 165272.0938 -
Epoch 44/500
250/250 [==============================] - ETA: 0s - loss: 158949.3281 - rmse: 3
Epoch 00044: val_loss improved from 614250.12500 to 611622.68750, saving model t
250/250 [==============================] - 62s 247ms/step - loss: 158949.3281 -
Epoch 45/500
```

Build ensemble model: improving prediction accuracy

- resnet
- xception
- vgg

```
WIDTH = 150
HEIGHT = 150

def build_resnet():
  input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
  base_model = ResNet152V2(
      include_top=False, weights='imagenet', input_tensor=input_tensor,
      classifier_activation='relu'
  )

  x=base_model.output
  x=GlobalAveragePooling2D()(x)

  x = Dense(1024,activation='relu')(x)
  x = Dense(1024,activation='relu')(x)

  output=Dense(1,activation='linear')(x)
  model_resnet=Model(inputs=input_tensor,outputs=output)
  return model_resnet

def build_vgg():
  input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
  base_model = VGG16(
      include_top=False, weights='imagenet', input_tensor=input_tensor,
      classifier_activation='relu'
  )

  x=base_model.output
  x=GlobalAveragePooling2D()(x)

  x = Dense(1024,activation='relu')(x)
  x = Dense(1024,activation='relu')(x)
```

```
  output=Dense(1,activation='linear')(x)
  model_vgg=Model(inputs=input_tensor,outputs=output)
  return model_vgg

def build_xception():
  input_tensor = Input(shape=(HEIGHT, WIDTH, 3))
  base_model = Xception(
      include_top=False, weights='imagenet', input_tensor=input_tensor,
      classifier_activation='relu'
  )

  x=base_model.output
  x=GlobalAveragePooling2D()(x)

  x = Dense(1024,activation='relu')(x)
  x = Dense(1024,activation='relu')(x)

  output=Dense(1,activation='linear')(x)
  model_xception=Model(inputs=input_tensor,outputs=output)
  return model_xception
```

Ensemble Models

- kfold: try 5 or 10
- for every model, for every fold, train the model
- predict submit file for every model and every fold, calculate the average value

```
x = df_train["filename"].values
y = df_train["sqft"].values
x_submit = df_test["filename"].values

oos_y = []
oos_pred = []

models = [build_vgg(),build_xception()]
dataset_blend_train = np.zeros((x.shape[0], len(models)))
dataset_blend_test = np.zeros((x_submit.shape[0], len(models)))
submit_datagen = ImageDataGenerator(rescale = 1./255)
submit_generator = submit_datagen.flow_from_dataframe(
        dataframe=df_test,
        directory=PATH,
        x_col="filename",
        batch_size = 1,
        shuffle = False,
        target_size=(HEIGHT, WIDTH),
        class_mode=None)
submit_generator.reset()

kf = KFold(3)
folds = list(kf.split(x))
fold=0
```

```
    for j, model in enumerate(models):
      print("Model: {} : {}".format(j, model))
      fold_sums = np.zeros((x_submit.shape[0], len(folds)))
      total_rmse = 0
      for i, (train, test) in enumerate(folds):
        #training set
        train_cut = df_train.loc[train]
        #testing set
        val_cut = df_train.loc[test]

        training_datagen = ImageDataGenerator(
          rescale = 1./255,
          horizontal_flip=True,
          fill_mode='nearest')

        train_generator = training_datagen.flow_from_dataframe(
              dataframe= train_cut,
              directory=PATH,
              x_col="filename",
              y_col="sqft",
              target_size=(HEIGHT, WIDTH),
              batch_size=32,
              class_mode='raw')

        validation_datagen = ImageDataGenerator(rescale = 1./255)

        val_generator = validation_datagen.flow_from_dataframe(
              dataframe=val_cut,
              directory=PATH,
              x_col="filename",
              y_col="sqft",
              target_size=(HEIGHT, WIDTH),
              batch_size=256, # Make the validation batch size as large as you have memory for
              class_mode='raw')

        STEP_SIZE_VALID=val_generator.n//val_generator.batch_size

        model.compile(loss = 'mean_squared_error', optimizer=Adam(learning_rate=1e-5), metrics
        monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=20, verbose=1, mo
                restore_best_weights=True)
        # model_checkpoint1=ModelCheckpoint(
        #     os.path.join(PATH, 'blend{}_bestmodel.h5'.format(j)),
        #     monitor="val_loss",save_best_only=True,mode="min",verbose=1)
        # model_checkpoint2=ModelCheckpoint(os.path.join(PATH, 'resnet152_latestmodel.h5'),mon
        #                                   save_weights_only=True, verbose=0)
        lr_reduce = ReduceLROnPlateau(monitor='val_loss',factor=0.6,patience=20,verbose=1,mode
        callback = [monitor,model_checkpoint1,lr_reduce]
        model.fit(train_generator, epochs=50, steps_per_epoch=250, validation_data = val_gener
                  callbacks=callback, verbose = 1, validation_steps=STEP_SIZE_VALID)

        pred = model.predict(val_generator)
        dataset_blend_train[test, j] = pred[:, 0]
        pred2 = model.predict(submit_generator,steps=len(df_test))
        fold_sums[:, i] = pred2[:, 0]
```

```
        rmse = np.sqrt(mean_squared_error(val_cut['sqft'].values, pred[:,0]))
        total_rmse+=rmse
        print("Fold #{}: rmse={}".format(i,rmse))

      print("{}: Mean rmse={}".format(model.__class__.__name__,total_rmse/len(folds)))
      dataset_blend_test[:, j] = fold_sums.mean(1)

    print()
    print("Blending models.")

    from sklearn.ensemble import BaggingRegressor
    blend = BaggingRegressor(SVR())
    blend.fit(dataset_blend_train, y)
    submit_data = blend.predict(dataset_blend_test)
    best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':submit_data.flatten()})
    best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
    best_df_submit.head()
```

```
Found 1070 validated image filenames.
Model: 0 : <keras.engine.functional.Functional object at 0x7fb405f61a10>
Found 15998 validated image filenames.
Found 8000 validated image filenames.
Epoch 1/50
250/250 [==============================] - ETA: 0s - loss: 9884918.0000 - rmse: 3144
Epoch 00001: val_loss did not improve from 38344.63281
250/250 [==============================] - 80s 314ms/step - loss: 9884918.0000 - rmse
Epoch 2/50
250/250 [==============================] - ETA: 0s - loss: 2486400.5000 - rmse: 1576
Epoch 00002: val_loss did not improve from 38344.63281
250/250 [==============================] - 76s 305ms/step - loss: 2486400.5000 - rmse
Epoch 3/50
250/250 [==============================] - ETA: 0s - loss: 1426387.3750 - rmse: 1194
Epoch 00003: val_loss did not improve from 38344.63281
250/250 [==============================] - 76s 305ms/step - loss: 1426387.3750 - rmse
Epoch 4/50
250/250 [==============================] - ETA: 0s - loss: 1114439.3750 - rmse: 1055
Epoch 00004: val_loss did not improve from 38344.63281
250/250 [==============================] - 76s 305ms/step - loss: 1114439.3750 - rmse
Epoch 5/50
250/250 [==============================] - ETA: 0s - loss: 949823.8125 - rmse: 974.58
Epoch 00005: val_loss did not improve from 38344.63281
250/250 [==============================] - 76s 304ms/step - loss: 949823.8125 - rmse
Epoch 6/50
250/250 [==============================] - ETA: 0s - loss: 860329.1875 - rmse: 927.53
Epoch 00006: val_loss did not improve from 38344.63281
250/250 [==============================] - 76s 303ms/step - loss: 860329.1875 - rmse
Epoch 7/50
250/250 [==============================] - ETA: 0s - loss: 770234.5625 - rmse: 877.63
Epoch 00007: val_loss did not improve from 38344.63281
250/250 [==============================] - 76s 303ms/step - loss: 770234.5625 - rmse
Epoch 8/50
250/250 [==============================] - ETA: 0s - loss: 768421.0000 - rmse: 876.59
Epoch 00008: val_loss did not improve from 38344.63281
```

use regression model to predict submit file's result

```
Epoch 00009: val_loss did not improve from 38344.63281
```

```
from sklearn.linear_model import SGDRegressor
blend = SGDRegressor()
blend.fit(dataset_blend_train, y)
submit_data = blend.predict(dataset_blend_test)
best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':submit_data.flatten()})
best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
```

```
from sklearn.svm import SVR
blend = SVR()
blend.fit(dataset_blend_train, y)
submit_data = blend.predict(dataset_blend_test)
best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':submit_data.flatten()})
best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
best_df_submit.head()
```

	id	sqft
0	24000	5822.377037
1	24001	5835.473316
2	24002	5834.567900

```
from sklearn.neighbors import KNeighborsRegressor
blend = KNeighborsRegressor(n_neighbors=5)
blend.fit(dataset_blend_train, y)
submit_data = blend.predict(dataset_blend_test)
best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':submit_data.flatten()})
best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
best_df_submit.head()
```

	id	sqft
0	24000	4675.2
1	24001	6828.4
2	24002	5757.0
3	24003	4643.0
4	24004	5999.6

```
from sklearn.ensemble import BaggingRegressor
blend = BaggingRegressor(SVR())
blend.fit(dataset_blend_train, y)
submit_data = blend.predict(dataset_blend_test)
best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':submit_data.flatten()})
best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
best_df_submit.head()
```

	id	sqft
0	24000	5835.957249
1	24001	5845.931830
2	24002	5830.816485
3	24003	5830.907638
4	24004	5837.341104

```
from sklearn import tree
blend = tree.DecisionTreeRegressor()
blend.fit(dataset_blend_train, y)
submit_data = blend.predict(dataset_blend_test)
best_df_submit = pd.DataFrame({"id":df_test['id'],'sqft':submit_data.flatten()})
best_df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learni
best_df_submit.head()
```

	id	sqft
0	24000	5746.0
1	24001	7105.0
2	24002	7947.0
3	24003	6026.0
4	24004	4804.0

```
# from sklearn.model_selection import KFold
# from sklearn.metrics import mean_squared_error

# x = df_train["filename"].values
# oos_y = []
# oos_pred = []

# kf = KFold(10)
# fold=0
# for train, test in kf.split(x):
#   fold+=1
#   print(f"fold #{fold}")

#   train_cut = df_train.loc[train]
#   val_cut = df_train.loc[test]

#   training_datagen = ImageDataGenerator(
#     rescale = 1./255,
#     horizontal_flip=True,
#     fill_mode='nearest')

#   train_generator = training_datagen.flow_from_dataframe(
#         dataframe= train_cut,
#         directory=PATH,
#         x_col="filename",
#         y_col="sqft",
#         target_size=(150, 150),
#         batch_size=32,
#         class_mode='raw')

#   validation_datagen = ImageDataGenerator(rescale = 1./255)

#   val_generator = validation_datagen.flow_from_dataframe(
#         dataframe=val_cut,
#         directory=PATH,
#         x_col="filename",
#         y_col="sqft",
#         target_size=(150, 150),
#         batch_size=256, # Make the validation batch size as large as you have memory for
#         class_mode='raw')

#   STEP_SIZE_VALID=val_generator.n//val_generator.batch_size

#   model.compile(loss = 'mean_squared_error', optimizer=Adam(learning_rate=1e-5), metrics
```

```
#   monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=20, verbose=1, mo
#           restore_best_weights=True)
#   model_checkpoint1=ModelCheckpoint(os.path.join(PATH, 'resnet152_bestmodel.h5'),monitor
#   # model_checkpoint2=ModelCheckpoint(os.path.join(PATH, 'resnet152_latestmodel.h5'),mon
#   #                                     save_weights_only=True, verbose=0)
#   lr_reduce = ReduceLROnPlateau(monitor='val_loss',factor=0.6,patience=20,verbose=1,mode
#   callback = [monitor,model_checkpoint1,model_checkpoint2,lr_reduce]
#   model.fit(train_generator, epochs=100, steps_per_epoch=250, validation_data = val_gene
#             callbacks=callback, verbose = 1, validation_steps=STEP_SIZE_VALID)

#   model.load_weights(os.path.join(PATH, 'resnet152_bestmodel.h5'))
#   pred = model.predict(val_generator)

#   oos_y.append(df_train.loc[test,"sqft"].values)
#   oos_pred.append(pred)

#   score = np.sqrt(mean_squared_error(pred,df_train.loc[test,"sqft"].values))
#   print(f"Fold score (RMSE): {score}")

# # Build the oos prediction list and calculate the error.
# oos_y = np.concatenate(oos_y)
# oos_pred = np.concatenate(oos_pred)
# score = np.sqrt(mean_squared_error(oos_pred,oos_y))
# print(f"Final, out of sample score (RMSE): {score}")
```

Bayesian-Optimization parameters tuning

reminder: it takes a lot GPU, and will be crashed if the GPU and RAM are full

save pre-trained model results

```
# feature_train = model0.predict(train_generator)
# feature_val = model0.predict(val_generator)
# np.save("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learning-wustlfall
# np.save("/content/drive/My·Drive/Colab·Notebooks/applications-of-deep-learning-wustlfall


# feature_train.shape
```

```
(21598, 5, 5, 2048)
```

```
# STEP_SIZE_VALID=val_generator.n//val_generator.batch_size

# base_model.compile(loss = 'mean_squared_error', optimizer=Adam(learning_rate=1e-4), metr
# monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=50, verbose=1, mode
#         restore_best_weights=True)
# basemodel_checkpoint=ModelCheckpoint('basemodel.h5',monitor="val_loss",save_best_only=Tr
#                                      save_weights_only=True, verbose=1)
# lr_reduce = ReduceLROnPlateau(monitor='val_loss',factor=0.6,patience=30,verbose=1,mode='
# base_model.fit(train_generator, epochs=300, steps_per_epoch=250, validation_data = val_g
#                callbacks=[monitor,basemodel_checkpoint,lr_reduce], verbose = 1, validati
```

```
# if os.path.isfile('basemodel.h5'):
#     base_model.load_weights('basemodel.h5')
```

clear RAM and GPU, start the following steps

```
# feature_val = np.load("/content/drive/My·Drive/Colab·Notebooks/applications-of-deep-lear
#·feature_train·=·np.load("/content/drive/My·Drive/Colab·Notebooks/applications-of-deep-le
```

build the model that should be tuned

```
# hyperparameters
def build_model(neuronCount=256, learning_rate=1e-4,l2=1e-3,activation=3,rate=0.7,layer_nu
  initializer = he_normal()
  input_tensor = Input(shape=(HEIGHT, WIDTH, 3)) # !caution here
  base = basemodel(input_tensor=input_tensor)
  x=GlobalAveragePooling2D()(base.output)

  activation_dict={1:"relu",2:"elu",3:"tanh"}
  layer = 0
  while layer<layer_number:
    x = Dense(units=neuronCount,activation=activation_dict[activation],
                kernel_initializer=initializer,kernel_regularizer=tensorflow.keras.regul
    x=Dropout(rate)(x)
    layer+=1

  output=Dense(1,activation="linear")(x)
  model=Model(inputs=input_tensor,outputs=output)
  model.compile(loss = 'mean_squared_error', optimizer=Adam(learning_rate=learning_rate),
  lr_reduce = tensorflow.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',factor=0.6,p
  monitor = EarlyStopping(monitor='val_loss', min_delta=1e-3, patience=50, verbose=1, mode
                          restore_best_weights=True)
  checkpoint = ModelCheckpoint('city.h5',monitor='val_loss',verbose=1,save_weights_only=Tr
                               mode='min',save_freq='epoch')
  return model, lr_reduce, monitor, checkpoint
```

build training model

```
def fit_model(model, lr_reduce, monitor, checkpoint):
  """function that trains the head classifier"""
  history = model.fit(feature_train, df_train_cut["sqft"],
                      batch_size=32,
·····················epochs=50,
·····················verbose=1,
                      callbacks=[lr_reduce, monitor, checkpoint],
                      validation_data=(feature_val, df_validate_cut['sqft']),
                      shuffle=True)
  return history
```

evaluate the model and use the evaluation to optimize the model

```
def evaluate_model(model):
··"""function·that·evaluates·the·head·classifier"""
··evaluation·=·model.evaluate(feature_val,·df_validate_cut['sqft'])
··return·evaluation
```

tuning parameters

```
from tensorflow.keras.backend import clear_session
!pip install GPy
!pip install GPyOpt
import GPy
import GPyOpt
from GPyOpt.methods import BayesianOptimization
from matplotlib import pyplot as plt
# define the kernel for the Bayesian surrogate model using the "radial basis function" (RB
kernel = GPy.kern.RBF(input_dim=1, variance=1.0, lengthscale=1.0)
# hyperparameter bounds
bounds = [
          {'name': 'neuronCount', 'type': 'discrete', 'domain': (256,512,1024)},
          {'name': 'learning_rate', 'type': 'discrete', 'domain': (1e-4, 1e-5)},
          {'name': 'l2', 'type': 'discrete', 'domain': (1e-4,1e-5)},
          {'name': 'activation', 'type': 'discrete', 'domain': (1, 2, 3)},
          {'name': 'rate', 'type': 'discrete', 'domain': (0.5,0.6,0.7)},
          {'name': 'layer_number', 'type': 'discrete', 'domain': (4,5,6)}
          ]
# Note: 'activation' domain parameters (1, 2, 3) correspond to strings ('relu', 'elu', 'ta
# objective function for the model optimization:
def f(x):
  """objective function of the Bayesian surrogate model"""
  print()
  print("Hyperparameters:", x)
  # Retrieve 'accuracy' from the previously saved model
  try:
    previous_best_model = load_model('city.h5')
    previous_evaluation = evaluate_model(previous_best_model)
  except Exception:
    previous_best_model = None
  model, lr_reduce, monitor, checkpoint = build_model(
                                        neuronCount=int(x[:,0]),
                                        learning_rate=float(x[:,1]),
                                        l2=float(x[:,2]),
                                        activation=int(x[:,3]),
                                        rate=float(x[:,4]),
                                        layer_number=int(x[:,5])
                                        )
  history = fit_model(model, lr_reduce, monitor, checkpoint)
  evaluation = evaluate_model(model)
  print()
  print("parameters:\t{0} LOSS:\t{1} \t rmse:\t{2}".format(x,evaluation[0], evaluation[1])
  print(evaluation)
  print()
```

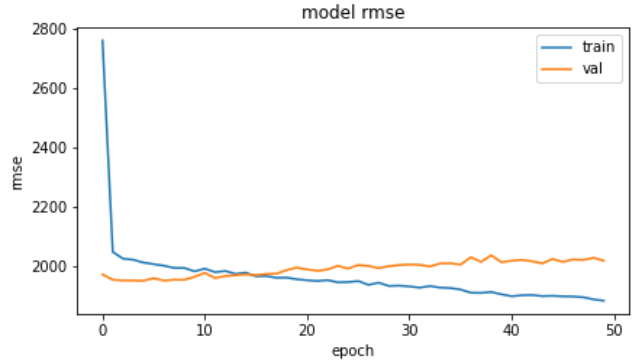
```
    # compare previous and current validation accuracies
    if not previous_best_model:
      save_model(model, 'city.h5', overwrite=False, include_optimizer=True)
    if previous_best_model and evaluation[-1] < previous_evaluation[-1]:
      save_model(model, 'city.h5', overwrite=True,include_optimizer=True)

    # Get the dictionary containing each metric and the loss for each epoch
    # history_dict = history.history
    # print(history_dict)
    def plot_history(history):
      """function that plots the model loss and accuracy"""
      plt.figure(1, figsize = (15,8))
      plt.subplot(221)
      plt.plot(history.history['rmse'])
      plt.plot(history.history['val_rmse'])
      plt.title('model rmse')
      plt.ylabel('rmse')
      plt.xlabel('epoch')
      plt.legend(['train', 'val'])
      plt.subplot(222)
      plt.plot(history.history['loss'])
      plt.plot(history.history['val_loss'])
      plt.title('model loss')
      plt.ylabel('loss')
      plt.xlabel('epoch')
      plt.legend(['train', 'val'])
      plt.show()
    # # plot the model accuracy and loss results
    plot_history(history)
    # delete the instantiated models from memory and clear the session
    del model
    del previous_best_model
    clear_session()
    return evaluation[1]

optimizer = BayesianOptimization(f=f,
                                 domain=bounds,
                                 model_type='GP',
                                 kernel=kernel,
                                 acquisition_type ='EI',
                                 acquisition_jitter = 0.01,
                                 exact_feval=False,
                                 normalize_Y=False,
                                #  maximize=True, we try to minimize the loss/rmse, so kee
                                 verbosity=True)

print()
print("=====================")
print("=====================")
print()
optimizer.run_optimization(max_iter=30, verbosity=False)
optimizer.plot_acquisition()
optimizer.plot_convergence()
optimizer.save_report('bayes_opt.txt')
```

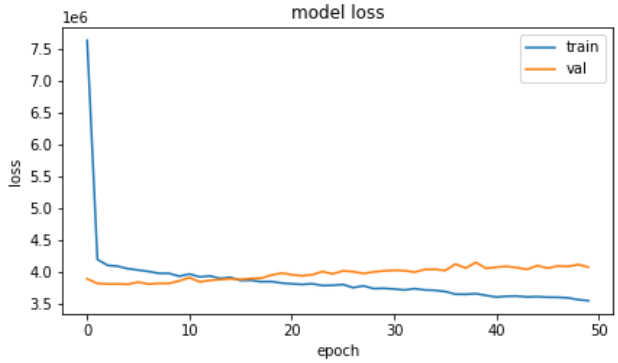
```
Requirement already satisfied: GPy in /usr/local/lib/python3.7/dist-packages (1.10.0)
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: cython>=0.29 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy>=1.3.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: paramz>=0.9.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from GP
Requirement already satisfied: decorator>=4.0.10 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: GPyOpt in /usr/local/lib/python3.7/dist-packages (1.2
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: scipy>=0.16 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: GPy>=1.8 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: paramz>=0.9.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cython>=0.29 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from GP
Requirement already satisfied: decorator>=4.0.10 in /usr/local/lib/python3.7/dist-pac

Hyperparameters: [[1.024e+03 1.000e-04 1.000e-05 2.000e+00 6.000e-01 4.000e+00]]
WARNING:tensorflow:Model was constructed with shape (None, 1) for input KerasTensor(t
Epoch 1/50
673/675 [============================>.] - ETA: 0s - loss: 7636229.0000 - rmse: 2763
Epoch 00001: val_loss improved from inf to 3888389.50000, saving model to city.h5
675/675 [==============================] - 6s 7ms/step - loss: 7628205.5000 - rmse: 2
Epoch 2/50
673/675 [============================>.] - ETA: 0s - loss: 4191992.5000 - rmse: 2047
Epoch 00002: val_loss improved from 3888389.50000 to 3817821.00000, saving model to c
675/675 [==============================] - 4s 6ms/step - loss: 4193941.0000 - rmse: 2
Epoch 3/50
672/675 [============================>.] - ETA: 0s - loss: 4098836.2500 - rmse: 2024
Epoch 00003: val_loss improved from 3817821.00000 to 3808268.75000, saving model to c
675/675 [==============================] - 4s 6ms/step - loss: 4101833.2500 - rmse: 2
Epoch 4/50
667/675 [============================>.] - ETA: 0s - loss: 4088750.7500 - rmse: 2022
Epoch 00004: val_loss did not improve from 3808268.75000
675/675 [==============================] - 4s 6ms/step - loss: 4086600.2500 - rmse: 2
Epoch 5/50
674/675 [============================>.] - ETA: 0s - loss: 4047378.5000 - rmse: 2011
Epoch 00005: val_loss improved from 3808268.75000 to 3804973.75000, saving model to c
675/675 [==============================] - 4s 6ms/step - loss: 4048124.7500 - rmse: 2
Epoch 6/50
666/675 [============================>.] - ETA: 0s - loss: 4026465.7500 - rmse: 2006
Epoch 00006: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 4026746.2500 - rmse: 2
Epoch 7/50
666/675 [============================>.] - ETA: 0s - loss: 4007218.0000 - rmse: 2001
Epoch 00007: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 4005704.7500 - rmse: 2
Epoch 8/50
673/675 [============================>.] - ETA: 0s - loss: 3978178.5000 - rmse: 1994
Epoch 00008: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3976215.0000 - rmse: 1
Epoch 9/50
666/675 [============================>.] - ETA: 0s - loss: 3977568.0000 - rmse: 1994
Epoch 00009: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3975298.2500 - rmse: 1
Epoch 10/50
669/675 [============================>.] - ETA: 0s - loss: 3926880.0000 - rmse: 1981
Epoch 00010: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3929741.2500 - rmse: 1
Epoch 11/50
671/675 [============================>.] - ETA: 0s - loss: 3968261.7500 - rmse: 1992
Epoch 00011: val_loss did not improve from 3804973.75000
```

```
675/675 [==============================] - 4s 6ms/step - loss: 3965847.0000 - rmse: 1
Epoch 12/50
673/675 [==============================>.] - ETA: 0s - loss: 3922565.2500 - rmse: 1980
Epoch 00012: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3919410.5000 - rmse: 1
Epoch 13/50
675/675 [==============================] - ETA: 0s - loss: 3934619.2500 - rmse: 1983
Epoch 00013: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3934619.2500 - rmse: 1
Epoch 14/50
674/675 [==============================>.] - ETA: 0s - loss: 3898537.7500 - rmse: 1974
Epoch 00014: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3896208.2500 - rmse: 1
Epoch 15/50
666/675 [==============================>.] - ETA: 0s - loss: 3912289.2500 - rmse: 1977
Epoch 00015: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3912478.7500 - rmse: 1
Epoch 16/50
669/675 [==============================>.] - ETA: 0s - loss: 3868907.2500 - rmse: 1966
Epoch 00016: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3863408.7500 - rmse: 1
Epoch 17/50
672/675 [==============================>.] - ETA: 0s - loss: 3870112.7500 - rmse: 1967
Epoch 00017: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3866753.0000 - rmse: 1
Epoch 18/50
669/675 [==============================>.] - ETA: 0s - loss: 3847114.0000 - rmse: 1961
Epoch 00018: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3844492.5000 - rmse: 1
Epoch 19/50
666/675 [==============================>.] - ETA: 0s - loss: 3847667.0000 - rmse: 1961
Epoch 00019: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3847112.0000 - rmse: 1
Epoch 20/50
668/675 [==============================>.] - ETA: 0s - loss: 3823968.5000 - rmse: 1955
Epoch 00020: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3823914.7500 - rmse: 1
Epoch 21/50
666/675 [==============================>.] - ETA: 0s - loss: 3814505.0000 - rmse: 1953
Epoch 00021: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3811576.2500 - rmse: 1
Epoch 22/50
671/675 [==============================>.] - ETA: 0s - loss: 3804809.7500 - rmse: 1950
Epoch 00022: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3802069.5000 - rmse: 1
Epoch 23/50
675/675 [==============================] - ETA: 0s - loss: 3813216.7500 - rmse: 1952
Epoch 00023: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3813216.7500 - rmse: 1
Epoch 24/50
673/675 [==============================>.] - ETA: 0s - loss: 3787444.2500 - rmse: 1946
Epoch 00024: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3784402.7500 - rmse: 1
Epoch 25/50
671/675 [==============================>.] - ETA: 0s - loss: 3787580.7500 - rmse: 1946
Epoch 00025: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3787740.7500 - rmse: 1
Epoch 26/50
668/675 [==============================>.] - ETA: 0s - loss: 3803474.5000 - rmse: 1950
Epoch 00026: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3800813.0000 - rmse: 1
Epoch 27/50
```

```
Epoch 27/50
666/675 [============================>.] - ETA: 0s - loss: 3749107.2500 - rmse: 1936
Epoch 00027: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3751129.5000 - rmse: 1
Epoch 28/50
675/675 [==============================] - ETA: 0s - loss: 3779648.5000 - rmse: 1944
Epoch 00028: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3779648.5000 - rmse: 1
Epoch 29/50
668/675 [============================>.] - ETA: 0s - loss: 3732359.0000 - rmse: 1931
Epoch 00029: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3736102.2500 - rmse: 1
Epoch 30/50
675/675 [==============================] - ETA: 0s - loss: 3740702.0000 - rmse: 1934
Epoch 00030: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3740702.0000 - rmse: 1
Epoch 31/50
667/675 [============================>.] - ETA: 0s - loss: 3734931.5000 - rmse: 1932
Epoch 00031: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3730088.7500 - rmse: 1
Epoch 32/50
673/675 [============================>.] - ETA: 0s - loss: 3715800.0000 - rmse: 1927
Epoch 00032: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3714484.7500 - rmse: 1
Epoch 33/50
674/675 [============================>.] - ETA: 0s - loss: 3735171.7500 - rmse: 1932
Epoch 00033: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3735478.5000 - rmse: 1
Epoch 34/50
673/675 [============================>.] - ETA: 0s - loss: 3714288.7500 - rmse: 1927
Epoch 00034: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3715188.7500 - rmse: 1
Epoch 35/50
669/675 [============================>.] - ETA: 0s - loss: 3713328.2500 - rmse: 1926
Epoch 00035: ReduceLROnPlateau reducing learning rate to 5.999999848427251e-05.

Epoch 00035: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3709428.0000 - rmse: 1
Epoch 36/50
665/675 [============================>.] - ETA: 0s - loss: 3692290.2500 - rmse: 1921
Epoch 00036: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3690311.2500 - rmse: 1
Epoch 37/50
668/675 [============================>.] - ETA: 0s - loss: 3648098.0000 - rmse: 1909
Epoch 00037: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3648875.0000 - rmse: 1
Epoch 38/50
666/675 [============================>.] - ETA: 0s - loss: 3650361.7500 - rmse: 1910
Epoch 00038: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3647033.7500 - rmse: 1
Epoch 39/50
669/675 [============================>.] - ETA: 0s - loss: 3658054.7500 - rmse: 1912
Epoch 00039: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3657556.5000 - rmse: 1
Epoch 40/50
672/675 [============================>.] - ETA: 0s - loss: 3627600.7500 - rmse: 1904
Epoch 00040: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3628933.0000 - rmse: 1
Epoch 41/50
668/675 [============================>.] - ETA: 0s - loss: 3607835.2500 - rmse: 1899
Epoch 00041: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3603634.2500 - rmse: 1
```

```
Epoch 42/50
673/675 [============================>.] - ETA: 0s - loss: 3615997.2500 - rmse: 1901
Epoch 00042: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3616454.0000 - rmse: 1
Epoch 43/50
669/675 [============================>.] - ETA: 0s - loss: 3617001.7500 - rmse: 1901
Epoch 00043: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3619236.0000 - rmse: 1
Epoch 44/50
675/675 [==============================] - ETA: 0s - loss: 3605075.0000 - rmse: 1898
Epoch 00044: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3605075.0000 - rmse: 1
Epoch 45/50
670/675 [============================>.] - ETA: 0s - loss: 3608101.5000 - rmse: 1899
Epoch 00045: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3609512.0000 - rmse: 1
Epoch 46/50
672/675 [============================>.] - ETA: 0s - loss: 3601209.2500 - rmse: 1897
Epoch 00046: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3602076.0000 - rmse: 1
Epoch 47/50
675/675 [==============================] - ETA: 0s - loss: 3600305.5000 - rmse: 1897
Epoch 00047: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3600305.5000 - rmse: 1
Epoch 48/50
673/675 [============================>.] - ETA: 0s - loss: 3589125.0000 - rmse: 1894
Epoch 00048: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3590800.7500 - rmse: 1
Epoch 49/50
674/675 [============================>.] - ETA: 0s - loss: 3563469.7500 - rmse: 1887
Epoch 00049: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3563199.5000 - rmse: 1
Epoch 50/50
675/675 [==============================] - ETA: 0s - loss: 3546989.2500 - rmse: 1883
Epoch 00050: val_loss did not improve from 3804973.75000
675/675 [==============================] - 4s 6ms/step - loss: 3546989.2500 - rmse: 1
75/75 [==============================] - 0s 4ms/step - loss: 4073042.0000 - rmse: 201

parameters:     [[1.024e+03 1.000e-04 1.000e-05 2.000e+00 6.000e-01 4.000e+00]] LOSS
[4073042.0, 2018.1778564453125]

[WARNING] city.h5 already exists - overwrite? [y/n]y
[TIP] Next time specify overwrite=True!
```






