IRP-bp221

Release 1.0

Bo Pang

Aug 31, 2022

CONTENTS:

CHAPTER

ONE

FIDN

1.1 FIDN package

1.1.1 Subpackages

FIDN.models package

Submodules

FIDN.models.densenet module

Reference:

- Keras Applications
- Densely Connected Convolutional Networks (CVPR 2017)

`FIDN.models.densenet.`**`FIDNEncoder`**`(`*blocks*, *input_shape=None*, *channel_last=True*, *name='FIDN'*`)`

Instantiates the FIDN Encoder architecture.

Parameters

- **`blocks`** – numbers of building blocks for the four dense layers.
- **`input_shape`** – optional shape tuple.
- **`channel_last`** – Indicates whether the last dimension of the image represents channel.
- **`name`** – Name of the model, default is FIDN

Returns

A keras.Model instance.

`FIDN.models.densenet.`**`conv_block`**`(`*x*, *growth_rate*, *name*, *channel_last=True*`)`

A building block for a dense block.

Parameters

- **`x`** – input tensor.
- **`growth_rate`** – float, growth rate at dense layers.
- **`name`** – string, block label.
- **`channel_last`** – Indicates whether the last dimension of the image represents channel.

Returns
Output tensor for the block.

`FIDN.models.densenet.`**`dense_block`**`(`*x*, *blocks*, *name*, *channel_last=True*`)`

A dense block.

Parameters

- **`x`** – input tensor.
- **`blocks`** – integer, the number of building blocks.
- **`name`** – string, block label.
- **`channel_last`** – Indicates whether the last dimension of the image represents channel.

Returns
Output tensor for the block.

`FIDN.models.densenet.`**`transition_block`**`(`*x*, *reduction*, *name*, *channel_last=True*`)`

A transition block.

Parameters

- **`x`** – input tensor.
- **`reduction`** – float, compression rate at transition layers.
- **`name`** – string, block label.
- **`channel_last`** – Indicates whether the last dimension of the image represents channel.

Returns
output tensor for the block.

FIDN.models.fidn module

`FIDN.models.fidn.`**`build_fidn_model`**`(`*inp*`)`

Construction of the FIDN model, see report/bp221-final-report.pdf for details of the model structure

Parameters
`inp` – Dimensions of the model input

Returns
A keras instance of the FIDN model

Module contents

1.1.2 Submodules

1.1.3 FIDN.dataset module

`FIDN.dataset.`**`create_shifted_frames`**`(`*data*`)`

Helper function for splitting a single piece of data into source and target

Parameters
`data` – Individual data to be split

Returns
Source data x, Target data y

`FIDN.dataset.`**`get_big_fire`**`(`*dataset*`)`

Filter the dataset for fires with a variation of >1000 between day 2 and final

Parameters
`dataset` – Data set to be filtered

Returns
Filtered data set

`FIDN.dataset.`**`load_dataset`**`(`*fpath*`)`

Reads data sets from files and normalises geographical and meteorological information.

Parameters
`fpath` – Path of the data set in the file system

Returns
Entire data set

`FIDN.dataset.`**`normalize`**`(`*data*, *scale_range=None*`)`

Min Max scale for a batch of data

Parameters

- **`data`** – The data to be processed
- **`scale_range`** – Scaling range

Returns
The data normalised to scale_range

`FIDN.dataset.`**`setup_dataset`**`(`*data_path*`)`

Read the dataset, normalise it. After dividing the dataset, data augmentation is performed on the training set, and finally the dataset is split into Source and Target.

Parameters
`data_path` – Path of the data set in the file system

Returns
full dataset, train_dataset, val_dataset, test_dataset, x_train, y_train, x_val, y_val, x_test, y_test

`FIDN.dataset.`**`split_dataset`**`(`*dataset*`)`

Split the dataset into training, validation and test sets in a ratio of 8:1:1

Parameters
`dataset` – Original full dataset

Returns
Training set, validation set, test set

1.1.4 FIDN.evaluate module

`FIDN.evaluate.`**`evaluate_and_save_pic`**`(`*model*, *config*, *dataset*`)`

Use the input model to make predictions on the input dataset and store pictures of the prediction results. The model output is compared with the true values, and metrics such as MSE, RRMSE, SSIM and PSNR are calculated and stored in a csv file.

Parameters

- **`model`** – Model to be evaluated
- **`config`** (*`dict`*) – Output-related configuration

- **dataset** – Data set used for the evaluation

Returns
A pandas DataFrame that stores the model's performance metrics on each sample

FIDN.evaluate.**load_model**(*model_path*)

Reading model from files

Parameters
model_path – Storage paths for models

Returns
A keras model

FIDN.evaluate.**visual_origin_data**(*dataset*, *index*)

Plot each layer of the dataset onto a single image

Parameters

- **dataset** – Raw data set to be read
- **index** – Index of fire events in the dataset

Returns
A figure with 15 subfigures

1.1.5 FIDN.losses module

FIDN.losses.**custom_mean_squared_error**(*y_true*, *y_pred*)

Calculate the mean square error after scaling each pixel of the image to the interval 0 to 1

loss = mean(square(y_true - y_pred))

Parameters

- **y_true** – Ground truth values. shape = *[batch_size, d0, .. dN]*.
- **y_pred** – The predicted values. shape = *[batch_size, d0, .. dN]*.

Returns
Mean squared error values of batch.

FIDN.losses.**psnr_metrics**(*y_true*, *y_pred*)

Calculate the Peak signal-to-noise ratio after scaling each pixel of the image to the interval 0 to 1

The last three dimensions of input are expected to be [height, width, depth].

Parameters

- **y_true** – Ground truth values. First set of images.
- **y_pred** – The predicted values. Second set of images.

Returns
Mean PSNR values of batch.

FIDN.losses.**relative_root_mean_squared_error**(*y_true*, *y_pred*)

Calculate the relative root mean square error.

loss = sqrt(sum(square(y_true - y_pred)) / sum(square(y_true)))

Parameters

- **y_true** – Ground truth values. shape = *[batch_size, d0, .. dN]*.

- **`y_pred`** – The predicted values. shape = *[batch_size, d0, .. dN]*.

Returns
Relative root mean squared error values of batch.

`FIDN.losses.`**`scale2range`**(*x*, *range*)

Scale x into a range, both expected to be floats (Numpy version)

Parameters

- **`x`** (*`np.ndarray`*) – input array.
- **`range`** (*`list`*) – list Range of data to be scaled to

Returns
Output tensor for the block.

`FIDN.losses.`**`scale2range_tf`**(*x*, *range*)

Scale x into a range, both expected to be floats (Tensorflow version)

Parameters

- **`x`** (*`tf.Tensor`*) – input tenspr.
- **`range`** (*`list`*) – Range of data to be scaled to

Returns
Output tensor for the block.

`FIDN.losses.`**`ssim_metrics`**(*y_true*, *y_pred*)

Calculate the structural similarity after scaling each pixel of the image to the interval 0 to 1

This function is based on the standard SSIM implementation from: Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing.

Parameters

- **`y_true`** – Ground truth values. 4-D Tensor of shape [batch, height, width, channels] with only Positive Pixel Values.
- **`y_pred`** – The predicted values. 4-D Tensor of shape [batch, height, width, channels] with only Positive Pixel Values.

Returns
Mean SSIM values of batch.

1.1.6 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX