```
"""
プログラム１：
トレーンデータとテストデータをそれぞれファイルから読み込み、
画像データとラベルに分け、numpyファイルで保存
"""

import numpy as np
import cv2

def cvt_data(infile, outfile):
    data = []
    labels = []

    with open(infile) as f:
        txt = f.readlines()
        txt = [line.split(' ') for line in txt]

    for i, line in enumerate(txt):
        data.append(cv2.imread('.'+line[0][7:]))
        labels.append(cv2.imread('.'+line[1][7:][:-1])[:,:,0])

    x = np.array(data)
    y = np.array(labels)
    np.savez(outfile, x = x, y = y)
    print('saved:', outfile)
    print('x shape =', x.shape)
    print('y shape =', y.shape)

train_infile = './CamVid/train.txt'
test_infile = './CamVid/test.txt'
train_outfile = './train.npz'
test_outfile = './test.npz'

cvt_data(train_infile, train_outfile)
cvt_data(test_infile, test_outfile)
```

```
"""
プログラム２：
画像データの正規化とラベルのone-hot形式化関数を作り、
データをnumpyファイルから読み込み、
学習に適する形式に自動変換するモデュールを作成
"""

%%writefile dataset.py

import cv2
import numpy as np
from keras.applications import imagenet_utils
import os

class Dataset:
    def __init__(self, input_file, classes):
        self.input_file = input_file
        self.classes = classes

    def normalized(self, img):
        norm = np.zeros(img.shape, np.float32)

        b = img[:,:,0]
        g = img[:,:,1]
        r = img[:,:,2]

        norm[:,:,0] = cv2.equalizeHist(b)
        norm[:,:,1] = cv2.equalizeHist(g)
        norm[:,:,2] = cv2.equalizeHist(r)

        return norm

    def one_hot_it(self, label):
        one_hot = np.zeros((label.shape[0],label.shape[1], self.classes), np.int32)

        for i in range(label.shape[0]):
            for j in range(label.shape[1]):
                one_hot[i, j, label[i, j]] = 1
```

```
                one_hot[i,j,label[i, j]] = 1

        return one_hot

    def load_data(self):
        dataset = np.load(self.input_file)
        x = dataset['x']
        y = dataset['y']

        imgs = []
        labels = []
        for i, img in enumerate(x):
            imgs.append(self.normalized(img))

        for i, label in enumerate(y):
            labels.append(self.one_hot_it(label))

        x = np.array(imgs)
        y = np.array(labels)
        x = imagenet_utils.preprocess_input(x)
        y = y.reshape(-1, y.shape[1]*y.shape[2], self.classes)

        return x, y
```

"""
プログラム３：
インデックスあるプーリング層とアップサンプリング層を作成
"""

```
%%writefile layers.py
import keras.backend as K
from keras.layers import Layer

class MaxPoolingWithArgmax2D(Layer):
    def __init__(self, pool_size=(2,2), strides=(2,2), padding='same', **kwargs):
        super(MaxPoolingWithArgmax2D, self).__init__(**kwargs)
        self.pool_size = pool_size
        self.strides = strides
```

```
        self.padding = padding

    def call(self, inputs, **kwargs):
        pool_size = self.pool_size
        strides = self.strides
        padding = self.padding
        if K.backend() == 'tensorflow':
            ksize = [1, pool_size[0], pool_size[1], 1]
            strides = [1, strides[0], strides[1], 1]
            padding = padding.upper()
            output, argmax = K.tf.nn.max_pool_with_argmax(inputs, ksize=ksize, ¥
                strides=strides, padding=padding)
        else:
            errmsg = '{} backend is not supported for layer {}'.format(
                K.backend(), type(self).__name__)
            raise NotImplementedError(errmsg)
        argmax = K.cast(argmax, K.floatx())
        return [output, argmax]

    def compute_output_shape(self, input_shape):
        ratio = (1,self.pool_size[0] ,self.pool_size[1], 1)
        output_shape = [dim // ratio[idx] if dim is not None else None ¥
            for idx, dim in enumerate(input_shape)]
        output_shape = tuple(output_shape)
        return [output_shape, output_shape]

    def compute_mask(self, inputs, mask=None):
        return 2 * [None]

class UpsamplingWithArgmax2D(Layer):
    def __init__(self, size, **kwargs):
        super(UpsamplingWithArgmax2D, self).__init__(**kwargs)
        self.size = size

    def call(self, inputs, output_shape=None):
        size = self.size
        updates, mask = inputs[0], inputs[1]
        with K.tf.variable_scope(self.name):
```

```
        with K.tf.variable_scope(self.name):
            mask = K.cast(mask, 'int32')
            input_shape = K.tf.shape(updates, out_type='int32')

            if output_shape is None:
                output_shape = (input_shape[0], input_shape[1]*size[0], ¥
                    input_shape[2]*size[1], input_shape[3])
            self.output_shape1 = output_shape

            one_like_mask = K.ones_like(mask, dtype='int32')
            batch_shape = (input_shape[0], 1, 1, 1)
            batch_range = K.reshape(K.tf.range(output_shape[0], dtype='int32'), ¥
                shape=batch_shape)
            b = one_like_mask * batch_range
            y = mask // (output_shape[2]*output_shape[3])
            x = (mask // output_shape[3]) % output_shape[2]
            feather_range = K.tf.range(output_shape[3], dtype='int32')
            f = one_like_mask * feather_range

            updates_size = K.tf.size(updates)
            indices = K.transpose(K.reshape(K.stack([b, y, x, f]), ¥
                [4, updates_size]))
            values = K.reshape(updates, [updates_size])
            ret = K.tf.scatter_nd(indices, values, output_shape)
            return ret

    def compute_output_shape(self, input_shape):
      input_shape = input_shape[0]
      return (input_shape[0],
              input_shape[1] * self.size[0],
              input_shape[2] * self.size[1],
              input_shape[3])
```

```
Overwriting layers.py
```