```
Hyperparameters: [[1.024e+03 1.000e-04 1.000e-04 2.000e+00 6.000e-01 4.000e+00]]
75/75 [==============================] - 0s 5ms/step - loss: 4073042.0000 - rmse: 201
---------------------------------------------------------------------------
InternalError                             Traceback (most recent call last)
<ipython-input-53-1dc267bc808a> in <module>()
     87                                 normalize_Y=False,
     88                               #  maximize=True. 求最小值
```

```
---> 89                                    verbosity=True)
     90
     91 print()
```

7 frames

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/constant_op.py in
convert_to_eager_tensor(value, ctx, dtype)
    104       dtype = dtypes.as_dtype(dtype).as_datatype_enum
    105   ctx.ensure_initialized()
--> 106   return ops.EagerTensor(value, ctx.device_name, dtype)
    107
    108

InternalError: Failed copying input tensor from
/job:localhost/replica:0/task:0/device:CPU:0 to
/job:localhost/replica:0/task:0/device:GPU:0 in order to run _EagerConst: Dst tensor
is not initialized.
```

SEARCH STACK OVERFLOW

load the model to predict and submit file

```
best_model = load_model('city.h5')

submit_datagen = ImageDataGenerator(rescale = 1./255)

submit_generator = submit_datagen.flow_from_dataframe(
        dataframe=df_test,
        directory=PATH,
        x_col="filename",
        batch_size = 1,
        shuffle = False,
        target_size=(HEIGHT, WIDTH),
        class_mode=None)

submit_generator.reset()
pred = best_model.predict(submit_generator,steps=len(df_test))
df_submit = pd.DataFrame({"id":df_test['id'],'sqft':pred[:,0].flatten()})
df_submit.to_csv("/content/drive/My Drive/Colab Notebooks/applications-of-deep-learning-wu
```

```
Epoch 1/300
250/250 [==============================] - 66s 239ms/step - loss: 7363987.5000 -
Epoch 2/300
250/250 [==============================] - 58s 230ms/step - loss: 1669661.0000 -
Epoch 3/300
250/250 [==============================] - 58s 233ms/step - loss: 1517575.3750 -
Epoch 4/300
250/250 [==============================] - 58s 230ms/step - loss: 1416379.3750 -
Epoch 5/300
250/250 [==============================] - 58s 231ms/step - loss: 1392923.3750 -
Epoch 6/300
250/250 [==============================] - 58s 230ms/step - loss: 1366217.2500 -
Epoch 7/300
250/250 [==============================] - 58s 231ms/step - loss: 1290442.1250 -
```

```
Epoch 8/300
250/250 [==============================] - 58s 231ms/step - loss: 1390184.7500 -
Epoch 9/300
250/250 [==============================] - 58s 230ms/step - loss: 1271122.7500 -
Epoch 10/300
250/250 [==============================] - 57s 229ms/step - loss: 1186863.6250 -
Epoch 11/300
250/250 [==============================] - 57s 229ms/step - loss: 1227311.0000 -
Epoch 12/300
250/250 [==============================] - 57s 229ms/step - loss: 1194605.3750 -
Epoch 13/300
250/250 [==============================] - 57s 229ms/step - loss: 1223517.5000 -
Epoch 14/300
250/250 [==============================] - 57s 228ms/step - loss: 1162990.3750 -
Epoch 15/300
250/250 [==============================] - 57s 229ms/step - loss: 1099538.0000 -
Epoch 16/300
250/250 [==============================] - 57s 229ms/step - loss: 1119877.8750 -
Epoch 17/300
250/250 [==============================] - 57s 228ms/step - loss: 1068267.5000 -
Epoch 18/300
250/250 [==============================] - 57s 228ms/step - loss: 1071955.8750 -
Epoch 19/300
250/250 [==============================] - 57s 228ms/step - loss: 1124639.0000 -
Epoch 20/300
250/250 [==============================] - 57s 228ms/step - loss: 1077422.2500 -
Epoch 21/300
250/250 [==============================] - 57s 229ms/step - loss: 1089136.8750 -
Epoch 22/300
250/250 [==============================] - 58s 230ms/step - loss: 1055068.1250 -
Epoch 23/300
250/250 [==============================] - 58s 230ms/step - loss: 991621.6250 -
Epoch 24/300
250/250 [==============================] - 57s 229ms/step - loss: 1073996.2500 -
Epoch 25/300
250/250 [==============================] - 57s 228ms/step - loss: 994070.3125 -
Epoch 26/300
250/250 [==============================] - 57s 230ms/step - loss: 1048044.9375 -
Epoch 27/300
250/250 [==============================] - 57s 229ms/step - loss: 1027593.0000 -
Epoch 28/300
250/250 [==============================] - 57s 228ms/step - loss: 979836.6875 -
Epoch 29/300
```

✓ 3s completed at 8:28 PM