

```

import tensorflow as tf
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Conv2DTranspose
from tensorflow.keras.layers import Cropping2D
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2

# create model
def standard_block(input_tensor, stage, num_filter, kernel_size=3, strides=(1, 1)):
    dropout_rate = 0.2
    # act = LeakyReLU()
    act = "relu"

    x = Conv2D(num_filter, kernel_size,
               activation=act, name='conv' + stage + '_1', padding='same',
               kernel_initializer='he_normal', kernel_regularizer=l2(1e-4))(input_tensor)
    x = Dropout(dropout_rate, name='dp' + stage + '_1')(x)
    x = Conv2D(num_filter, kernel_size, strides=strides,
               activation=act, name='conv' + stage + '_2', padding='same',
               kernel_initializer='he_normal', kernel_regularizer=l2(1e-4))(x)
    x = Dropout(dropout_rate, name='dp' + stage + '_2')(x)
    return x

# The shape of resistance (shape of NN input data): (2127, 235, 1)
# The shape of resistivity (shape of NN target data): (44, 188, 1)
def my_model():
    # setting
    input_shape = (2127, 235, 1)
    dropout_rate = 0.2
    num_filter = [16, 32, 64, 128, 256]
    up_strides = (1, 2)
    down_strides = (3, 2)
    crop = [((0, 0), (0, 0)), ((0, 2), (1, 1)), ((0, 2), (3, 3)), ((0, 3), (3, 3)), ((0, 3), (0, 0))]

    # construct keras model
    # with tf.device('/cpu:0'):
    inputs = Input(input_shape, name='main_input')
    x = Dropout(dropout_rate, name='dp_0')(inputs)

    conv1_1 = standard_block(x, stage='11', num_filter=num_filter[0],
                             kernel_size=(5, 3), strides=down_strides)
    conv2_1 = standard_block(conv1_1, stage='21', num_filter=num_filter[1],
                             kernel_size=(5, 3), strides=down_strides)
    conv3_1 = standard_block(conv2_1, stage='31', num_filter=num_filter[2],
                             kernel_size=(5, 3), strides=down_strides)
    conv4_1 = standard_block(conv3_1, stage='41', num_filter=num_filter[3],
                             kernel_size=(5, 3), strides=down_strides)

    conv5_1 = standard_block(conv4_1, stage='51', num_filter=num_filter[4])

```

```

conv5_1 = Cropping2D(crop[0])(conv5_1)

up4_2 = Conv2DTranspose(num_filter[3], (3, 3), strides=(2, 2), name='up42', padding='same')(conv5_1)
conv4_2 = standard_block(up4_2, stage='42', num_filter=num_filter[3])
conv4_2 = Cropping2D(crop[1])(conv4_2)

up3_3 = Conv2DTranspose(num_filter[2], (3, 3), strides=up_strides, name='up33', padding='same')(conv4_2)
conv3_3 = standard_block(up3_3, stage='33', num_filter=num_filter[2])
conv3_3 = Cropping2D(crop[2])(conv3_3)

up2_4 = Conv2DTranspose(num_filter[1], (3, 3), strides=up_strides, name='up24', padding='same')(conv3_3)
conv2_4 = standard_block(up2_4, stage='24', num_filter=num_filter[1])
conv2_4 = Cropping2D(crop[3])(conv2_4)

up1_5 = Conv2DTranspose(num_filter[0], (3, 3), strides=up_strides, name='up15', padding='same')(conv2_4)
conv1_5 = standard_block(up1_5, stage='15', num_filter=num_filter[0])
conv1_5 = Cropping2D(crop[4])(conv1_5)

x = standard_block(conv1_5, stage='_out', num_filter=8)
outputs = Conv2D(1, (1, 1), name='main_output', kernel_initializer='he_normal',
                  padding='same', kernel_regularizer=l2(3e-4))(x)

outputs = Activation('linear', dtype='float32')(outputs) # for mixed_float16

model = Model(inputs=inputs, outputs=outputs, name='FCN')
model.summary()
return model

