CNN_Volcanic_deformation

August 9, 2021

```
<h1> Tutorial 1 </h1>
<h2> Classification of Volcanic Deformation using Convolutional Neural Networks </h2>
```

0.1 Overview

This tutorial is based on work done by Matthew Gaddes and full code VUDLNet_21[1]. Creating a Convolutional Neural Network that will detect and localise deformation in Sentinel-1 Interferogram. A database of labelled Sentinel-1 data hosted at VolcNet is used to train the CNN.

A small subset of the data required is provided to create a tutorial that will both run in a short time frame and not be over file size limits set by GitHub. An option is included to download and use larger bottleneck files created from a larger data set to see a higher performance example.

[1] Matthew Gaddes, Andy Hooper , Fabien Albino 2021

0.1.1 Summary for non voclanologists

Interferograms are ground deformation maps produced by interferometric4synthetic aperture radar (InSAR). Specific deformation patterns are associated with different types of sources of deformation (volcanic activity). A trained person can, by eye, identify the source and location of deformation. However, when looking at a large number of interferograms automation is required.

taken from USGS

Convolutional Neural Networks

This tutorial will use Convolutional Neural Networks to classify volcanic deformation.

0.2 The very basics

If you know nothing about neural networks there is a toy neural network python code example included in the LIFD ENV ML Notebooks Repository. Creating a 2 layer neural network to illustrate the fundamentals of how Neural Networks work and the equivlent code using the python machine learning library keras.

0.3 Recommended reading

The in-depth theory behind convolution neural networks will not be covered here as this tutorial is focusing on how to use them for a certain earth science application. If you wish to learn more here are some great starting points.

1. The very basics in a Victor Zhou Blog
2. A deep dive into CNNs in towards data science

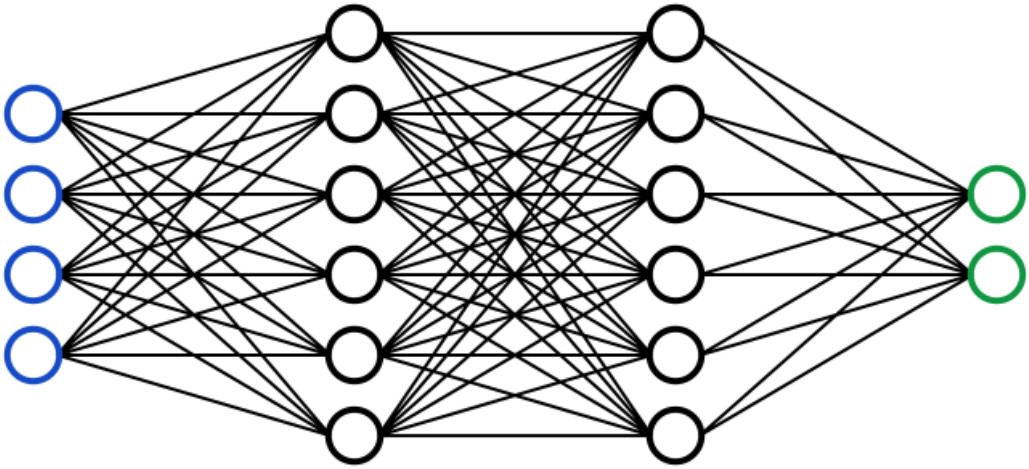
3. More information on transfer learning (using pre-trained models)
4. Section 5 of the example ipython notebooks from the fchollet deep learning with python repository

Machine Learning Theory

To create a convolutional neural network that both classify the type of deformation and the location a "two-headed model" is built to return both from one forward pass of an interferogram through the network. As models have already been trained to identify features it is possible to transfer weights from other models designed for different problems. In this tutorial, the VGG16 model is used as this was found to be sensitive to the signals of interest in interferograms.

0.4 Convolutional Neural Networks

A Neural Network is essentially a mathematical function that maps a given input to the desired output by adjusting weights and biases over many layers.



Convolutional Neural Networks (CCNs) are a popular variant of neural networks used for image classification. CNNs do not require as many weights as a normal neural network, they use the fact that each pixels neighbour provides some context to allow for localised features to be detected. CNN's use convolutions to create a set of filters for each layer to create an output image by convolving a set of filters with the input image to create an output volume.

CNNs go through a set of steps.

1. Convolve the input image with a set of filters to create output volumes
2. Pool layers as much of the information contained in each layer's output is redundant as neighbouring pixels produce similar values. (This essentially reduces the output volume)
3. Create a softmax layer fully connected (dense) layer to predict the outcome with the highest probability.

0.5 VGG16 Model

Deep CCNs can take days or weeks to train if using very large datasets. This process can be shortened by re-using model weights from pre-trained models that were developed for standard computer vision benchmark datasets. i.e. some layers of a model trained on one problem that is similar to the actual problem that you're interested in can be used as a shortcut in training your model.

The VGG16 model was developed by the Visual Graphics Group (VGG) at Oxford and works on recognising consistent and repeating structures. Here we will use 5 convolutional blocks of VGG16 on the input data before using a home built fully connected CNN. VGG16 is easily accessible via `Keras`.

Python

Basic python knowledge is assumed for this tutorial. A number of complex data processing and visualisation functions have been written and stored in aux_functions.py (code taken from VUDL-Net_21). For this tutorial the main machine learning library we'll be working Keras. Python specific information will be colour coded blue.

0.6 Keras

There are many machine learning python libraries available, Keras is one such library although it now comes bundled with TensorFlow and the recommendation is to use TensorFlow's Keras API instead of the standalone `Keras` package. Keras is a high-level API designed to make using a more low-level library like TensorFlow easier and more intuitive to use so you don't need to know about the hardware you're using or some of the more technical details of the models you're using. Throughout this tutorial, you will see some complex machine learning tasks executed in just a few lines of code by calling Keras functions.

Requirements

These notebooks should run

Python Packages:

- Python 3
- Keras
- tensorflow
- pydot
- graphviz
- ipdb
- matplotlib=3.0
- basemap-data-hires
- geopy

Data Requirements

This notebook referes to some data included in the git hub repositroy

Contents:

3. Merge and rescale synthetic data
4. Compute bottleneck features.
5. Train fully connected network
6. Fine tune the fully connected network and the 5th convolutional block.

Load in all required modules (includig some auxillary code) and turn off warnings. Make sure Keras session is clear

[1]:
```
# For readability: disable warnings
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# import modules
# general file system utilites
import sys
import glob
import os
from pathlib import Path
# Maths and
import numpy as np
import numpy.ma as ma
# Premade data is provided as pickles
import pickle
# Plotting utilies
import matplotlib
import matplotlib.pyplot as plt
import shutil
# Machine learning Library Keras
from tensorflow import keras
#from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from tensorflow.keras import backend as K
from tensorflow.keras import losses, optimizers
from tensorflow.keras.applications.vgg16 import VGG16
from keras.utils.vis_utils import plot_model
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Dense, Dropout, Flatten
# import axillary plotting functions
# these functions
from aux_functions import *
```

[3]:
```
# Clear Keras session
K.clear_session()
```

1 Load In Provided Data

The model will be trained on a large dataset of synthetic interferograms which feature labels of both the type and location of any deformation. The performance improved by including a small

amount of augmented real Sentinel-1 data.

The first steps to prepare the data to have to be pre-made synthetic interferograms are provided in the `data` folder.

Sythetic Interferograms

As this is a tutorial focusing on Machine learning the Synthetic Interferograms are provided as pickle files. These files were genrated using SyInteferoPy. SyInteferoPy generates synthetic images similar to those produced by Sentinel-1 satellites from the SRTM3 digital elevation model (DEM) [2]

[2] Gaddes & Bagnardi 2019

If you wanted to generate your own synthetic data you would need to use the tools in SyInteferoPy. SyInteferoPy and the pickled volcano dem data `data/volcano_dems.pkl`

```
git submodule add https://github.com/matthew-gaddes/SyInterferoPy SyInterferoPy
dependency_paths = {'syinterferopy_bin' : 'SyInterferoPy/lib/}
sys.path.append(dependency_paths['syinterferopy_bin'])
```

```
from random_generation_functions import create_random_synthetic_ifgs
os.mkdir(Path(f"./data/synthetic_data/"))
for file_n in range(synthetic_ifgs_n_files):
    print(f"Generating file {file_n} of {synthetic_ifgs_n_files} files.  ")
    X_all, Y_class, Y_loc, Y_source_kwargs = create_random_synthetic_ifgs(volcano_dems,
                                                                          **synthetic_ifgs_settings)
    # convert to one hot encoding (from class labels)
    Y_class = keras.utils.to_categorical(Y_class, len(synthetic_ifgs_settings['defo_sources'])
                                                        dtype='float32')
    with open(Path(f'./data/synthetic_data/data_file_{file_n}.pkl'), 'wb') as f:
        pickle.dump(X_all[synthetic_ifgs_settings['outputs'][0]], f)
        pickle.dump(Y_class, f)
        pickle.dump(Y_loc, f)
    f.close()
    del X_all, Y_class, Y_loc
# output the settings as a text file so that we know how data were generated in the future.
with open(f"./data/synthetic_data/synth_data_settings.txt", 'w') as f:
    print(f"Number of data per file : {ifg_settings['n_per_file']}" ,file = f)
    print(f"Number of files: {synthetic_ifgs_n_files}" ,file = f)
    for key in synthetic_ifgs_settings:
        print(f"{key} : {synthetic_ifgs_settings[key]}", file = f)
```

```
<h3>Settings for generating interfeorgrams.</h3>
```

Passing this information to SyInteferoPy. SyInteferoPy will generate 650 synthetic Interfograms

- **n_per_file:** number of ifgs per data file.

- **synthetic_ifgs_n_files:** numer of files of synthetic data
- **defo_sources:** deformation patterns that will be included in the dataset.

- **n_ifgs:** the number of synthetic interferograms to generate PER FILE
- **n_pix:** number of 3 arc second pixels (~90m) in x and y direction
- **outputs:** channel outputs. uuu = unwrapped across all 3
- **intermediate_figure:** if True, a figure showing the steps taken during creation of each ifg is displayed.

- **cov_coh_scale:** The length scale of the incoherent areas, in meters. A smaller value creates smaller patches, and a larger one creates larger pathces.

- **coh_threshold:** if 1, there are no areas of incoherence, if 0 all of ifg is incoherent.

- **min_deformation:** deformation pattern must have a signals of at least this many metres.

- **max_deformation:** deformation pattern must have a signal no bigger than this many metres.

- **snr_threshold signal:** to noise ratio (deformation vs turbulent and topo APS) to ensure that deformation is visible. A lower value creates more subtle deformation signals.
- **turb_aps_mean:** turbulent APS will have, on average, a maximum strenghto this in metres (e.g 0.02 = 2cm)
- **turb_aps_length:** turbulent APS will be correlated on this length scale, in metres.

[4]:
```
# Define some settings (outlined above)
ifg_settings            = {'n_per_file'         : 50}    # number of ifgs per␣
 ↪data file.
synthetic_ifgs_n_files  =  13                            # numer of files of␣
 ↪synthetic data
synthetic_ifgs_settings = {'defo_sources'       : ['dyke', 'sill', 'no_def'],
                           'n_ifgs'             : ifg_settings['n_per_file'],
                           'n_pix'              : 224,
                           'outputs'            : ['uuu'],
                           'cov_coh_scale'      : 5000,
                           'coh_threshold'      : 0.7,
                           'min_deformation'    : 0.05,
                           'max_deformation'    : 0.25,
                           'snr_threshold'      : 2.0,
                           'turb_aps_mean'      : 0.02,
                           'turb_aps_length'    : 5000}
n_synth_data = ifg_settings['n_per_file'] * synthetic_ifgs_n_files
print('\nDetermining if files containing the synthetic deformation patterns␣
 ↪exist...\n ', end = '')
synthetic_data_files = glob.glob(str(Path(f"./data/synthetic_data/*.pkl")))
if len(synthetic_data_files) == synthetic_ifgs_n_files:
    print(f"\nThe correct number of files were found ({synthetic_ifgs_n_files})␣
 ↪so no new ones will be generated.  "
          f"\nHowever, this doesn't guarantee that the files were made using␣
 ↪the settings in synthetic_ifgs_settings."
```

```
        f"\nCheck synth_data_settings.txt to be sure.   ")
else:
    print(f"\nCheck for pickles- do you have the data files required (in GitHub␣
 ↪Repo)")
```

```
Determining if files containing the synthetic deformation patterns exist…

The correct number of files were found (13) so no new ones will be generated.
However, this doesn't guarantee that the files were made using the settings in
synthetic_ifgs_settings.
Check synth_data_settings.txt to be sure.
```

Load in Real Data

Included in this repository is a git submodule VolcNet which is a set of 250 labelled unwrapped interferograms that contain labels of both the type of deformation (including examples of no deformation) and the location of deformation within the interferograms. In the form of pickle files, i.e. interferograms have been stored as masked NumPy arrays and labelled with location and deformation source.

If you have not already then in your repositoy directory please run the following code.

```
git submodule init
git submodule update --init --recursive
```

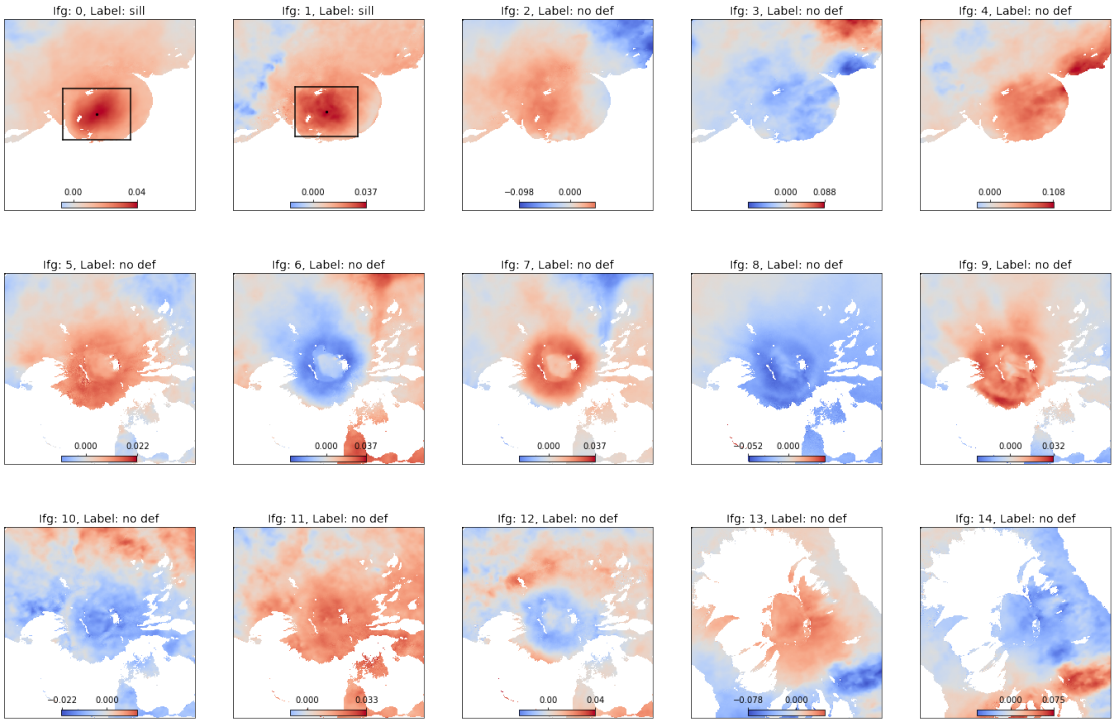
This labled data will be used to train our model.

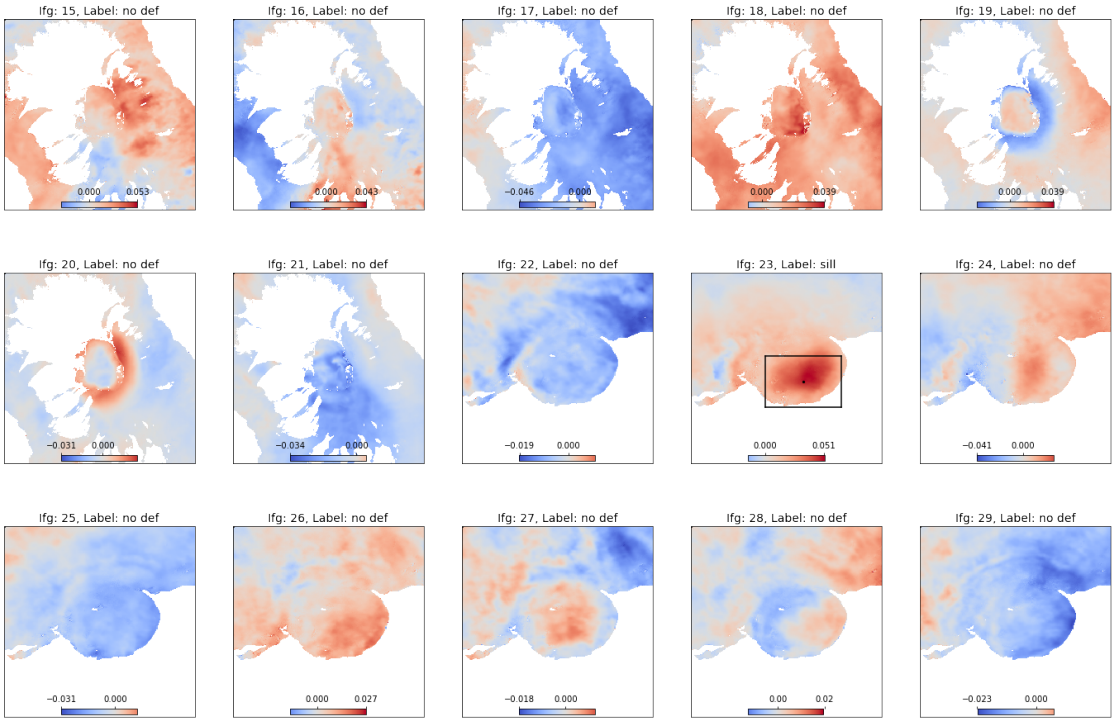
The below code checks for the Volcnet files and uses plotting functions in the provided `aux_functions.py` to show the data.

For an example file it will show a plot of the interferrogram and give the lable of the deformation source (if applicable).

[5]:
```
# Load the real data
# Note that these are in metres, and use one hot encoding for the class,
# and are masked arrays (incoherence and water are masked)
VolcNet_path = Path('./VolcNet')
# factor to auument by.  E.g. if set to 3 and there are 250 data, there will be␣
 ↪650 augmented
real_ifg_settings       = {'augmentation_factor' : 3}
#  get a list of the paths to all the VolcNet files
VolcNet_files = sorted(glob.glob(str(VolcNet_path / '*.pkl')))
if len(VolcNet_files) == 0:
    raise Exception('No VolcNet files have been found.'  +
                    'Perhaps the path is wrong? Or perhaps you only want to use␣
 ↪synthetic data?'+
                    'In which case, this section can be removed.  Exiting...')

X_1s = []
Y_class_1s = []
```

```
Y_loc_1s = []
for VolcNet_file in VolcNet_files:
    X_1, Y_class_1, Y_loc_1 = open_VolcNet_file(VolcNet_file,␣
 ↪synthetic_ifgs_settings['defo_sources'])
    X_1s.append(X_1)
    Y_class_1s.append(Y_class_1)
    Y_loc_1s.append(Y_loc_1)
X = ma.concatenate(X_1s, axis = 0)
Y_class = np.concatenate(Y_class_1s, axis = 0)
Y_loc = np.concatenate(Y_loc_1s, axis = 0)
del X_1s, Y_class_1s, Y_loc_1s, X_1, Y_class_1, Y_loc_1
# plot the data in it (note that this can be across multiople windows)
plot_data_class_loc_caller(X[:30,], Y_class[:30,], Y_loc[:30,], source_names =␣
 ↪['dyke', 'sill', 'no def'], window_title = 'Sample of Real data')
```





2 Augment Real Data

To improve the performance of the model real data is incorporated. Because we can't include as much real data as the synthetic data we must 'augment' the data into the same size as the number of synthetic interferograms by creating random flips, rotations, and translations

As this augmentation is purely data manipulation we'll use some functions from `aux_functions.py` to help augment the set of 250 interferograms and generate a set of 650 augmented interferograms. The below code will generate pickle files of the augmented data that you can reuse so this step only needs to be done once.

If you've already done this the below code will check if the pickles already exist and only calculated the augmented data if required

[6]:
```
n_augmented_files = int((X.shape[0] * real_ifg_settings['augmentation_factor'])␣
↪/ ifg_settings['n_per_file'])                    # detemine how many files␣
↪will be needed, given the agumentation factor.
print('    Determining if files containing the augmented real data exist.')
real_augmented_files = glob.glob(str(Path(f"./data/real/augmented/*.pkl")))    ␣
↪        #
if len(real_augmented_files) == n_augmented_files:
    print(f"    The correct number of augmented real data files were found␣
↪({n_augmented_files}) "
          f"so no new ones will be generated.  "
```

```
          f"However, this doesn't guarantee that the files were made using the␣
↪current real data.  ")
else:
        try:
            shutil.rmtree(str(Path(f"./data/real/augmented/")))
        except:
            pass
        os.mkdir((Path(f"./data/real/")))
        os.mkdir((Path(f"./data/real/augmented/")))
        print(f"There are {X.shape[0]} real data and the augmentation factor is␣
↪set" +
              f"to {real_ifg_settings['augmentation_factor']}.  ")
        print(f"    With {ifg_settings['n_per_file']} data per file, the␣
↪nearest integer" +
              f"number of files is {n_augmented_files}.  ")
        # loop through each file that is to be made
        for n_augmented_file in range(n_augmented_files):
            print(f'    File {n_augmented_file} of {n_augmented_files}...', end␣
↪= '')
            X_sample, Y_class_sample, Y_loc_sample = choose_for_augmentation(X,␣
↪Y_class, Y_loc,                                        # make a new␣
↪selection of the data with balanced classes
                                                                              ␣
↪n_per_class = int(X.shape[0] / Y_class.shape[1]))          # set it so there␣
↪are as many per class as there are (on average) for the real data.
            X_aug, Y_class_aug, Y_loc_aug = augment_data(X_sample,␣
↪Y_class_sample, Y_loc_sample,                                  #␣
↪augment the sample of real data
                                                        n_data =␣
↪ifg_settings['n_per_file'])                                     # make␣
↪as many new data as are set to be in a single file.

            with open(f"./data/real/augmented/data_file_{n_augmented_file}.
↪pkl", 'wb') as f:                                       # save the output␣
↪as a pickle
                pickle.dump(X_aug, f)
                pickle.dump(Y_class_aug, f)
                pickle.dump(Y_loc_aug, f)
            f.close()
            print('Done!')
        # fill variable with new generated files
        real_augmented_files = glob.glob(str(Path(f"./data/real/augmented/*.
↪pkl")))
        print('Done!')
```

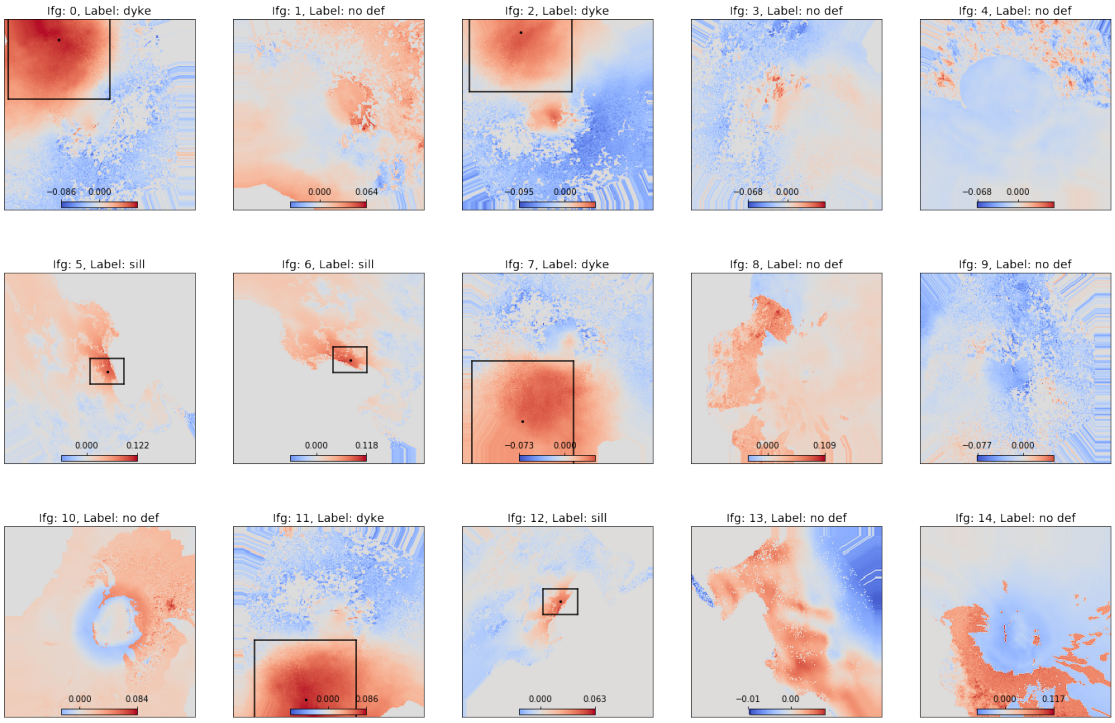
```
Determining if files containing the augmented real data exist.
The correct number of augmented real data files were found (13) so no new
```

ones will be generated. However, this doesn't guarantee that the files were made using the current real data.

Plot Augmented Data

Check the produced data looks sensible, this should look similar to our real data just different transformations

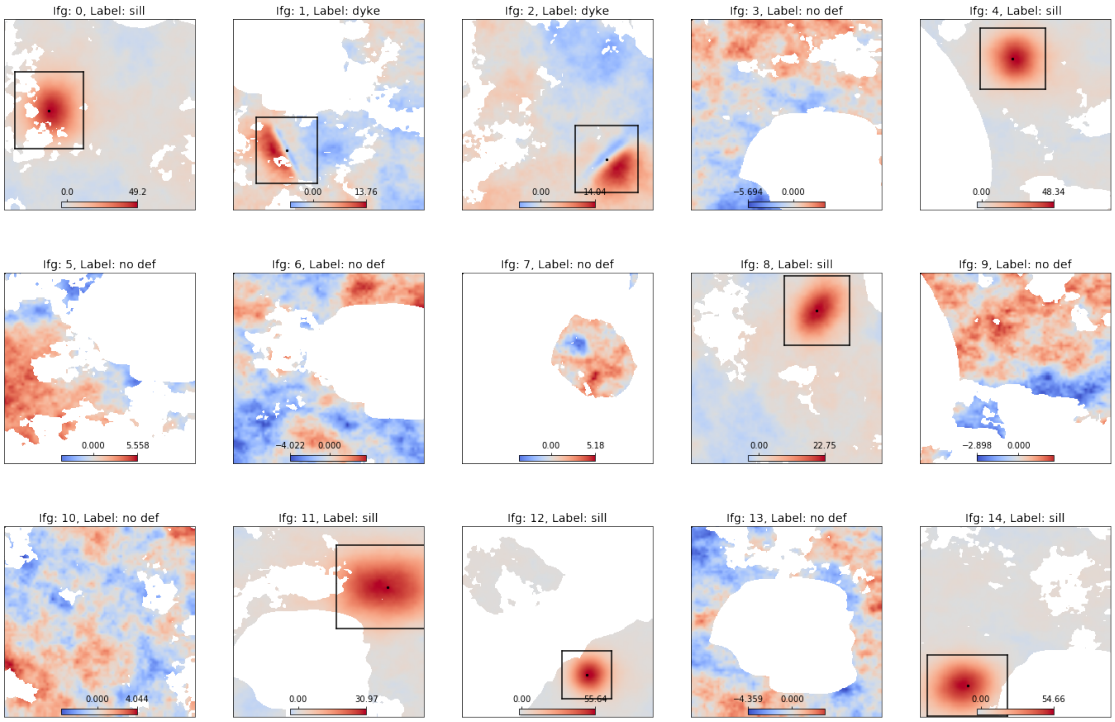
[7]:
```
open_datafile_and_plot("./data/real/augmented/data_file_0.pkl", n_data = 15,␣
↪window_title = '03 Sample of augmented real data')
```



Plot Synthetic Data

Check the Synthetic data looks sensible, this should also look similar to our real data!

[8]:
```
open_datafile_and_plot(f"data/synthetic_data/data_file_0.pkl", n_data = 15,
                       window_title ='01 Sample of synthetic data')  # open and␣
↪plot the data in 1 file
```



Merging real and synthetic inteferograms and rescaling to CCN's input range

First, we're going to merge our two datasets and format them into an output range suitable for the CNN used. E.g. our data might be in meters and rads and we need to rescale to values in the RGB range 0-255 (python's first indice is 0 so 0 -255 gives 256 values)

[9]:
```
cnn_settings = {'input_range': {'min':0, 'max':255}}
```

[10]:
```
#%%
def merge_and_rescale_data(synthetic_data_files, real_data_files, output_range␣
↪= {'min':0, 'max':225}):
    """ Given a list of synthetic data files and real data files (usually the␣
↪augmented real data),

    Inputs:
        synthetic_data_files | list of Paths or string | locations of the .pkl␣
↪files containing the masked arrays
        reak_data_files      | list of Paths or string | locations of the .pkl␣
↪files containing the masked arrays
        output_range         | dict                    | min and maximum of␣
↪each channel in each image.
                                                         Should be set to suit␣
↪the CNN being used.
    Returns:
```

```
        .npz files in step_04_merged_rescaled_data
    History:
        2020_10_29 | MEG | Written
        2021_01_06 | MEG | Fix bug in that mixed but not rescaled data was␣
↪being written to the numpy arrays.
    """
    def data_channel_checker(X, n_cols = None, window_title = None):
        """ Plot some of the data in X.   All three channels are shown.
        """
        if n_cols == None:        # if n_cols is None, we'll plot all the data
            n_cols = X.shape[0]    # so n_cols is the number of data
            plot_args = np.arange(0, n_cols) # and we'll be plotting each of␣
↪them
        else:
            plot_args = np.random.randint(0, X.shape[0], n_cols)        # else,␣
↪pick some at random to plot
        f, axes = plt.subplots(3,n_cols)
        if window_title is not None:
            f.canvas.set_window_title(window_title)
        for plot_n, im_n in enumerate(plot_args):                              #␣
↪loop through each data (column)
            axes[0, plot_n].set_title(f"Data: {im_n}")
            for channel_n in range(3):                                         #␣
↪loop through each row
                axes[channel_n, plot_n].imshow(X[im_n, :,:,channel_n])
                if plot_n == 0:
                    axes[channel_n, plot_n].set_ylabel(f"Channel {channel_n}")

    if len(synthetic_data_files) != len(real_data_files):
        raise Exception('This funtion is only designed to be used when the␣
↪number of real and synthetic data files are the same.  Exiting.  ')

    n_files = len(synthetic_data_files)
    out_file = 0
    try:
            shutil.rmtree(str(Path(f"./data/merged_out/")))
    except:
        pass
    os.mkdir((Path(f"./data/merged_out")))#
    for n_file in range(n_files):
        print(f'    Opening and merging file {n_file} of each type... ', end =␣
↪'')
        with open(real_data_files[n_file], 'rb') as f:       # open the real␣
↪data file
            X_real = pickle.load(f)
            Y_class_real = pickle.load(f)
```

```
            Y_loc_real = pickle.load(f)
        f.close()

        with open(synthetic_data_files[n_file], 'rb') as f:       # open the␣
↪synthetic data file
            X_synth = pickle.load(f)
            Y_class_synth = pickle.load(f)
            Y_loc_synth = pickle.load(f)
        f.close()

        X = ma.concatenate((X_real, X_synth), axis = 0)        # concatenate␣
↪the data
        Y_class = ma.concatenate((Y_class_real, Y_class_synth), axis = 0)    #␣
↪and the class labels
        Y_loc = ma.concatenate((Y_loc_real, Y_loc_synth), axis = 0)        #␣
↪and the location labels

        mix_index = np.arange(0, X.shape[0])          # mix them, get a lis of␣
↪arguments for each data
        np.random.shuffle(mix_index)            # shuffle the arguments
        X = X[mix_index,]               # reorder the data using the shuffled␣
↪arguments
        Y_class = Y_class[mix_index]     # reorder the class labels
        Y_loc = Y_loc[mix_index]      # and the location labels
        # resacle the data from metres/rads etc. to desired input range of cnn␣
↪(e.g. [0, 255]),
        # and convert to numpy array
        X_rescale = custom_range_for_CNN(X, output_range, mean_centre = False)
        data_mid = int(X_rescale.shape[0] / 2)
        np.savez(f'data/merged_out/data_file_{out_file}.npz',
                 X = X_rescale[:data_mid,:,:,:],
                 Y_class= Y_class[:data_mid,:],
                 Y_loc = Y_loc[:data_mid,:])           # save the first half of␣
↪the data
        out_file += 1                                                          ␣
↪                                                                              ␣
↪           # after saving once, update
        np.savez(f'data/merged_out/data_file_{out_file}.npz',
                 X = X_rescale[data_mid:,:,:,:],
                 Y_class= Y_class[data_mid:,:],
                 Y_loc = Y_loc[data_mid:,:])           # save the second half␣
↪of the data
        out_file += 1                                                          ␣
↪                                                                              ␣
↪           # and after saving again, update again.
        print('Done.  ')
```

```
def expand_to_r4(r2_array, shape = (224,224)):
    """
    Calcaulte something for every image and channel in rank 4 data (e.g.␣
↪100x224x224x3 to get 100x3)
    Expand new rank 2 to size of original rank 4 for elemtiwise operations
    """

    r4_array = r2_array[:, np.newaxis, np.newaxis, :]
    r4_array = np.repeat(r4_array, shape[0], axis = 1)
    r4_array = np.repeat(r4_array, shape[1], axis = 2)
    return r4_array

def custom_range_for_CNN(r4_array, min_max, mean_centre = False):
    """ Rescale a rank 4 array so that each channel's image lies in custom range
    e.g. input with range of (-5, 15) is rescaled to (-125 125) or (-1 1) for␣
↪use with VGG16.
    Designed for use with masked arrays.
    Inputs:
        r4_array | r4 masked array | works with masked arrays?
        min_max | dict | 'min' and 'max' of range desired as a dictionary.
        mean_centre | boolean | if True, each image's channels are mean␣
↪centered.
    Returns:
        r4_array | rank 4 numpy array | masked items are set to zero, rescaled␣
↪so that each channel for each image lies between min_max limits.
    History:
        2019/03/20 | now includes mean centering so doesn't stretch data to␣
↪custom range.
                     Instead only stretches until either min or max touches,␣
↪whilst mean is kept at 0
        2020/11/02 | MEG | Update so range can have a min and max, and not just␣
↪a range
        2021/01/06 | MEG | Upate to work with masked arrays.  Not test with␣
↪normal arrays.
    """
    if mean_centre:
        # get the average for each image (in all thre channels)
        im_channel_means = ma.mean(r4_array, axis = (1,2))
        # expand to r4 so we can do elementwise manipulation
        im_channel_means = expand_to_r4(im_channel_means, r4_array[0,:,:,0].
↪shape)
        # do mean centering
        r4_array -= im_channel_means

    # get the minimum of each image and each of its channels
```

```
    im_channel_min = ma.min(r4_array, axis = (1,2))
    # exapnd to rank 4 for elementwise applications
    im_channel_min = expand_to_r4(im_channel_min, r4_array[0,:,:,0].shape)
    # set so lowest channel for each image is 0
    r4_array -= im_channel_min
    # get the maximum of each image and each of its channels
    im_channel_max = ma.max(r4_array, axis = (1,2))
    # make suitable for elementwise applications
    im_channel_max = expand_to_r4(im_channel_max, r4_array[0,:,:,0].shape)
    r4_array /= im_channel_max  # should now be in range [0, 1]

    r4_array *= (min_max['max'] - min_max['min'])    # should now be in range␣
↪[0, new max-min]
    r4_array += min_max['min']                # and now in range [new min, new␣
↪max]
    # convert to numpy array, maksed incoherent areas are set to zero.
    r4_nparray = r4_array.filled(fill_value = 0)
    return r4_nparray
```

Plot the new reformatted data

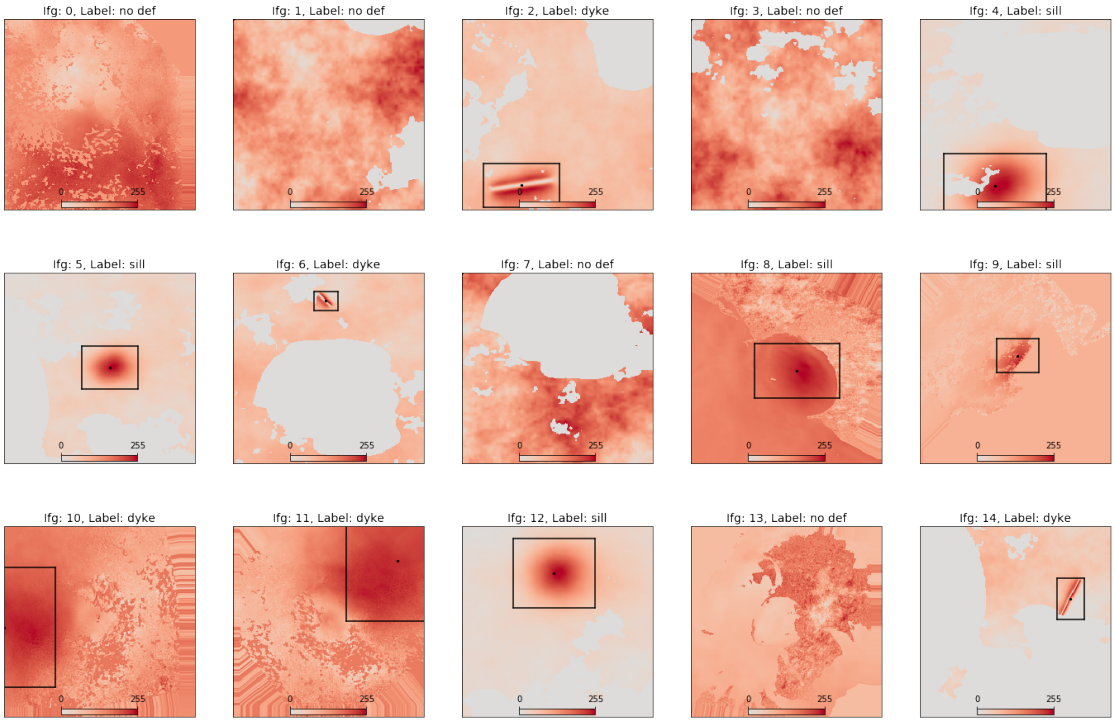
The following cell calls the `merge_and_rescale` function and plots the output. You will notice these images still look similar to our original images just on a new scale.

This might take a couple minutes….

[11]:
```
# get the paths to each file of real data
merge_and_rescale_data(synthetic_data_files, real_augmented_files,␣
↪cnn_settings['input_range'])                                    # merge the␣
↪real and synthetic data, and rescale it into the correct range for use with␣
↪the CNN
open_datafile_and_plot("./data/merged_out/data_file_0.npz", n_data = 15,
                       window_title = ' 04 Sample of merged and rescaled data')
```

```
Opening and merging file 0 of each type… Done.
Opening and merging file 1 of each type… Done.
Opening and merging file 2 of each type… Done.
Opening and merging file 3 of each type… Done.
Opening and merging file 4 of each type… Done.
Opening and merging file 5 of each type… Done.
Opening and merging file 6 of each type… Done.
Opening and merging file 7 of each type… Done.
Opening and merging file 8 of each type… Done.
Opening and merging file 9 of each type… Done.
```

```
Opening and merging file 10 of each type… Done.
Opening and merging file 11 of each type… Done.
Opening and merging file 12 of each type… Done.
```



Bottleneck Features

To train the model using different types of synthetic data, bottleneck learning is used. First, we compute the results from passing our entire dataset through the first five blocks of VGG16, before then training only the fully connected parts of our network (i.e.the classification output).

this uses the Keras VGG16 module which if the machine you are using has GPU's will automatically use. The below code

```
vgg16_block_1to5 = VGG16(weights='imagenet', include_top=False, input_shape = (224,224,3))
```

loads the first 5 convolutional blocks of VVG16 model trained for imagenet and tells it our input interferograms will be in the shape 224 X 224 x 3

```
X_btln = vgg16_block_1to5.predict(X, verbose = 1)
```

will pass the data through the blocks to create a tensor of shape (7 x 7 x 512)

If the machine you are using does not have GPUS then

The below segment of code may take a while if running on your laptop, if this is going to take too long then set

```
UsePreMadeBottlenecks = True
```

[12]:
```
UsePreMadeBottlenecks = False
```

[13]:
```
# Compute bottleneck features):

# load the first 5 (convolutional) blocks of VGG16 and their weights.
vgg16_block_1to5 = VGG16(weights='imagenet', include_top=False, input_shape =␣
↪(224,224,3))


data_out_files = sorted(glob.glob(f'./data/merged_out/*.npz'))

if UsePreMadeBottlenecks is False:
    try:
            shutil.rmtree(str(Path(f"./data/bottleneck_out/")))
    except:
        pass
    os.mkdir((Path(f"./data/bottleneck_out")))#
    bottleneck_folder = 'bottleneck_out'
    # get a list of the files output by step 05 (augmented real data and␣
↪synthetic data mixed and
    # rescaed to correct range, with 0s for masked areas.  )
    for file_n, data_out_file in enumerate(data_out_files):
        # loop through each of the step 05 files.
        print(f'Bottlneck file {file_n}:')
        data_out_file = Path(data_out_file)                                 ␣
↪                                                   # convert to path
        bottleneck_file_name = data_out_file.parts[-1].split('.')[0]        ␣
↪                                                   # and get last part which is␣
↪filename
        data = np.load(data_out_file)                                       ␣
↪                                                   # load the numpy file
        X = data['X']                                                       ␣
↪                                                   # extract the data for it
        Y_class = data['Y_class']                                           ␣
↪                                                   # and class labels.
        Y_loc = data['Y_loc']                                               ␣
↪                                                   # and location labels.
        X_btln = vgg16_block_1to5.predict(X, verbose = 1)
        # predict up to bottleneck
        np.savez(f'data/{bottleneck_folder}/{bottleneck_file_name}_bottleneck.
↪npz',
                 X = X_btln, Y_class = Y_class, Y_loc = Y_loc)
        # save the bottleneck file, and the two types of label.
else:
    print('using premade bottleneck files')
    bottleneck_folder = 'bottleneck_provided'
```

```
Bottlneck file 0:
2/2 [==============================] - 1s 476ms/step
Bottlneck file 1:
2/2 [==============================] - 1s 475ms/step
Bottlneck file 2:
2/2 [==============================] - 1s 438ms/step
Bottlneck file 3:
2/2 [==============================] - 1s 428ms/step
Bottlneck file 4:
2/2 [==============================] - 1s 476ms/step
Bottlneck file 5:
2/2 [==============================] - 1s 431ms/step
Bottlneck file 6:
2/2 [==============================] - 1s 476ms/step
Bottlneck file 7:
2/2 [==============================] - 1s 434ms/step
Bottlneck file 8:
2/2 [==============================] - 1s 436ms/step
Bottlneck file 9:
2/2 [==============================] - 1s 418ms/step
Bottlneck file 10:
2/2 [==============================] - 1s 443ms/step
Bottlneck file 11:
2/2 [==============================] - 1s 443ms/step
Bottlneck file 12:
2/2 [==============================] - 1s 453ms/step
Bottlneck file 13:
2/2 [==============================] - 1s 434ms/step
Bottlneck file 14:
2/2 [==============================] - 1s 432ms/step
Bottlneck file 15:
2/2 [==============================] - 1s 473ms/step
Bottlneck file 16:
2/2 [==============================] - 1s 416ms/step
Bottlneck file 17:
2/2 [==============================] - 1s 420ms/step
Bottlneck file 18:
2/2 [==============================] - 1s 438ms/step
Bottlneck file 19:
2/2 [==============================] - 1s 432ms/step
Bottlneck file 20:
2/2 [==============================] - 1s 419ms/step
Bottlneck file 21:
2/2 [==============================] - 1s 416ms/step
Bottlneck file 22:
2/2 [==============================] - 1s 433ms/step
Bottlneck file 23:
2/2 [==============================] - 1s 449ms/step
```

```
Bottlneck file 24:
2/2 [==============================] - 1s 395ms/step
Bottlneck file 25:
2/2 [==============================] - 1s 427ms/step
```

Training the Neural Network

First, we need two functions to divide the list into a training and testing dataset. Two functions are written below to divide the data files and bottleneck files into testing and training datasets and to load various features into arrays in your computers RAM

```
train_test_validate = file_list_divider(data_files, cnn_settings['n_files_train'],
                                        cnn_settings['n_files_validate'],
                                        cnn_settings['n_files_test'])
# assign the outputs
[data_files_train, data_files_validate, data_files_test] = train_test_validate
```

creates a set of files to use for training, validation and testing based of perscibed CNN settings

[14]:
```
def file_list_divider(file_list, n_files_train, n_files_validate, n_files_test):
    """ Given a list of files, divide it up into training, validating, and␣
 ↪testing lists.
    Inputs
        file_list | list | list of files
        n_files_train | int | Number of files to be used for training
        n_files_validate | int | Number of files to be used for validation␣
 ↪(during training)
        n_files_test | int | Number of files to be used for testing
    Returns:
        file_list_train | list | list of training files
        file_list_validate | list | list of validation files
        file_list_test | list | list of testing files
    History:
        2019/??/?? | MEG | Written
        2020/11/02 | MEG | Write docs
        """
    file_list_train = file_list[:n_files_train]
    file_list_validate = file_list[n_files_train:
 ↪(n_files_train+n_files_validate)]
    file_list_test = file_list[(n_files_train+n_files_validate) :␣
 ↪(n_files_train+n_files_validate+n_files_test)]
    return file_list_train, file_list_validate, file_list_test

def file_merger(files):
    """Given a list of files, open them and merge into one array.
    Inputs:
        files | list | list of paths to the .npz files
```

```
    Returns
        X | r4 array | data
        Y_class | r2 array | class labels, ? x n_classes
        Y_loc | r2 array | locations of signals, ? x 4 (as x,y, width, heigh)
    History:
        2020/10/?? | MEG | Written
        2020/11/11 | MEG | Update to remove various input arguments

    """

    def open_synthetic_data_npz(name_with_path):
        """Open a file data file """
        data = np.load(name_with_path)
        X = data['X']
        Y_class = data['Y_class']
        Y_loc = data['Y_loc']
        return X, Y_class, Y_loc

    n_files = len(files)
    for i, file in enumerate(files):
        X_batch, Y_class_batch, Y_loc_batch = open_synthetic_data_npz(file)
        if i == 0:
            n_data_per_file = X_batch.shape[0]
            # initate array, rank4 for image, get the size from the first file
            X = np.zeros((n_data_per_file * n_files, X_batch.shape[1], X_batch.
↪shape[2], X_batch.shape[3]))
            # should be flexible with class labels or one hot encoding
            Y_class = np.zeros((n_data_per_file  * n_files, Y_class_batch.
↪shape[1]))
            Y_loc = np.zeros((n_data_per_file * n_files, 4))    # four columns␣
↪for bounding box


        X[i*n_data_per_file:(i*n_data_per_file)+n_data_per_file,:,:,:] = X_batch
        Y_class[i*n_data_per_file:(i*n_data_per_file)+n_data_per_file,:] =␣
↪Y_class_batch
        Y_loc[i*n_data_per_file:(i*n_data_per_file)+n_data_per_file,:] =␣
↪Y_loc_batch

    return X, Y_class, Y_loc
```

[15]:
```
# Tain the fully connected part of the network)
cnn_settings = {'input_range'       : {'min':0, 'max':255}}
# the number of files that will be used to train the network
cnn_settings['n_files_train']     = 22
# the number of files that wil be used to validate the network (i.e. passed␣
↪through once per epoch)
```

```
cnn_settings['n_files_validate']  = 2
# the number of files held back for testing.
cnn_settings['n_files_test']      = 2
```

[16]:
```
data_files = sorted(glob.glob(f'./data/merged_out/*npz'), key = os.path.
 ↪getmtime)    # make list of data files
# and make a list of bottleneck files (ie files that have been passed through␣
 ↪the first 5 blocks of vgg16)
bottleneck_files = sorted(glob.glob('./data/'+str(bottleneck_folder)+'/
 ↪'+f'*npz'), key = os.path.getmtime)
if len(data_files) < (cnn_settings['n_files_train'] +␣
 ↪cnn_settings['n_files_validate'] + cnn_settings['n_files_test']):
    raise Exception(f"There are {len(data_files)} data files, but␣
 ↪{cnn_settings['n_files_train']} have been selected for training, "
                    f"{cnn_settings['n_files_validate']} for validation, and␣
 ↪{cnn_settings['n_files_test']} for testing, "
                    f"which sums to greater than the number of data files. ␣
 ↪Perhaps adjust the number of files used for the training stages? "
                    f"For now, exiting.")

data_files_train, data_files_validate, data_files_test =␣
 ↪file_list_divider(data_files,
                                                                        ␣
 ↪cnn_settings['n_files_train'],
                                                                        ␣
 ↪cnn_settings['n_files_validate'],
                                                                        ␣
 ↪cnn_settings['n_files_test'])                                  # divide the␣
 ↪files into train, validate and test
# also divide the bottleneck files
bottleneck_files_train, bottleneck_files_validate, bottleneck_files_test =␣
 ↪file_list_divider(bottleneck_files,
                                                                         ␣
 ↪cnn_settings['n_files_train'],
                                                                         ␣
 ↪cnn_settings['n_files_validate'],
                                                                         ␣
 ↪cnn_settings['n_files_test'])


# Open all the validation data to RAM
X_validate, Y_class_validate, Y_loc_validate      =␣
 ↪file_merger(data_files_validate)
# Open the validation data bottleneck features to RAM
X_validate_btln, Y_class_validate, Y_loc_validate =␣
 ↪file_merger(bottleneck_files_validate)
```

```
# Open the test data to RAM
X_test, Y_class_test, Y_loc_test                    = file_merger(data_files_test)
# Open the test data bottleneck features to RAM
X_test_btln, Y_class_test_btln, Y_loc_test_btln   =␣
↪file_merger(bottleneck_files_test)


print(f"    There are {len(data_files)} data files.  {len(data_files_train)}␣
↪will be used for training,"
      f"{len(data_files_validate)} for validation, and {len(data_files_test)}␣
↪for testing.  ")
```

```
    There are 26 data files.  22 will be used for training,2 for validation, and
2 for testing.
```

Define two headed model and training

The interferograms of shape (224×224×3) are passed through the five convolutional blocks of VGG16 to create a tensor of shape (7×7×512). This is flattened to make a vector of size 25,088, before being passed through fully connected layers of size 256, 128, and an output layer of size three (i.e., dyke, sill/point, or no deformation). This is done in our `define_two_head_model` which takes the input from our VGG16 model to give our 3 class output.

```
vgg16_block_1to5 = VGG16(weights='imagenet', include_top=False, input_shape = (224,224,3))
```

we then need to make the input to the fully connected model the same shape as the output of the 5th block of vgg16 then build the full connected part of the model and get the two model outputs using the `define_two_head_model` function

```
fc_model_input = Input(shape = vgg16_block_1to5.output_shape[1:])
output_class, output_loc = define_two_head_model(fc_model_input,
                                                 len(synthetic_ifgs_settings['defo_sources']))
```

to build our headed model

```
vgg16_2head_fc = Model(inputs=fc_model_input, outputs=[output_class, output_loc])
```

[17]:
```
def define_two_head_model(model_input, n_class_outputs = 3):
    """ Define the two headed model that we have designed to performed␣
↪classification and localisation.
    Inputs:
        model_input | tensorflow.python.framework.ops.Tensor |
                      The shape of the tensor that will be input to our model.
                      Usually the output of VGG16 (?x7x7x512)  Nb ? = batch␣
↪size.
        n_class_output | int | For a one hot encoding style output, there must␣
↪be as many neurons as classes
    Returns:
        output_class |tensorflow.python.framework.ops.Tensor |
```

```
                    The shape of the tensor output by the classifiction head.␣
↪ Usually ?x3
        output_loc | tensorflow.python.framework.ops.Tensor |
                    The shape of the tensor output by the localisation head. ␣
↪Usually ?x4
    History:
        2020_11_11 | MEG | Written
    """
    vgg16_block_1to5_flat = Flatten(name = 'vgg16_block_1to5_flat')(model_input)
    # flatten the model input (ie deep representation turned into a column␣
↪vector)
    # 1: the clasification head
    x = Dropout(0.2, name='class_dropout1')(vgg16_block_1to5_flat)
    # add a fully connected layer
    x = Dense(256, activation='relu', name='class_dense1')(x)
    x = Dropout(0.2, name='class_dropout2')(x)
    # add a fully connected layer
    x = Dense(128, activation='relu', name='class_dense2')(x)
    # and an ouput layer with 7 outputs (ie one per label)
    output_class = Dense(n_class_outputs, activation='softmax',  name =␣
↪'class_dense3')(x)


    # 2: the localization head
    x = Dense(2048, activation='relu', name='loc_dense1')(vgg16_block_1to5_flat)
    # add a fully connected layer
    x = Dense(1024, activation='relu', name='loc_dense2')(x)
    # add a fully connected layer
    x = Dense(1024, activation='relu', name='loc_dense3')(x)
    x = Dropout(0.2, name='loc_dropout1')(x)
    # add a fully connected layer
    x = Dense(512, activation='relu', name='loc_dense4')(x)
    # add a fully connected layer
    x = Dense(128, activation='relu', name='loc_dense5')(x)
    output_loc = Dense(4, name='loc_dense6')(x)

    return output_class, output_loc
```

[18]:
```
# Define, compile, and train the model
# VGG16 is used for its convolutional layers and weights (but no fully␣
 ↪connected part as we define out own )
vgg16_block_1to5 = VGG16(weights='imagenet', include_top=False, input_shape =␣
 ↪(224,224,3))
# the input to the fully connected model must be the same shape as the output␣
 ↪of the 5th block of vgg16
fc_model_input = Input(shape = vgg16_block_1to5.output_shape[1:])
```

```
# build the full connected part of the model, and get the two model outputs
output_class, output_loc = define_two_head_model(fc_model_input,␣
↪len(synthetic_ifgs_settings['defo_sources']))
# define the model.  Input is the shape of vgg16 block 1 to 5 output, and there␣
↪are two outputs (hence list)
vgg16_2head_fc = Model(inputs=fc_model_input, outputs=[output_class,␣
↪output_loc])
```

Plot model

Note Graphviz might not work on all systems, the code below will provide an alternative solution if graphviz fails

[19]:
```
try:
    os.mkdir((Path(f"./data/train_fully_connected_model")))
except:
    pass
try:
    plot_model(vgg16_2head_fc, to_file=f'data/train_fully_connected_model/
↪vgg16_2head_fc.png',
           # also plot the model.  This funtcion is known to be fragile due to␣
↪Graphviz dependencies.
           show_shapes = True, show_layer_names = True)
except:
    vgg16_2head_fc.summary()
```

Define two headed model and training

Now we can compile the model passing in some optimizations: the standard Adam gradient-based optimizer and the loss functions (Cross-Entropy loss as Softmax is to be used) and request the accuracy metric to be reported

```
vgg16_2head_fc.compile(optimizer = opt_used, loss=[loss_class, loss_loc],
                       # compile the model
                       loss_weights = fc_loss_weights, metrics=['accuracy'])
```

And write a function to train out double-headed model

```
train_double_network(vgg16_2head_fc,  bottleneck_files_train,n_epochs_fc,
                     ['class_dense3_loss','loc_dense6_loss'], Xvalidate_btln,
                     Y_class_validate, Y_loc_validate,
                     len(synthetic_ifgs_settings['defo_sources']))
```

which takes our compiled model `vgg16_2head_fc` and our subset of bottleneck files and trains over a number of "epochs" here set to 10 for speed rather than accuracy.