"""

プログラム４：

SegNetモデルを作成

```
"""

%%writefile model_SegNet.py
from keras.layers import Input
from keras.layers.convolutional import Conv2D, MaxPooling2D, UpSampling2D
from keras.layers.core import Activation, Reshape
from keras.layers.normalization import BatchNormalization
from keras.models import Model
from keras.constraints import max_norm
from layers import MaxPoolingWithArgmax2D, UpsamplingWithArgmax2D

def SegNet(input_shape, nb_classes=12, kernel=3, pool_size=(2,2)):
    inputs = Input(input_shape)

    conv_1 = Conv2D(64, (kernel, kernel), padding="same")(inputs)
    conv_1 = BatchNormalization()(conv_1)
    conv_1 = Activation("relu")(conv_1)

    pool_1, mask_1 = MaxPoolingWithArgmax2D(pool_size)(conv_1)

    conv_3 = Conv2D(128, (kernel, kernel), padding="same")(pool_1)
    conv_3 = BatchNormalization()(conv_3)
    conv_3 = Activation("relu")(conv_3)

    pool_2, mask_2 = MaxPoolingWithArgmax2D(pool_size)(conv_3)

    conv_5 = Conv2D(256, (kernel, kernel), padding="same")(pool_2)
    conv_5 = BatchNormalization()(conv_5)
    conv_5 = Activation("relu")(conv_5)

    pool_3, mask_3 = MaxPoolingWithArgmax2D(pool_size)(conv_5)

    conv_8 = Conv2D(512, (kernel, kernel), padding="same")(pool_3)
    conv_8 = BatchNormalization()(conv_8)
    conv_8 = Activation("relu")(conv_8)

    pool_4, mask_4 = MaxPoolingWithArgmax2D(pool_size=(3,2),strides=(3,2))(conv_8)
```

```
    print("Build enceder done..")

    upsample_2 = UpsamplingWithArgmax2D(size=(3,2))([pool_4, mask_4])

    conv_17 = Conv2D(256, (kernel, kernel), padding="same")(upsample_2)
    conv_17 = BatchNormalization()(conv_17)
    conv_17 = Activation("relu")(conv_17)

    upsample_3 = UpsamplingWithArgmax2D(pool_size)([conv_17, mask_3])

    conv_20 = Conv2D(128, (kernel, kernel), padding="same")(upsample_3)
    conv_20 = BatchNormalization()(conv_20)
    conv_20 = Activation("relu")(conv_20)

    upsample_4 = UpsamplingWithArgmax2D(pool_size)([conv_20, mask_2])

    conv_23 = Conv2D(64, (kernel, kernel), padding="same")(upsample_4)
    conv_23 = BatchNormalization()(conv_23)
    conv_23 = Activation("relu")(conv_23)

    upsample_5 = UpsamplingWithArgmax2D(pool_size)([conv_23, mask_1])

    conv_25 = Conv2D(64, (kernel, kernel), padding="same")(upsample_5)
    conv_25 = BatchNormalization()(conv_25)
    conv_25 = Activation("relu")(conv_25)

    conv_26 = Conv2D(nb_classes, (1, 1), padding="valid")(conv_25)
    conv_26 = BatchNormalization()(conv_26)
    conv_26 = Reshape((input_shape[0]*input_shape[1], nb_classes), ¥
        input_shape=(input_shape[0],input_shape[1], nb_classes))(conv_26)

    outputs = Activation("softmax")(conv_26)
    print("Build decoder done..")

    model = Model(inputs, outputs, name="SegNet")

    return model
```

```
Overwriting model_SegNet.py
```

```
"""
プログラム５：
SegNetモデルによりトレーンデータを学習させ、
学習済みの重みを書き出す
"""

%tensorflow_version 1.x

import os
dir_path = "/content/drive/My Drive/Python/SegNet/"
os.chdir(dir_path)

import keras
import keras.backend as K
from dataset import Dataset
from model_SegNet import SegNet
import numpy as np

config = K.tf.ConfigProto(gpu_options=K.tf.GPUOptions(allow_growth=True, ¥
    per_process_gpu_memory_fraction=0.8))
session = K.tf.Session(config=config)
K.tensorflow_backend.set_session(session)

input_shape = (360, 480, 3)
classes = 12
batch_size = 10
epochs=100
class_weight=[0.2595, 0.1826, 4.5640, 0.1417, 0.5051, 0.3826, 9.6446, 1.8418, 6.6823, 6.2478, 3.0, 7.3614]
train_file = './train.npz'
log_filepath = './logs/'

def main():
    print("loading data...")
    train_ds = Dataset(train_file, classes)
```

```
    data_x, data_y = train_ds.load_data()
    rand_num = np.random.randint(100)
    np.random.seed(rand_num)
    np.random.shuffle(data_x)
    np.random.seed(rand_num)
    np.random.shuffle(data_y)
    train_x = data_x[20:]
    train_y = data_y[20:]
    print("input train_data shape:", train_x.shape)
    print("input train_labels shape:", train_y.shape)
    val_x = data_x[:20]
    val_y = data_y[:20]
    print("input val_data shape:", val_x.shape)
    print("input val_labels shape:", val_y.shape)

    tb_cb = keras.callbacks.TensorBoard(log_dir=log_filepath, histogram_freq=1, ¥
        write_graph=True, write_images=True)

    print("training....")
    model = SegNet(input_shape, classes)
    model.compile(loss="categorical_crossentropy", optimizer='adadelta', ¥
        metrics=["accuracy"])

    model.fit(train_x, train_y, batch_size= batch_size, epochs=epochs, verbose=1, ¥
        class_weight=class_weight, validation_data=(val_x, val_y), shuffle=True, ¥
        callbacks=[tb_cb])

    model.save("SegNet_w.h5")

if __name__ == '__main__':

    main()
```

```
TensorFlow 1.x selected.
Using TensorFlow backend.
loading data...
input train_data shape: (347, 360, 480, 3)
input train_labels shape: (347, 172800, 12)
input val_data shape: (20, 360, 480, 3)
input val_labels shape: (20, 172800, 12)
training....
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:74: The name tf.get_default_graph is depre

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecatec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is deprec

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:174: The name tf.get_default_session is de

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.global_variables is depre

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:199: The name tf.is_variable_initialized i

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:206: The name tf.variables_initializer is

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1834: The name tf.nn.fused_batch_norm is c

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:133: The name tf.placeholder_with_default

Build enceder done..
WARNING:tensorflow:From /content/drive/My Drive/Python/SegNet/layers.py:46: The name tf.variable_scope is deprecated. Please use tf.compat.v1

Build decoder done..
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:790: The name tf.train.Optimizer is deprecated. Please use

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3295: The name tf.log is deprecated. Pleas

WARNING:tensorflow:From /tensorflow-1.15.2/python3.6/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_op
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:986: The name tf.assign_add is deprecated.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is deprecated. Ple

Train on 347 samples, validate on 20 samples
```

```
Epoch 80/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1519 - acc: 0.9515 - val_loss: 0.4383 - val_acc: 0.8764
Epoch 81/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1694 - acc: 0.9454 - val_loss: 0.4292 - val_acc: 0.8742
Epoch 82/100
347/347 [==============================] - 35s 102ms/step - loss: 0.1480 - acc: 0.9525 - val_loss: 0.3920 - val_acc: 0.8902
Epoch 83/100
347/347 [==============================] - 36s 102ms/step - loss: 0.1379 - acc: 0.9559 - val_loss: 0.6890 - val_acc: 0.8042
Epoch 84/100
347/347 [==============================] - 36s 102ms/step - loss: 0.1338 - acc: 0.9569 - val_loss: 0.4620 - val_acc: 0.8665
Epoch 85/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1394 - acc: 0.9551 - val_loss: 0.4922 - val_acc: 0.8573
Epoch 86/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1346 - acc: 0.9565 - val_loss: 0.4832 - val_acc: 0.8639
Epoch 87/100
347/347 [==============================] - 35s 102ms/step - loss: 0.1269 - acc: 0.9588 - val_loss: 0.4261 - val_acc: 0.8765
Epoch 88/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1913 - acc: 0.9385 - val_loss: 0.4728 - val_acc: 0.8678
Epoch 89/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1282 - acc: 0.9585 - val_loss: 0.3968 - val_acc: 0.8855
Epoch 90/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1226 - acc: 0.9603 - val_loss: 0.4836 - val_acc: 0.8634
Epoch 91/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1243 - acc: 0.9596 - val_loss: 0.4087 - val_acc: 0.8849
Epoch 92/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1256 - acc: 0.9590 - val_loss: 0.4447 - val_acc: 0.8734
Epoch 93/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1976 - acc: 0.9355 - val_loss: 0.5465 - val_acc: 0.8530
Epoch 94/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1355 - acc: 0.9555 - val_loss: 0.4196 - val_acc: 0.8853
Epoch 95/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1217 - acc: 0.9603 - val_loss: 0.4304 - val_acc: 0.8827
Epoch 96/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1180 - acc: 0.9614 - val_loss: 0.4610 - val_acc: 0.8699
Epoch 97/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1251 - acc: 0.9591 - val_loss: 0.4422 - val_acc: 0.8741
Epoch 98/100
347/347 [==============================] - 36s 102ms/step - loss: 0.1151 - acc: 0.9621 - val_loss: 0.4328 - val_acc: 0.8846
Epoch 99/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1155 - acc: 0.9620 - val_loss: 0.4474 - val_acc: 0.8794
Epoch 100/100
347/347 [==============================] - 36s 103ms/step - loss: 0.1172 - acc: 0.9613 - val_loss: 0.4182 - val_acc: 0.8863
```

```
"""
プログラム６：
テンソルボードを使て、
学習結果をグラフで表す
"""

import os
dir_path = "/content/drive/My Drive/Python/SegNet/"
os.chdir(dir_path)

%reload_ext tensorboard
%tensorboard --logdir logs  --host=127.0.0.1
```

TensorBoard SCALARS IMAGES INACTIVE

Show data download links

Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE

WALL

Runs

Write a regex to filter runs

.

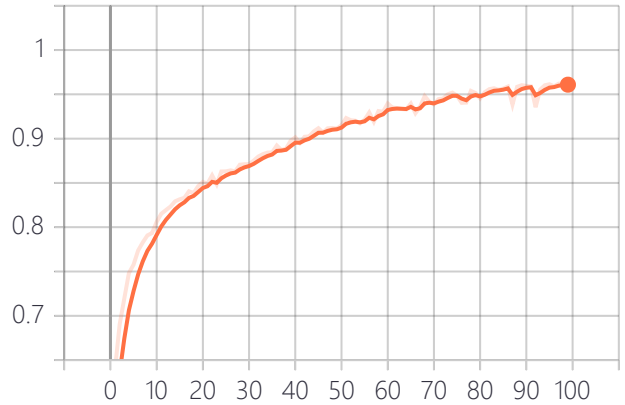
TOGGLE ALL RUNS

logs

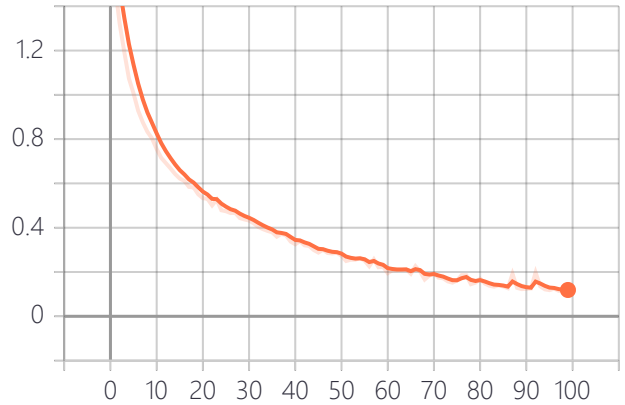
.*

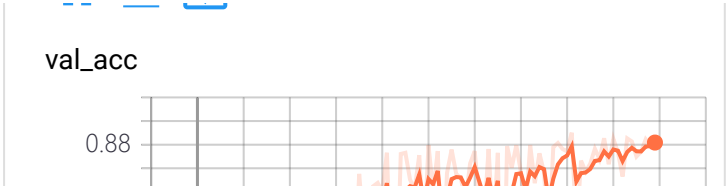
Tags matching /.*/(all tags) 4

acc



loss






```
"""
プログラム７：
テストデータを読み込んで予測させ、
画像データに変換し、
pngファイルに書き出す
"""

%tensorflow_version 1.x

import os
dir_path = "/content/drive/My Drive/Python/SegNet/"
os.chdir(dir_path)

import numpy as np
import keras
from model_SegNet import SegNet
from dataset import Dataset
from PIL import Image

test_file = './test.npz'
model_file = './SegNet_w.h5'
dirpath = './pred_images/'
input_shape = (360, 480, 3)
classes = 12
batch_size= 10

def predict(test_x, test_y, model_file, batch_size=batch_size):
    model = SegNet(input_shape, classes)
    model.load_weights(model_file)
    pred = model.predict(test_x, batch_size=batch_size)
```

```
        pred = pred.reshape((test_x.shape[0], test_x.shape[1], test_x.shape[2], classes))
        pred_img = np.zeros((pred.shape[0], pred.shape[1], pred.shape[2]))
        pred_img = pred.argmax(axis=3)
        model.compile(loss="categorical_crossentropy", optimizer='adadelta', ¥
            metrics=["accuracy"])
        score = model.evaluate(test_x, test_y, verbose = 1)
        print("正確率=", score[1], "loss=", score[0])
        return pred_img

    def write_img(images, dirpath):
        if not os.path.isdir(dirpath): os.mkdir(dirpath)
        Sky = [128,128,128]
        Building = [128,0,0]
        Pole = [192,192,128]
        Road_marking = [255,69,0]
        Road = [128,64,128]
        Pavement = [60,40,222]
        Tree = [128,128,0]
        SignSymbol = [192,128,128]
        Fence = [64,64,128]
        Car = [64,0,128]
        Pedestrian = [64,64,0]
        Bicyclist = [0,128,192]
        Unlabelled = [0,0,0]
        obj_labels = np.array([Sky, Building, Pole, Road_marking, Road, Pavement, ¥
