WaterClassification

unknown

Aug 08, 2022

CONTENTS

CHAPTER

ONE

WATER CLASSIFICATION'S MODULE AND FUNCTIONS

This package is a tool to preprocess the data, use Multiview learning/ Feature stack deep learning in order to class Water bodies and evaluate the performance of the model.

See :*tools* folder for more information.

CHAPTER

TWO

CONFIG.PY

class tools.config.**configuration**(*PROJECT_TITLE*, *BANDS1*, *TRAIN_SIZE*, *EVAL_SIZE*, *BANDS2=[]*, *BANDS3=[]*, *country='TH'*, *image=None*, *sam_arr=None*, *type_=1*, *LOSS='categorical_crossentropy'*, *EPOCHS=10*, *BATCH_SIZE=16*, *dropout_prob=0.3*)

In each experiment, the combinations of satellite's bands that is used to train the neural network is different. Also the way to train the neural network is also different, whether it is feature stack, multiview learning with two or three perceptrons. As each experiment has different settings, it is important to store them and reuse this throughout the project. This class enables user to store the settings and reuse the settings.

Initialising/storing the parameters to use later

Parameters

- **PROJECT_TITLE** (*string*) –
- **BANDS1** (*list*) –
- **TRAIN_SIZE** (*int/float*) –
- **EVAL_SIZE** (*int/float*) –
- **BANDS2** (*list*) –
- **BANDS3** (*list*) –
- **country** (*string*) –
- **image** (*ee.image.Image*) –
- **sam_arr** (*ee.image.Image*) –
- **type** (*int/float*) –

CHAPTER

THREE

METRICS_.PY

tools.metrics_.**MetricCalculator**(*model*, *test_data*, *total_steps*)

This function takes in the feature stack model loaded from google cloud bucket, the test_data which is the tensor object and the number of steps and returns the metrics including accuracy, recall, precision and F1

Parameters

- **model** (*keras.engine.functional.Functional*) –
- **test_data** (*RepeatDataset with tf.float32*) –
- **total_steps** (*int/float*) –

Return type

Returns the precision, recall, f1, accuracy metric based on the model performance.

Notes

This function should be used instead of the F1, custom_accuracy written above. The code is obtained/modified from:

https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras

https://neptune.ai/blog/implementing-the-macro-f1-score-in-keras

tools.metrics_.**MetricCalculator_NDWI**(*test_data*, *total_steps*)

This function takes in the test_data which is the tensor object and the number of steps and returns the metrics including accuracy, recall, precision and F1 for NDWI performance.

Parameters

- **test_data** (*RepeatDataset with tf.float32*) –
- **total_steps** (*int/float*) –

Return type

Returns the precision, recall, f1, accuracy metric based on the NDWI performance

tools.metrics_.**MetricCalculator_multiview_2**(*model*, *test_data*, *total_steps*)

This function takes in the multiview-2 model loaded from google cloud bucket, the test_data which is the tensor object and the number of steps and returns the metrics including accuracy, recall, precision and F1

Parameters

- **model** (*keras.engine.functional.Functional*) –
- **test_data** (*RepeatDataset with tf.float32*) –
- **total_steps** (*int/float*) –

Return type
Returns the precision, recall, f1, accuracy metric based on the model performance.

Notes

This function should be used instead of the F1, custom_accuracy written above. The code is obtained/modified from:

https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras

https://neptune.ai/blog/implementing-the-macro-f1-score-in-keras

tools.metrics_.**MetricCalculator_multiview_3**(*model*, *test_data*, *total_steps*)

This function takes in the multiview-3 model loaded from google cloud bucket, the test_data which is the tensor object and the number of steps and returns the metrics including accuracy, recall, precision and F1

Parameters

- **model** (*keras.engine.functional.Functional*) –
- **test_data** (*RepeatDataset with tf.float32*) –
- **total_steps** (*int/float*) –

Return type
Returns the precision, recall, f1, accuracy metric based on the model performance.

Notes

This function should be used instead of the F1, custom_accuracy written above. The code is obtained/modified from:

https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras

https://neptune.ai/blog/implementing-the-macro-f1-score-in-keras

tools.metrics_.**custom_accuracy**(*y_true*, *y_pred*)

The function is used as tensorflow metrics when training. It takes in the ground truth and the model predicted result and evaluate the accuracy score. This is an experimental function and should not be used as further model training metric.