Training the model may take a few minutes …

[20]:
```
def train_double_network(model, files, n_epochs, loss_names,
                                    X_validate, Y_class_validate,␣
↪Y_loc_validate, n_classes):
    """Train a double headed model using training data stored in separate files.
↪
    Inputs:
        model | keras model | the model to be trained
        files | list | list of paths and filenames for the files used during␣
↪training
        n_epochs | int | number of epochs to train for
        loss names | list | names of outputs of losses (e.g. "class_dense3_loss)
    Returns
        model | keras model | updated by the fit process
        metrics_loss | r2 array | columns are: total loss/class loss/loc loss /
↪validate total loss/validate
                                                class loss/ validate loc loss
        metrics_class | r2 array | columns are class accuracy, validation class␣
↪accuracy

    2019/03/25 | Written.
    """
    n_files_train = len(files)          # get the number of training files

    metrics_class = np.zeros((n_files_train*n_epochs, 2))   # train class␣
↪acuracy, valiate class accuracy
    # total loss/class loss/loc loss /validate total loss/validate class loss/␣
↪validate loc loss
    metrics_loss = np.zeros((n_files_train*n_epochs, 6))
    for e in range(n_epochs):     # loop through the number of epochs
        for file_num, file in enumerate(files):    # for each epoch, loop␣
↪through all files once

            data = np.load(file)
            X_batch = data['X']
            Y_batch_class = data['Y_class']
            Y_batch_loc = data['Y_loc']

            if n_classes !=  Y_batch_class.shape[1]:
                # convert to one hot encoding (from class labels)
                Y_batch_class = keras.utils.to_categorical(Y_batch_class,␣
↪n_classes, dtype='float32')

            history_train_temp = model.fit(X_batch, [Y_batch_class,␣
↪Y_batch_loc], batch_size=32,
                                           epochs=1, verbose = 0)
            # main loss
```

```
            metrics_loss[(e*n_files_train)+file_num, 0] = history_train_temp.
↪history['loss'][0]
            # class loss
            metrics_loss[(e*n_files_train)+file_num, 1] = history_train_temp.
↪history[loss_names[0]][0]
            # localization loss
            metrics_loss[(e*n_files_train)+file_num, 2] = history_train_temp.
↪history[loss_names[1]][0]
            metrics_class[(e*n_files_train)+file_num, 0] = history_train_temp.
↪history['class_dense3_accuracy'][0]           # classification accuracy     ␣
↪
            print(f'Epoch {e}, file {file_num}: Loss =␣
↪{round(metrics_loss[(e*n_files_train)+file_num, 0],0)}, '
                                        f'Class. loss =␣
↪{round(metrics_loss[(e*n_files_train)+file_num, 1],2)}, '
                                        f'Class. acc. =␣
↪{round(metrics_class[(e*n_files_train)+file_num, 0],2)}, '
                                         f'Loc. loss =␣
↪{round(metrics_loss[(e*n_files_train)+file_num, 2],0)}')



        history_validate_temp = model.evaluate(X_validate, [Y_class_validate,␣
↪Y_loc_validate],
                                               batch_size = 32, verbose = 0)
        metrics_loss[(e*n_files_train)+file_num, 3] = history_validate_temp[0] ␣
↪    # main loss
        metrics_loss[(e*n_files_train)+file_num, 4] = history_validate_temp[1] ␣
↪    # class loss
        metrics_loss[(e*n_files_train)+file_num, 5] = history_validate_temp[2] ␣
↪    # localisation loss
        metrics_class[(e*n_files_train)+file_num, 1] = history_validate_temp[3]␣
↪    # classification  accuracy
        print(f'Epoch {e}, valid.: Loss =␣
↪{round(metrics_loss[(e*n_files_train)+file_num, 3],0)}, '
                           f'Class. loss =␣
↪{round(metrics_loss[(e*n_files_train)+file_num, 4],2)}, '
                           f'Class. acc. =␣
↪{round(metrics_class[(e*n_files_train)+file_num, 1],2)}, '
                            f'Loc. loss =␣
↪{round(metrics_loss[(e*n_files_train)+file_num, 5],0)}')

    # class loss, validate class loss, class accuracy, validate class accuracy
    metrics_class = np.hstack((metrics_loss[:,1:2], metrics_loss[:,4:5],␣
↪metrics_class ))
```

```
    # localisation loss, validate localisation loss, localisation accuracy,␣
↪validate localisation accuracy
    metrics_localisation = np.hstack((metrics_loss[:,2:3], metrics_loss[:,5:]))
    # class accuracy, validate class accuracy
    metrics_combined_loss = np.hstack((metrics_loss[:,1:2], metrics_loss[:,3:
↪4]))
    return model, metrics_class, metrics_localisation, metrics_combined_loss
```

[21]:
```
# the relative weighting of the two losses (classificaiton and localisation)
# to contribute to the global loss.  Classification first, localisation second.
fc_loss_weights = [0.05, 0.95]
# the number of epochs to train the fully connected network for
# (ie. the number of times all the training data are passed through the model)
n_epochs_fc = 10
```