            Tree, SignSymbol, Fence, Car, Pedestrian, Bicyclist, Unlabelled])

        rgb = np.zeros((images.shape[1], images.shape[2], 3))
        num = 0
        for b in range(images.shape[0]):
            num += 1
            print('image number:', num)
            for h in range(images.shape[1]):
                for w in range(images.shape[2]):
                    rgb[h, w] = obj_labels[images[b, h, w]]
            img = Image.fromarray(np.uint8(rgb))
            savepath = dirpath + str(num) + '.png'
            img.save(savepath)
```

```
        print('images saved.')

    def main():
        print('loading data...')
        test_ds = Dataset(test_file, classes)
        test_x, test_y = test_ds.load_data()
        print("input test_data shape:", test_x.shape)
        print("input test_labels shape:", test_y.shape)

        print('predicting images...')
        pred_img = predict(test_x, test_y, model_file)

        print('writing images...')
        write_img(pred_img, dirpath)

    if __name__ == '__main__':
        main()
```

```
TensorFlow is already loaded. Please restart the runtime to change versions.
loading data...
input test_data shape: (233, 360, 480, 3)
input test_labels shape: (233, 172800, 12)
predicting images...
Build enceder done..
Build decoder done..
233/233 [==============================] - 249s 1s/step
正確率= 0.7880066825084932 loss= 0.8400136030283097
writing images...
image number: 1
image number: 2
image number: 3
image number: 4
image number: 5
image number: 6
image number: 7
image number: 8
image number: 9
image number: 10
image number: 11
image number: 12
image number: 13
image number: 14
image number: 15
image number: 16
image number: 17
image number: 18
image number: 19
image number: 20
image number: 21
image number: 22
image number: 23
image number: 24
image number: 25
image number: 26
image number: 27
image number: 28
image number: 29
image number: 30
image number: 31
image number: 32
```