Parameters

- **y_true** (*tf.tensor*) –
- **y_pred** (*tf.tensor*) –

Return type
accuracy score in keras backend

Notes

This function is modified from the F1 metric above to fit the definition of accuracy. However, tensorflow's "categorical_accuracy" is used instead. The accuracy metric would also be recalculated again in MetricCalculator, MetricCalculator_multiview_2 and MetricCalculator_multiview_3.

The reason this function is kept is because the model was initially trained with these metrics and stored in the google cloud bucket. To retrieve the models these metrics must be passed inorder to retrieve the model. Since the model is optimize on the loss rather than the metrics, the incorrect metric would not effect the model training process. The code is obtained/modified from:

https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras

https://neptune.ai/blog/implementing-the-macro-f1-score-in-keras

`tools.metrics_.`**f1**(*y_true*, *y_pred*)

The function is used as tensorflow metrics when training. It takes in the ground truth and the model predicted result and evaluate the F1 score. This is an experimental function and should not be used as further model training metric.

Parameters

- **y_true** (*tf.tensor*) –
- **y_pred** (*tf.tensor*) –

Return type
F1 score in keras backend

Notes

This function is flawed because keras calculates the metrics batchwise which is why F1 metric is removed from keras. To properly calulate the F1 score, we can use the callback function or manually calculate F1 score after the model has finished training. The latter is chosen and this could be seen in MetricCalculator, MetricCalculator_multiview_2 and MetricCalculator_multiview_3.

The reason this function is kept is because the model was initially trained with these metrics and stored in the google cloud bucket. To retrieve the models these metrics must be passed inorder to retrieve the model. Since the model is optimize on the loss rather than the metrics, the incorrect metric would not effect the model training process. The code is obtained/modified from:

https://stackoverflow.com/questions/43547402/how-to-calculate-f1-macro-in-keras

https://neptune.ai/blog/implementing-the-macro-f1-score-in-keras

`tools.metrics_.`**ndwi_threashold**(*B3*, *B5*)

This function takes in bands 3 and bands 5 from the landsat imagery and returns the tuple prediction of whether there is water present or not. The threashold is set at 0.

Parameters

- **test_data** (*RepeatDataset with tf.float32*) –
- **total_steps** (*int/float*) –

Return type
tuple of whether there is water or not

CHAPTER

FOUR

MODEL.PY

class tools.model.**CustomModel**(**args*, ***kwargs*)

This class allows us to create custom model by modifying the functions of interest including the train_step test_step in order to enable the model to take in multilayered inputs. Also, the execution is switched from eager to graph in order to increase the speed of training

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

test_step(*data*)

This function is a standard test_step in tensorflow, but graph execution is used instead. The function takes in the data and return the corresponding metrics

Parameters
data (*tuple of tf.float32/tf.int*) –

Return type
The function returns the corresponding metrics

train_step(*data*)

This function is a standard train_step in tensorflow, but graph execution is used instead. The function takes in the data and return the corresponding metrics

Parameters
data (*tuple of tf.float32/tf.int*) –

Return type
The function returns the corresponding metrics

class tools.model.**CustomModel_multiview_2**(**args*, ***kwargs*)

This class allows us to create custom model by modifying the functions of interest including the train_step test_step in order to enable the model to take in 2 layer inputs for multiview learning. Also, the execution is switched from eager to graph in order to increase the speed of training

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

test_step(*data*)

This function modifies the standard test_step in tensorflow in order to manipulate and split the input data to put into the multiview deep learning model, and graph execution is used instead. The function takes in the data and return the corresponding metrics

Parameters
data (*tuple of tf.float32/tf.int*) –

Return type
The function returns the corresponding metrics

train_step(*data*)

This function modifies the standard train_step in tensorflow in order to manipulate and split the input data to put into the multiview deep learning model, and graph execution is used instead. The function takes in the data and return the corresponding metrics

Parameters
data (*tuple of tf.float32/tf.int*) –

Return type
The function returns the corresponding metrics

class tools.model.**CustomModel_multiview_3**(**args*, ***kwargs*)

This class allows us to create custom model by modifying the functions of interest including the train_step test_step in order to enable the model to take in 3 layer inputs for multiview learning. Also, the execution is switched from eager to graph in order to increase the speed of training

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

test_step(*data*)

This function modifies the standard test_step in tensorflow in order to manipulate and split the input data to put into the multiview deep learning model, and graph execution is used instead. The function takes in the data and return the corresponding metrics

Parameters
data (*tuple of tf.float32/tf.int*) –

Return type
The function returns the corresponding metrics

train_step(*data*)

This function modifies the standard train_step in tensorflow in order to manipulate and split the input data to put into the multiview deep learning model, and graph execution is used instead. The function takes in the data and return the corresponding metrics

Parameters
data (*tuple of tf.float32/tf.int*) –

Return type
The function returns the corresponding metrics

tools.model.**DecoderMiniBlock**(*prev_layer_input*, *skip_layer_input*, *num_filters=32*)

Decoder miniblock will enable creation of all other decoder layers in the get_model function. The function takes in the previous layer inputs, the corresponding encoder and number of filters. The function returns the next layer and the corresponding layer which will be used in decoding later on.

Parameters

- **prev_layer_input** (*tf.float32/tf.int*) –
- **skip_layer_input** (*tf.float32/tf.int*) –
- **num_filters** (*int/float*) –

Returns

- *The function returns the next layer and the corresponding layer which will be used in decoding later on as a*
- *tensor object*

Notes

The code is obtained/modified from:

https://medium.com/geekculture/u-net-implementation-from-scratch-using-tensorflow-b4342266e406

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.model.**EncoderMiniBlock**(*inputs*, *num_filters=32*, *dropout_prob=0.3*, *max_pooling=True*)

Encoder miniblock that will enable creation of all other encoder layers in the get_model function. The function takes in inputs, number of filter, a dropout probability and max_pooling parameter. The function returns the next layer and the corresponding layer which will be used in decoding later on.

Parameters

- **input_tensor** (*tf.float32/tf.int*) –
- **num_filters** (*int/float*) –
- **dropout_prob** (*float*) –
- **max_pooling** (*bool*) –

Returns

- *The function returns the next layer and the corresponding layer which will be used in decoding later on as a*
- *tensor object*

Notes

The code is obtained/modified from:

https://medium.com/geekculture/u-net-implementation-from-scratch-using-tensorflow-b4342266e406

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.model.**conv_block**(*input_tensor*, *num_filters*)

This is processes the tensor right after the encoder to give the center block. The function takes in input tensor and number of filters and returns the next layer which is the center layer

Parameters

- **input_tensor** (*tf.float32/tf.int*) –
- **num_filters** (*int/float*) –

Return type
returns the next layer which is the center layer which is a tensor object

Notes

The code is obtained/modified from:

https://medium.com/geekculture/u-net-implementation-from-scratch-using-tensorflow-b4342266e406

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.model.**get_model**()

This function puts all the previous mini encoders, decoder and conv_block and the modified custom model together in order to compile and return a customized model for feature stack method.

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

tools.model.**get_model_multiview_2**()

This function puts all the previous mini encoders, decoder and conv_block and the modified custom model together in order to compile and return a customized model for multiview learning with 2 inputs

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

tools.model.**get_model_multiview_2_HT**()

This function puts all the previous mini encoders, decoder and conv_block and the modified custom model together in order to compile and return a customized model for multiview learning with 2 inputs. This function is also used in hyperparameter tuning for loss functions and dropouts rate.

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

tools.model.**get_model_multiview_3**()

This function puts all the previous mini encoders, decoder and conv_block and the modified custom model together in order to compile and return a customized model for multiview learning with 3 inputs

Notes

The code is obtained/modified from:

https://towardsdatascience.com/eager-execution-vs-graph-execution-which-is-better-38162ea4dbf6#:~:text=Eager%20execution%20is%20a%20powerful,they%20occur%20in%20your%20code.

https://www.tensorflow.org/guide/keras/customizing_what_happens_in_fit

CHAPTER

FIVE

PREPROCESSING.PY

tools.preprocessing.**EnsureTwodigit**(*number*)

Transform the input month into string in the correct format for date and time. ———- number: int

Return type
months in string.

tools.preprocessing.**GenSeasonalDatesMonthly**(*start*, *end*, *month_frequency=3*)

Given two dictionary containing the key month and year, return two arrays that contains the time between the interval of start and end. ———- start: dict end: dict

Return type
Two arrays containing the time elapsed between start and end

class tools.preprocessing.**Preprocessor**(*config*)

Class that preprocessese and returns the training, evaluation and testing data from google cloud bucket

get_dataset(*pattern*)

Function to read, parse and format to tuple a set of input tfrecord files. Get all the files matching the pattern, parse and convert to tuple. :param pattern: :type pattern: A file pattern to match in a Cloud Storage bucket.

Return type
A tf.data.Dataset

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

get_eval_dataset(*location*)

Get the preprocessed evaluation dataset :param location: :type location: string

Return type
A tf.data.Dataset of evaluation data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

get_test_dataset(*location*, *test_base*)

Get the preprocessed testing dataset :param location: :type location: string

Return type
A tf.data.Dataset of testing data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

get_training_dataset(*location*)

Get the preprocessed training dataset :param location: :type location: string

Return type
A tf.data.Dataset of training data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

get_training_dataset_for_testing(*location*)

Get the preprocessed training dataset for testing :param location: :type location: string

Return type
A tf.data.Dataset of training data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

parse_tfrecord(*example_proto*)

The parsing function Read a serialized example into the structure defined by FEATURES_DICT.

Parameters
example_proto (*a serialized Example*) –

Return type
A dictionary of tensors, keyed by feature name.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

to_tuple(*inputs*)

Function to convert a dictionary of tensors to a tuple of (inputs, outputs). Turn the tensors returned by parse_tfrecord into a stack in HWC shape. :param inputs: :type inputs: A dictionary of tensors, keyed by feature name.

Return type
A tuple of (inputs, outputs).

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.preprocessing.**maskL8sr**(*image*)

image: ee.image.Image

Return type
A maksed landsat-8 ee.image.Image

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

CHAPTER

SIX

SAMPLING.PY

tools.sampling.**Eval_task**(*evalPolys*, *n*, *N*, *arrays*, *setting*, *foldername*)

Exporting Evaluating data to google cloud bucket :param evalPolys: :type evalPolys: ee.featurecollection.FeatureCollection :param n: :type n: int/float :param N: :type N: int/float :param arrays: :type arrays: ee.image.Image :param setting: :type setting: tools.config.configuration :param foldername: :type foldername: string

Return type
A tf.data.Dataset of testing data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.sampling.**Testing_task**(*testPolys*, *n*, *N*, *arrays*, *setting*, *foldername*, *Test_base*)

Exporting Testing data to google cloud bucket :param testPolys: :type testPolys: ee.featurecollection.FeatureCollection :param n: :type n: int/float :param N: :type N: int/float :param arrays: :type arrays: ee.image.Image :param setting: :type setting: tools.config.configuration :param foldername: :type foldername: string

Return type
A tf.data.Dataset of testing data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.sampling.**Training_task**(*trainingPolys*, *n*, *N*, *arrays*, *setting*, *foldername*)

Exporting Training data to google cloud bucket :param trainingPolys: :type trainingPolys: ee.featurecollection.FeatureCollection :param n: :type n: int/float :param N: :type N: int/float :param arrays: :type arrays: ee.image.Image :param setting: :type setting: tools.config.configuration :param foldername: :type foldername: string

Return type
A tf.data.Dataset of testing data.

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

CHAPTER

SEVEN

IMAGES.PY

tools.images.**LoadImage**(*out_image_base*, *user_folder*, *kernel_buffer*, *setting*, *extra_folder=''*)

Load the image from the google cloud bucket and preprocess the image for prediction and provide crutial information for exporting to GEE asset

Parameters

- **out_image_base** (*string*) –
- **user_folder** (*string*) –
- **kernel_buffer** (*list*) –
- **setting** (*dict*) –
- **extra_folder** (*string*) –

Notes

The code is obtained/modified from:

Returns

- *imageDataset as tensor data for prediction, patches as int, x_buffer as int, y_buffer as int, jsonFile as string*
- **https** (*//github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb*)

tools.images.**doExport**(*out_image_base*, *kernel_buffer*, *region*, *setting*, *extra_folder=''*)

Export the image with features and area of interest to google cloud bucket. The function doesn't exit until the task is complete. The optional extra_folder arguement lets you put the exported tf.record.gz in a folder

Parameters

- **out_image_base** (*string*) –
- **kernel_buffer** (*list*) –
- **region** (*ee.Geometry.BBox*) –
- **setting** (*dict*) –
- **extra_folder** (*string*) –

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**doPrediction_featurestack**(*out_image_base*, *user_folder*, *kernel_buffer*, *model*, *suffix*, *setting*, *extra_folder=''*)

Putting all the image functions together. Load the Image, predict the output with featurestack U-Net, return information ready to be exported to GEE asset ————- pattern: A file pattern to match in a Cloud Storage bucket.

Returns

- *A out_image_asset a location to the output GEE asset folder*
- *as a string, out_image_file of the prediction as TFRecord and the*
- *corresponding json file.*

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**doPrediction_multiview_2**(*out_image_base*, *user_folder*, *kernel_buffer*, *model*, *suffix*, *setting*, *extra_folder=''*)

Putting all the image functions together. Load the Image, predict the output with Multi-view U-Net (2 inputs), return information ready to be exported to GEE asset ————- pattern: A file pattern to match in a Cloud Storage bucket.

Returns

- *A out_image_asset a location to the output GEE asset folder*
- *as a string, out_image_file of the prediction as TFRecord and the*
- *corresponding json file.*

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**doPrediction_multiview_3**(*out_image_base*, *user_folder*, *kernel_buffer*, *model*, *suffix*, *setting*, *extra_folder=''*)

Putting all the image functions together. Load the Image, predict the output with Multi-view U-Net (3 inputs), return information ready to be exported to GEE asset ————- pattern: A file pattern to match in a Cloud Storage bucket.

Returns

- *A out_image_asset a location to the output GEE asset folder*
- *as a string, out_image_file of the prediction as TFRecord and the*

- *corresponding json file.*

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**predictionMultipleinput**(*model*, *imageDataset*, *patches*, *setting*)

Given the model, and image for prediction, predict the image with Multi-view learning U-Net

Parameters

- **model** (*model : keras.engine.functional.Functional*) –
- **imageDataset** (*tf.data.Dataset of training data (BatchDataset)*) –
- **patches** (*int*) –
- **setting** (*dict*) –

Return type
A tf.float32 with same dimension as imageDataset

Notes

The code is modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**predictionMultipleinput_3**(*model*, *imageDataset*, *patches*, *setting*)

Given the model, and image for prediction, predict the image with Multi-view learning U-Net with 3 inputs

Parameters

- **model** (*model : keras.engine.functional.Functional*) –
- **imageDataset** (*tf.data.Dataset of training data (BatchDataset)*) –
- **patches** (*int*) –
- **setting** (*dict*) –

Return type
A tf.float32 with same dimension as imageDataset

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**predictionSingleinput**(*model*, *imageDataset*, *patches*)

Given the model, and image for prediction, predict the image with feature stack U-Net

Parameters

- **model** (*model : keras.engine.functional.Functional*) –

- **imageDataset** (*tf.data.Dataset of training data (BatchDataset)*) –
- **patches** (*int*) –

Return type
A tf.float32 with same dimension as imageDataset

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

tools.images.**uploadToGEEAsset**(*x_buffer*, *y_buffer*, *predictions*, *out_image_base*, *jsonFile*, *suffix*, *setting*, *multiview=False*, *user_folder='users/mewchayutaphong'*)

Given the predictions, exported file location other information on the image to be exported, return the required information in order to export the predicted image to the GEE asset

Parameters

- **x_buffer** (*int*) –
- **y_buffer** (*int*) –
- **predictions** (*tf.data.Dataset of training data (BatchDataset)*) –
- **out_image_base** (*string*) –
- **jsonFile** (*string*) –
- **suffix** (*string*) –
- **setting** (*dict*) –
- **user_folder** (*string*) –

Returns

- *A out_image_asset a location to the output GEE asset folder*
- *as a string, out_image_file of the prediction as TFRecord and the*
- *corresponding json file.*

Notes

The code is obtained/modified from:

https://github.com/google/earthengine-api/blob/master/python/examples/ipynb/UNET_regression_demo.ipynb

CHAPTER

EIGHT

LOSSES_.PY

tools.losses_.**dice_coef**(*y_true*, *y_pred*, *smooth=1*)

Recieve the true and predicted tensor and return the resulting dice loss to prevent overfitting. ———- y_true: tf.float32 y_pred: tf.float32 smooth: int/float

Return type
A tf.float32 with same dimension as input tf.float32

Notes

The code is obtained/modified from:

https://www.kaggle.com/code/kmader/u-net-with-dice-and-augmentation/notebook

tools.losses_.**dice_p_cc**(*in_gt*, *in_pred*)

in_gt: tf.float32 in_pred: tf.float32

Return type
A tf.float32 with same dimension as input tf.float32

Notes

The code is obtained/modified from:

https://www.kaggle.com/code/kmader/u-net-with-dice-and-augmentation/notebook

References

PYTHON MODULE INDEX

t

tools, ??
tools.config, ??