[22]:
```
# good loss to use for classification problems, may need to switch to binary if␣
↪only two classes though?
loss_class = losses.categorical_crossentropy
# loss for localisation
loss_loc = losses.mean_squared_error
# adam with Nesterov accelerated gradient
opt_used = optimizers.Nadam(clipnorm = 1., clipvalue = 0.5)
# accuracy is useful to have on the terminal during training
vgg16_2head_fc.compile(optimizer = opt_used, loss=[loss_class, loss_loc],
                       # compile the model
                       loss_weights = fc_loss_weights, metrics=['accuracy'])


[vgg16_2head_fc,  metrics_class_fc,
 metrics_localisation_fc, metrics_combined_loss_fc ] =␣
↪train_double_network(vgg16_2head_fc,
                                                                          ␣
↪bottleneck_files_train,
                                                                          ␣
↪n_epochs_fc,
                                                                          ␣
↪['class_dense3_loss',
                                                                           ␣
↪'loc_dense6_loss'],
                                                                          ␣
↪X_validate_btln,
                                                                          ␣
↪Y_class_validate,
                                                                          ␣
↪Y_loc_validate,
```

```
↪len(synthetic_ifgs_settings['defo_sources']))
```

```
Epoch 0, file 0: Loss = 7799.0, Class. loss = 18.92, Class. acc. = 0.46, Loc.
loss = 8209.0
Epoch 0, file 1: Loss = 8778.0, Class. loss = 8.23, Class. acc. = 0.42, Loc.
loss = 9239.0
Epoch 0, file 2: Loss = 6615.0, Class. loss = 6.83, Class. acc. = 0.32, Loc.
loss = 6962.0
Epoch 0, file 3: Loss = 4257.0, Class. loss = 4.09, Class. acc. = 0.52, Loc.
loss = 4481.0
Epoch 0, file 4: Loss = 4445.0, Class. loss = 4.43, Class. acc. = 0.44, Loc.
loss = 4678.0
Epoch 0, file 5: Loss = 4287.0, Class. loss = 4.69, Class. acc. = 0.5, Loc. loss
= 4512.0
Epoch 0, file 6: Loss = 3890.0, Class. loss = 4.34, Class. acc. = 0.5, Loc. loss
= 4094.0
Epoch 0, file 7: Loss = 3726.0, Class. loss = 1.63, Class. acc. = 0.64, Loc.
loss = 3922.0
Epoch 0, file 8: Loss = 3549.0, Class. loss = 4.89, Class. acc. = 0.44, Loc.
loss = 3736.0
Epoch 0, file 9: Loss = 3123.0, Class. loss = 1.51, Class. acc. = 0.62, Loc.
loss = 3288.0
Epoch 0, file 10: Loss = 4023.0, Class. loss = 1.76, Class. acc. = 0.62, Loc.
loss = 4234.0
Epoch 0, file 11: Loss = 3060.0, Class. loss = 2.07, Class. acc. = 0.6, Loc.
loss = 3221.0
Epoch 0, file 12: Loss = 2471.0, Class. loss = 0.92, Class. acc. = 0.64, Loc.
loss = 2601.0
Epoch 0, file 13: Loss = 3457.0, Class. loss = 3.77, Class. acc. = 0.6, Loc.
loss = 3638.0
Epoch 0, file 14: Loss = 1614.0, Class. loss = 1.58, Class. acc. = 0.74, Loc.
loss = 1699.0
Epoch 0, file 15: Loss = 2877.0, Class. loss = 3.59, Class. acc. = 0.52, Loc.
loss = 3029.0
Epoch 0, file 16: Loss = 2653.0, Class. loss = 3.05, Class. acc. = 0.58, Loc.
loss = 2793.0
Epoch 0, file 17: Loss = 2339.0, Class. loss = 1.47, Class. acc. = 0.66, Loc.
loss = 2462.0
Epoch 0, file 18: Loss = 2936.0, Class. loss = 3.34, Class. acc. = 0.58, Loc.
loss = 3091.0
Epoch 0, file 19: Loss = 2159.0, Class. loss = 1.71, Class. acc. = 0.76, Loc.
loss = 2272.0
Epoch 0, file 20: Loss = 2610.0, Class. loss = 1.79, Class. acc. = 0.64, Loc.
loss = 2747.0
Epoch 0, file 21: Loss = 1510.0, Class. loss = 2.66, Class. acc. = 0.66, Loc.
loss = 1589.0
```

```
Epoch 0, valid.: Loss = 1999.0, Class. loss = 2.46, Class. acc. = 0.64, Loc.
loss = 2104.0
Epoch 1, file 0: Loss = 2317.0, Class. loss = 2.76, Class. acc. = 0.68, Loc.
loss = 2439.0
Epoch 1, file 1: Loss = 2455.0, Class. loss = 3.49, Class. acc. = 0.62, Loc.
loss = 2584.0
Epoch 1, file 2: Loss = 2613.0, Class. loss = 0.64, Class. acc. = 0.82, Loc.
loss = 2751.0
Epoch 1, file 3: Loss = 1922.0, Class. loss = 1.35, Class. acc. = 0.76, Loc.
loss = 2023.0
Epoch 1, file 4: Loss = 2337.0, Class. loss = 1.19, Class. acc. = 0.8, Loc. loss
= 2460.0
Epoch 1, file 5: Loss = 2276.0, Class. loss = 1.57, Class. acc. = 0.78, Loc.
loss = 2396.0
Epoch 1, file 6: Loss = 2365.0, Class. loss = 2.16, Class. acc. = 0.7, Loc. loss
= 2489.0
Epoch 1, file 7: Loss = 1294.0, Class. loss = 2.49, Class. acc. = 0.72, Loc.
loss = 1362.0
Epoch 1, file 8: Loss = 1989.0, Class. loss = 1.75, Class. acc. = 0.72, Loc.
loss = 2094.0
Epoch 1, file 9: Loss = 1522.0, Class. loss = 1.91, Class. acc. = 0.68, Loc.
loss = 1602.0
Epoch 1, file 10: Loss = 1585.0, Class. loss = 0.76, Class. acc. = 0.88, Loc.
loss = 1668.0
Epoch 1, file 11: Loss = 1933.0, Class. loss = 1.57, Class. acc. = 0.66, Loc.
loss = 2034.0
Epoch 1, file 12: Loss = 1484.0, Class. loss = 0.39, Class. acc. = 0.88, Loc.
loss = 1562.0
Epoch 1, file 13: Loss = 2501.0, Class. loss = 2.13, Class. acc. = 0.82, Loc.
loss = 2633.0
Epoch 1, file 14: Loss = 983.0, Class. loss = 0.59, Class. acc. = 0.88, Loc.
loss = 1034.0
Epoch 1, file 15: Loss = 1519.0, Class. loss = 1.65, Class. acc. = 0.74, Loc.
loss = 1599.0
Epoch 1, file 16: Loss = 1578.0, Class. loss = 1.56, Class. acc. = 0.76, Loc.
loss = 1661.0
Epoch 1, file 17: Loss = 1026.0, Class. loss = 1.24, Class. acc. = 0.84, Loc.
loss = 1080.0
Epoch 1, file 18: Loss = 1439.0, Class. loss = 2.99, Class. acc. = 0.64, Loc.
loss = 1515.0
Epoch 1, file 19: Loss = 1688.0, Class. loss = 0.91, Class. acc. = 0.76, Loc.
loss = 1776.0
Epoch 1, file 20: Loss = 1311.0, Class. loss = 0.75, Class. acc. = 0.84, Loc.
loss = 1380.0
Epoch 1, file 21: Loss = 1307.0, Class. loss = 1.29, Class. acc. = 0.74, Loc.
loss = 1376.0
Epoch 1, valid.: Loss = 1406.0, Class. loss = 0.81, Class. acc. = 0.74, Loc.
loss = 1480.0
```

```
Epoch 2, file 0: Loss = 1225.0, Class. loss = 0.41, Class. acc. = 0.9, Loc. loss
= 1289.0
Epoch 2, file 1: Loss = 1316.0, Class. loss = 0.86, Class. acc. = 0.82, Loc.
loss = 1385.0
Epoch 2, file 2: Loss = 1594.0, Class. loss = 0.34, Class. acc. = 0.92, Loc.
loss = 1678.0
Epoch 2, file 3: Loss = 1476.0, Class. loss = 1.4, Class. acc. = 0.86, Loc. loss
= 1553.0
Epoch 2, file 4: Loss = 1623.0, Class. loss = 1.05, Class. acc. = 0.84, Loc.
loss = 1709.0
Epoch 2, file 5: Loss = 1264.0, Class. loss = 0.57, Class. acc. = 0.82, Loc.
loss = 1331.0
Epoch 2, file 6: Loss = 1233.0, Class. loss = 1.03, Class. acc. = 0.84, Loc.
loss = 1298.0
Epoch 2, file 7: Loss = 1392.0, Class. loss = 0.55, Class. acc. = 0.92, Loc.
loss = 1465.0
Epoch 2, file 8: Loss = 1231.0, Class. loss = 0.88, Class. acc. = 0.74, Loc.
loss = 1296.0
Epoch 2, file 9: Loss = 1055.0, Class. loss = 0.91, Class. acc. = 0.8, Loc. loss
= 1110.0
Epoch 2, file 10: Loss = 1650.0, Class. loss = 1.02, Class. acc. = 0.82, Loc.
loss = 1737.0
Epoch 2, file 11: Loss = 813.0, Class. loss = 0.51, Class. acc. = 0.88, Loc.
loss = 856.0
Epoch 2, file 12: Loss = 1091.0, Class. loss = 0.93, Class. acc. = 0.9, Loc.
loss = 1148.0
Epoch 2, file 13: Loss = 1242.0, Class. loss = 1.23, Class. acc. = 0.78, Loc.
loss = 1307.0
Epoch 2, file 14: Loss = 1101.0, Class. loss = 0.2, Class. acc. = 0.92, Loc.
loss = 1159.0
Epoch 2, file 15: Loss = 1216.0, Class. loss = 1.6, Class. acc. = 0.82, Loc.
loss = 1280.0
Epoch 2, file 16: Loss = 1403.0, Class. loss = 1.33, Class. acc. = 0.84, Loc.
loss = 1477.0
Epoch 2, file 17: Loss = 974.0, Class. loss = 0.65, Class. acc. = 0.9, Loc. loss
= 1025.0
Epoch 2, file 18: Loss = 1280.0, Class. loss = 1.07, Class. acc. = 0.82, Loc.
loss = 1347.0
Epoch 2, file 19: Loss = 888.0, Class. loss = 1.47, Class. acc. = 0.82, Loc.
loss = 935.0
Epoch 2, file 20: Loss = 1404.0, Class. loss = 0.44, Class. acc. = 0.92, Loc.
loss = 1478.0
Epoch 2, file 21: Loss = 705.0, Class. loss = 0.94, Class. acc. = 0.8, Loc. loss
= 742.0
Epoch 2, valid.: Loss = 1270.0, Class. loss = 1.44, Class. acc. = 0.8, Loc. loss
= 1337.0
Epoch 3, file 0: Loss = 766.0, Class. loss = 1.17, Class. acc. = 0.84, Loc. loss
= 806.0
```

```
Epoch 3, file 1: Loss = 1626.0, Class. loss = 1.15, Class. acc. = 0.84, Loc.
loss = 1712.0
Epoch 3, file 2: Loss = 1054.0, Class. loss = 0.13, Class. acc. = 0.94, Loc.
loss = 1110.0
Epoch 3, file 3: Loss = 856.0, Class. loss = 0.48, Class. acc. = 0.88, Loc. loss
= 901.0
Epoch 3, file 4: Loss = 1119.0, Class. loss = 0.85, Class. acc. = 0.9, Loc. loss
= 1178.0
Epoch 3, file 5: Loss = 1033.0, Class. loss = 0.81, Class. acc. = 0.9, Loc. loss
= 1087.0
Epoch 3, file 6: Loss = 780.0, Class. loss = 0.8, Class. acc. = 0.9, Loc. loss =
821.0
Epoch 3, file 7: Loss = 725.0, Class. loss = 0.18, Class. acc. = 0.96, Loc. loss
= 763.0
Epoch 3, file 8: Loss = 914.0, Class. loss = 1.07, Class. acc. = 0.8, Loc. loss
= 962.0
Epoch 3, file 9: Loss = 969.0, Class. loss = 0.3, Class. acc. = 0.9, Loc. loss =
1020.0
Epoch 3, file 10: Loss = 636.0, Class. loss = 0.5, Class. acc. = 0.92, Loc. loss
= 670.0
Epoch 3, file 11: Loss = 719.0, Class. loss = 0.21, Class. acc. = 0.92, Loc.
loss = 756.0
Epoch 3, file 12: Loss = 798.0, Class. loss = 0.13, Class. acc. = 0.94, Loc.
loss = 840.0
Epoch 3, file 13: Loss = 1058.0, Class. loss = 0.38, Class. acc. = 0.96, Loc.
loss = 1114.0
Epoch 3, file 14: Loss = 556.0, Class. loss = 0.03, Class. acc. = 1.0, Loc. loss
= 585.0
Epoch 3, file 15: Loss = 1059.0, Class. loss = 0.42, Class. acc. = 0.9, Loc.
loss = 1115.0
Epoch 3, file 16: Loss = 736.0, Class. loss = 1.38, Class. acc. = 0.8, Loc. loss
= 775.0
Epoch 3, file 17: Loss = 714.0, Class. loss = 0.36, Class. acc. = 0.96, Loc.
loss = 751.0
Epoch 3, file 18: Loss = 981.0, Class. loss = 0.88, Class. acc. = 0.88, Loc.
loss = 1032.0
Epoch 3, file 19: Loss = 485.0, Class. loss = 0.28, Class. acc. = 0.96, Loc.
loss = 510.0
Epoch 3, file 20: Loss = 592.0, Class. loss = 0.63, Class. acc. = 0.9, Loc. loss
= 623.0
Epoch 3, file 21: Loss = 863.0, Class. loss = 0.39, Class. acc. = 0.92, Loc.
loss = 908.0
Epoch 3, valid.: Loss = 1329.0, Class. loss = 0.87, Class. acc. = 0.84, Loc.
loss = 1399.0
Epoch 4, file 0: Loss = 1251.0, Class. loss = 1.03, Class. acc. = 0.88, Loc.
loss = 1317.0
Epoch 4, file 1: Loss = 806.0, Class. loss = 0.43, Class. acc. = 0.96, Loc. loss
= 848.0
```

```
Epoch 4, file 2: Loss = 634.0, Class. loss = 0.22, Class. acc. = 0.96, Loc. loss
= 668.0
Epoch 4, file 3: Loss = 760.0, Class. loss = 0.07, Class. acc. = 0.98, Loc. loss
= 800.0
Epoch 4, file 4: Loss = 777.0, Class. loss = 0.38, Class. acc. = 0.94, Loc. loss
= 818.0
Epoch 4, file 5: Loss = 859.0, Class. loss = 0.49, Class. acc. = 0.94, Loc. loss
= 904.0
Epoch 4, file 6: Loss = 650.0, Class. loss = 0.45, Class. acc. = 0.9, Loc. loss
= 684.0
Epoch 4, file 7: Loss = 483.0, Class. loss = 0.16, Class. acc. = 0.98, Loc. loss
= 508.0
Epoch 4, file 8: Loss = 1154.0, Class. loss = 0.32, Class. acc. = 0.92, Loc.
loss = 1214.0