```

```

# - Use simpeg to generate dataset for training/validation/testing
# -- Generic setting
save_dataset_dir: "../data/raw_data3" # In this directory, the program will automatically create a training
/ validation / testing directory for storing data and a "simulator.pkl" for recording simulation information.
num_examples: 10 # Number of examples. That is, the total number of synthetic models that will be gene
rated in one round.
train_ratio: 0.8 # Ratio of the number of examples (num_examples) to be used for training.
valid_ratio: 0.1 # Ratio of the number of examples (num_examples) to be used for validating.
test_ratio: 0.1 # Ratio of the number of examples (num_examples) to be used for testing.
geometry_urf: "../config/Liyutan.urf" # The path to the urf file is used to construct the electrode array in f
orward simulation.
terrain_file: "../config/Liyutan_flat.trn" # The path to the terrain file is used to construct the electrode arr
ay in forward simulation.
array_type: "all_combination" # {"all_combination", "Wenner"}. Select the electrode pair that matches the
array configuration. All array types are dipole-dipole settings.

# -- Generate synthetic models randomly. (NOTE: we will use truncated normal distribution in linear scale)

# --- Generic setting
# ---- For SimPEG.EM.Static.DC.IO.set_mesh function
# =====
# core_x_length = lineLength + dx * 6
# if core_x_length * 0.2 > delta_terrain:
#   core_z_length = core_x_length * 0.2
# else:
#   core_z_length = delta_terrain * 1.75
# =====
dx: 1 # Mesh length in x direction.
dz: 1 # Mesh length in z direction.
num_pad_x: 12 # Number of padding mesh in the x direction (both sides).
num_pad_z: 12 # Number of padding mesh in the z direction (lower side only).
pad_rate_x: 1 # Rate of change of padding mesh in the x direction.
pad_rate_z: 1 # Rate of change of padding mesh in the z direction.
# ---- This kernel is used to smooth (moving average) synthetic resistivity models in linear scale.
x_kernel_size: 3 # Kernel size in the x direction.
z_kernel_size: 3 # Kernel size in the z direction.

# --- Background
scale_background: "log10" # {"linear", "log10"}. The resistivity scale during the sampling phase.
pdf_background: "normal" # {"uniform", "normal"}. The probability distribution function of the sample.
a_background: 2.7 # Lower bound or mu (mean) of pdf_background.
b_background: 0.6 # Upper bound or std (standard deviation) of pdf_background.
# ---- Use following parameters to simulate porous media (Each mesh has a different value).
use_hidden_background: True # {True, False}. If True, it will use 'hidden_*' to control a_background and
b_background.
hidden_pdf_background: "normal" # {"uniform", "normal"}. The hidden probability distribution function of t
he a_background and b_background.
hidden_a_for_a_background: 2.7 # Lower bound or mu (mean) of hidden_pdf_background for hidden var
iable a_background.
hidden_b_for_a_background: 0.3 # Upper bound or std(standard deviation) of hidden_pdf_background fo
r hidden variable a_background.
hidden_a_for_b_background: 0.6 # Lower bound or mu (mean) of hidden_pdf_background for hidden var
iable b_background.
hidden_b_for_b_background: 0.1 # Upper bound or std(standard deviation) for hidden variable b_backgr
ound.

```

```
# --- Rectangle (block)
num_rect: # Number of rectangles.
  type: "list" # {"list", "range"}. The interval of type "range" including start but excluding stop.
  value: [0, 1, 2] # Random sampling in 'value' defined by 'type'.
# ---- Geometry
w_range: [1, 150] # Width range (x direction).
h_range: [1, 30] # Height range (z direction).
# ---- Value
scale_rect: "log10" # {"linear", "log10"}. The resistivity scale during the sampling phase.
pdf_rect: "normal" # {"uniform", "normal"}. The probability distribution function of the sample.
a_rect: 2.7 # Lower bound or mu (mean) of pdf_background.
b_rect: 0.6 # Upper bound or std (standard deviation) of pdf_background.
# ----- Use following parameters to simulate porous media (Each mesh has a different value).
use_hidden_rect: True # {True, False}. If True, it will use 'hidden_*' to control a_rect and b_rect.
hidden_pdf_rect: "normal" # {"uniform", "normal"}. The hidden probability distribution function of the a_rect and b_rect.
hidden_a_for_a_rect: 2.7 # Lower bound or mu (mean) of hidden_pdf_background for hidden variable a_rect.
hidden_b_for_a_rect: 0.3 # Upper bound or std(standard deviation) of hidden_pdf_background for hidden variable a_rect.
hidden_a_for_b_rect: 0.6 # Lower bound or mu (mean) of hidden_pdf_background for hidden variable b_rect.
hidden_b_for_b_rect: 0.1 # Upper bound or std(standard deviation) for hidden variable b_rect.

# --- Circle
num_circle: # Number of circles.
  type: "list" # {"list", "range"}. The interval of type "range" including start but excluding stop.
  value: [0, 1, 2] # Random sampling in 'value' defined by 'type'.
# ---- Geometry
radius_bound: [1, 75] # Lower bound and upper bound of radius.
# ---- Value
scale_circle: "log10" # {"linear", "log10"}. The resistivity scale during the sampling phase.
pdf_circle: "normal" # {"uniform", "normal"}. The probability distribution function of the sample.
a_circle: 2.7 # Lower bound or mu (mean) of pdf_background.
b_circle: 0.6 # Upper bound or std (standard deviation) of pdf_background.
# ----- Use following parameters to simulate porous media (Each mesh has a different value).
use_hidden_circle: True # {True, False}. If True, it will use 'hidden_*' to control a_circle and b_circle.
hidden_pdf_circle: "normal" # {"uniform", "normal"}. The hidden probability distribution function of the a_circle and b_circle.
hidden_a_for_a_circle: 2.7 # Lower bound or mu (mean) of hidden_pdf_background for hidden variable a_circle.
hidden_b_for_a_circle: 0.3 # Upper bound or std(standard deviation) of hidden_pdf_background for hidden variable a_circle.
hidden_a_for_b_circle: 0.6 # Lower bound or mu (mean) of hidden_pdf_background for hidden variable b_circle.
hidden_b_for_b_circle: 0.1 # Upper bound or std(standard deviation) for hidden variable b_circle.
```

Tx

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,
38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64

Rx

1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,
38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64

RxP2

65

:Geometry

1,2.00,0.00,0.00
2,4.00,0.00,0.00
3,6.00,0.00,0.00
4,8.00,0.00,0.00
5,10.00,0.00,0.00
6,12.00,0.00,0.00
7,14.00,0.00,0.00
8,16.00,0.00,0.00
9,18.00,0.00,0.00
10,20.00,0.00,0.00
11,22.00,0.00,0.00
12,24.00,0.00,0.00
13,26.00,0.00,0.00
14,28.00,0.00,0.00
15,30.00,0.00,0.00
16,32.00,0.00,0.00
17,34.00,0.00,0.00
18,36.00,0.00,0.00
19,38.00,0.00,0.00
20,40.00,0.00,0.00
21,42.00,0.00,0.00
22,44.00,0.00,0.00
23,46.00,0.00,0.00
24,48.00,0.00,0.00
25,50.00,0.00,0.00
26,52.00,0.00,0.00
27,54.00,0.00,0.00
28,56.00,0.00,0.00
29,58.00,0.00,0.00
30,60.00,0.00,0.00
31,62.00,0.00,0.00
32,64.00,0.00,0.00
33,66.00,0.00,0.00
34,68.00,0.00,0.00
35,70.00,0.00,0.00
36,72.00,0.00,0.00
37,74.00,0.00,0.00
38,76.00,0.00,0.00
39,78.00,0.00,0.00
40,80.00,0.00,0.00
41,82.00,0.00,0.00
42,84.00,0.00,0.00
43,86.00,0.00,0.00
44,88.00,0.00,0.00
45,90.00,0.00,0.00

46,92.00,0.00,0.00
47,94.00,0.00,0.00
48,96.00,0.00,0.00
49,98.00,0.00,0.00
50,100.00,0.00,0.00
51,102.00,0.00,0.00
52,104.00,0.00,0.00
53,106.00,0.00,0.00
54,108.00,0.00,0.00
55,110.00,0.00,0.00
56,112.00,0.00,0.00
57,114.00,0.00,0.00
58,116.00,0.00,0.00
59,118.00,0.00,0.00
60,120.00,0.00,0.00
61,122.00,0.00,0.00
62,124.00,0.00,0.00
63,126.00,0.00,0.00
64,128.00,0.00,0.00
65,0,0,0