Epoch 4, file 9: Loss = 504.0, Class. loss = 0.74, Class. acc. = 0.86, Loc. loss
= 531.0
Epoch 4, file 10: Loss = 671.0, Class. loss = 0.09, Class. acc. = 0.94, Loc.
loss = 706.0
Epoch 4, file 11: Loss = 608.0, Class. loss = 0.12, Class. acc. = 0.96, Loc.
loss = 640.0
Epoch 4, file 12: Loss = 582.0, Class. loss = 0.61, Class. acc. = 0.88, Loc.
loss = 613.0
Epoch 4, file 13: Loss = 771.0, Class. loss = 0.21, Class. acc. = 0.96, Loc.
loss = 811.0
Epoch 4, file 14: Loss = 585.0, Class. loss = 0.24, Class. acc. = 0.92, Loc.
loss = 616.0
Epoch 4, file 15: Loss = 592.0, Class. loss = 0.39, Class. acc. = 0.94, Loc.
loss = 624.0
Epoch 4, file 16: Loss = 587.0, Class. loss = 1.59, Class. acc. = 0.84, Loc.
loss = 618.0
Epoch 4, file 17: Loss = 581.0, Class. loss = 0.53, Class. acc. = 0.94, Loc.
loss = 612.0
Epoch 4, file 18: Loss = 476.0, Class. loss = 0.4, Class. acc. = 0.92, Loc. loss
= 501.0
Epoch 4, file 19: Loss = 638.0, Class. loss = 0.97, Class. acc. = 0.86, Loc.
loss = 672.0
Epoch 4, file 20: Loss = 588.0, Class. loss = 0.65, Class. acc. = 0.9, Loc. loss
= 618.0
Epoch 4, file 21: Loss = 933.0, Class. loss = 0.5, Class. acc. = 0.9, Loc. loss
= 982.0
Epoch 4, valid.: Loss = 1288.0, Class. loss = 1.14, Class. acc. = 0.9, Loc. loss
= 1356.0
Epoch 5, file 0: Loss = 957.0, Class. loss = 0.61, Class. acc. = 0.92, Loc. loss
= 1007.0
Epoch 5, file 1: Loss = 718.0, Class. loss = 1.2, Class. acc. = 0.9, Loc. loss =
755.0
Epoch 5, file 2: Loss = 611.0, Class. loss = 0.48, Class. acc. = 0.92, Loc. loss
= 644.0
```

```
Epoch 5, file 3: Loss = 542.0, Class. loss = 0.37, Class. acc. = 0.9, Loc. loss
= 571.0
Epoch 5, file 4: Loss = 710.0, Class. loss = 0.29, Class. acc. = 0.94, Loc. loss
= 747.0
Epoch 5, file 5: Loss = 773.0, Class. loss = 0.54, Class. acc. = 0.9, Loc. loss
= 814.0
Epoch 5, file 6: Loss = 525.0, Class. loss = 0.73, Class. acc. = 0.9, Loc. loss
= 552.0
Epoch 5, file 7: Loss = 690.0, Class. loss = 0.17, Class. acc. = 0.98, Loc. loss
= 727.0
Epoch 5, file 8: Loss = 526.0, Class. loss = 1.18, Class. acc. = 0.78, Loc. loss
= 554.0
Epoch 5, file 9: Loss = 612.0, Class. loss = 0.56, Class. acc. = 0.94, Loc. loss
= 644.0
Epoch 5, file 10: Loss = 642.0, Class. loss = 0.56, Class. acc. = 0.94, Loc.
loss = 676.0
Epoch 5, file 11: Loss = 467.0, Class. loss = 0.46, Class. acc. = 0.96, Loc.
loss = 492.0
Epoch 5, file 12: Loss = 570.0, Class. loss = 0.17, Class. acc. = 0.96, Loc.
loss = 600.0
Epoch 5, file 13: Loss = 499.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 526.0
Epoch 5, file 14: Loss = 648.0, Class. loss = 0.19, Class. acc. = 0.96, Loc.
loss = 682.0
Epoch 5, file 15: Loss = 457.0, Class. loss = 0.13, Class. acc. = 0.94, Loc.
loss = 481.0
Epoch 5, file 16: Loss = 717.0, Class. loss = 0.25, Class. acc. = 0.96, Loc.
loss = 754.0
Epoch 5, file 17: Loss = 542.0, Class. loss = 0.43, Class. acc. = 0.92, Loc.
loss = 571.0
Epoch 5, file 18: Loss = 675.0, Class. loss = 0.34, Class. acc. = 0.96, Loc.
loss = 710.0
Epoch 5, file 19: Loss = 508.0, Class. loss = 0.42, Class. acc. = 0.9, Loc. loss
= 535.0
Epoch 5, file 20: Loss = 477.0, Class. loss = 0.62, Class. acc. = 0.92, Loc.
loss = 502.0
Epoch 5, file 21: Loss = 680.0, Class. loss = 0.36, Class. acc. = 0.88, Loc.
loss = 716.0
Epoch 5, valid.: Loss = 937.0, Class. loss = 1.3, Class. acc. = 0.86, Loc. loss
= 986.0
Epoch 6, file 0: Loss = 485.0, Class. loss = 0.29, Class. acc. = 0.98, Loc. loss
= 511.0
Epoch 6, file 1: Loss = 635.0, Class. loss = 0.02, Class. acc. = 0.98, Loc. loss
= 668.0
Epoch 6, file 2: Loss = 548.0, Class. loss = 1.19, Class. acc. = 0.92, Loc. loss
= 576.0
Epoch 6, file 3: Loss = 394.0, Class. loss = 0.64, Class. acc. = 0.92, Loc. loss
= 415.0
```

```
Epoch 6, file 4: Loss = 655.0, Class. loss = 0.67, Class. acc. = 0.98, Loc. loss
= 689.0
Epoch 6, file 5: Loss = 511.0, Class. loss = 0.29, Class. acc. = 0.88, Loc. loss
= 537.0
Epoch 6, file 6: Loss = 623.0, Class. loss = 0.42, Class. acc. = 0.9, Loc. loss
= 656.0
Epoch 6, file 7: Loss = 339.0, Class. loss = 0.09, Class. acc. = 0.96, Loc. loss
= 356.0
Epoch 6, file 8: Loss = 382.0, Class. loss = 0.68, Class. acc. = 0.94, Loc. loss
= 402.0
Epoch 6, file 9: Loss = 638.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
672.0
Epoch 6, file 10: Loss = 351.0, Class. loss = 0.31, Class. acc. = 0.94, Loc.
loss = 370.0
Epoch 6, file 11: Loss = 484.0, Class. loss = 0.5, Class. acc. = 0.92, Loc. loss
= 509.0
Epoch 6, file 12: Loss = 523.0, Class. loss = 0.24, Class. acc. = 0.96, Loc.
loss = 551.0
Epoch 6, file 13: Loss = 582.0, Class. loss = 0.58, Class. acc. = 0.92, Loc.
loss = 612.0
Epoch 6, file 14: Loss = 301.0, Class. loss = 0.25, Class. acc. = 0.96, Loc.
loss = 317.0
Epoch 6, file 15: Loss = 540.0, Class. loss = 0.62, Class. acc. = 0.9, Loc. loss
= 569.0
Epoch 6, file 16: Loss = 492.0, Class. loss = 0.14, Class. acc. = 0.96, Loc.
loss = 518.0
Epoch 6, file 17: Loss = 418.0, Class. loss = 0.34, Class. acc. = 0.92, Loc.
loss = 439.0
Epoch 6, file 18: Loss = 866.0, Class. loss = 0.17, Class. acc. = 0.96, Loc.
loss = 912.0
Epoch 6, file 19: Loss = 537.0, Class. loss = 0.53, Class. acc. = 0.96, Loc.
loss = 565.0
Epoch 6, file 20: Loss = 464.0, Class. loss = 0.12, Class. acc. = 0.98, Loc.
loss = 488.0
Epoch 6, file 21: Loss = 330.0, Class. loss = 0.02, Class. acc. = 0.98, Loc.
loss = 348.0
Epoch 6, valid.: Loss = 1187.0, Class. loss = 1.3, Class. acc. = 0.83, Loc. loss
= 1249.0
Epoch 7, file 0: Loss = 724.0, Class. loss = 0.68, Class. acc. = 0.88, Loc. loss
= 762.0
Epoch 7, file 1: Loss = 762.0, Class. loss = 0.81, Class. acc. = 0.94, Loc. loss
= 802.0
Epoch 7, file 2: Loss = 496.0, Class. loss = 0.21, Class. acc. = 0.94, Loc. loss
= 522.0
Epoch 7, file 3: Loss = 453.0, Class. loss = 0.95, Class. acc. = 0.98, Loc. loss
= 477.0
Epoch 7, file 4: Loss = 516.0, Class. loss = 0.19, Class. acc. = 0.98, Loc. loss
= 543.0
```

```
Epoch 7, file 5: Loss = 526.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
554.0
Epoch 7, file 6: Loss = 641.0, Class. loss = 0.35, Class. acc. = 0.96, Loc. loss
= 675.0
Epoch 7, file 7: Loss = 267.0, Class. loss = 0.07, Class. acc. = 0.98, Loc. loss
= 281.0
Epoch 7, file 8: Loss = 484.0, Class. loss = 0.37, Class. acc. = 0.9, Loc. loss
= 509.0
Epoch 7, file 9: Loss = 386.0, Class. loss = 0.87, Class. acc. = 0.92, Loc. loss
= 407.0
Epoch 7, file 10: Loss = 304.0, Class. loss = 0.74, Class. acc. = 0.96, Loc.
loss = 319.0
Epoch 7, file 11: Loss = 377.0, Class. loss = 0.26, Class. acc. = 0.94, Loc.
loss = 397.0
Epoch 7, file 12: Loss = 457.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 481.0
Epoch 7, file 13: Loss = 401.0, Class. loss = 1.6, Class. acc. = 0.88, Loc. loss
= 423.0
Epoch 7, file 14: Loss = 304.0, Class. loss = 0.4, Class. acc. = 0.94, Loc. loss
= 320.0
Epoch 7, file 15: Loss = 329.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 346.0
Epoch 7, file 16: Loss = 484.0, Class. loss = 0.35, Class. acc. = 0.94, Loc.
loss = 509.0
Epoch 7, file 17: Loss = 361.0, Class. loss = 0.24, Class. acc. = 0.98, Loc.
loss = 380.0
Epoch 7, file 18: Loss = 384.0, Class. loss = 0.89, Class. acc. = 0.94, Loc.
loss = 405.0
Epoch 7, file 19: Loss = 238.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 251.0
Epoch 7, file 20: Loss = 333.0, Class. loss = 0.13, Class. acc. = 0.98, Loc.
loss = 351.0
Epoch 7, file 21: Loss = 441.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 464.0
Epoch 7, valid.: Loss = 719.0, Class. loss = 1.2, Class. acc. = 0.82, Loc. loss
= 757.0
Epoch 8, file 0: Loss = 338.0, Class. loss = 0.21, Class. acc. = 0.98, Loc. loss
= 356.0
Epoch 8, file 1: Loss = 450.0, Class. loss = 0.1, Class. acc. = 0.98, Loc. loss
= 474.0
Epoch 8, file 2: Loss = 314.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
330.0
Epoch 8, file 3: Loss = 431.0, Class. loss = 0.35, Class. acc. = 0.96, Loc. loss
= 454.0
Epoch 8, file 4: Loss = 332.0, Class. loss = 0.8, Class. acc. = 0.96, Loc. loss
= 349.0
Epoch 8, file 5: Loss = 304.0, Class. loss = 0.08, Class. acc. = 0.98, Loc. loss
= 320.0
```

```
Epoch 8, file 6: Loss = 376.0, Class. loss = 0.15, Class. acc. = 0.96, Loc. loss
= 396.0
Epoch 8, file 7: Loss = 301.0, Class. loss = 0.1, Class. acc. = 0.98, Loc. loss
= 316.0
Epoch 8, file 8: Loss = 402.0, Class. loss = 0.72, Class. acc. = 0.92, Loc. loss
= 424.0
Epoch 8, file 9: Loss = 385.0, Class. loss = 0.04, Class. acc. = 0.98, Loc. loss
= 405.0
Epoch 8, file 10: Loss = 362.0, Class. loss = 0.48, Class. acc. = 0.96, Loc.
loss = 381.0
Epoch 8, file 11: Loss = 410.0, Class. loss = 0.04, Class. acc. = 0.98, Loc.
loss = 431.0
Epoch 8, file 12: Loss = 271.0, Class. loss = 0.05, Class. acc. = 0.98, Loc.
loss = 285.0
Epoch 8, file 13: Loss = 511.0, Class. loss = 0.05, Class. acc. = 0.98, Loc.
loss = 537.0
Epoch 8, file 14: Loss = 363.0, Class. loss = 0.34, Class. acc. = 0.96, Loc.
loss = 382.0
Epoch 8, file 15: Loss = 354.0, Class. loss = 0.08, Class. acc. = 0.98, Loc.
loss = 372.0
Epoch 8, file 16: Loss = 482.0, Class. loss = 0.11, Class. acc. = 0.94, Loc.
loss = 507.0
Epoch 8, file 17: Loss = 443.0, Class. loss = 0.36, Class. acc. = 0.94, Loc.
loss = 466.0
Epoch 8, file 18: Loss = 342.0, Class. loss = 0.99, Class. acc. = 0.96, Loc.
loss = 360.0
Epoch 8, file 19: Loss = 374.0, Class. loss = 0.35, Class. acc. = 0.94, Loc.
loss = 394.0
Epoch 8, file 20: Loss = 246.0, Class. loss = 0.05, Class. acc. = 0.96, Loc.
loss = 259.0
Epoch 8, file 21: Loss = 399.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 420.0
Epoch 8, valid.: Loss = 958.0, Class. loss = 0.84, Class. acc. = 0.9, Loc. loss
= 1009.0
Epoch 9, file 0: Loss = 427.0, Class. loss = 0.46, Class. acc. = 0.94, Loc. loss
= 450.0
Epoch 9, file 1: Loss = 339.0, Class. loss = 0.43, Class. acc. = 0.96, Loc. loss
= 357.0
Epoch 9, file 2: Loss = 424.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
446.0
Epoch 9, file 3: Loss = 299.0, Class. loss = 0.02, Class. acc. = 1.0, Loc. loss
= 315.0
Epoch 9, file 4: Loss = 369.0, Class. loss = 0.03, Class. acc. = 0.98, Loc. loss
= 388.0
Epoch 9, file 5: Loss = 758.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
797.0
Epoch 9, file 6: Loss = 289.0, Class. loss = 0.71, Class. acc. = 0.94, Loc. loss
= 304.0
```

```
Epoch 9, file 7: Loss = 444.0, Class. loss = 0.02, Class. acc. = 0.98, Loc. loss
= 467.0
Epoch 9, file 8: Loss = 569.0, Class. loss = 0.06, Class. acc. = 0.98, Loc. loss
= 599.0
Epoch 9, file 9: Loss = 479.0, Class. loss = 0.07, Class. acc. = 0.98, Loc. loss
= 504.0
Epoch 9, file 10: Loss = 325.0, Class. loss = 0.18, Class. acc. = 0.98, Loc.
loss = 342.0
Epoch 9, file 11: Loss = 441.0, Class. loss = 0.45, Class. acc. = 0.94, Loc.
loss = 464.0
Epoch 9, file 12: Loss = 270.0, Class. loss = 0.07, Class. acc. = 0.98, Loc.
loss = 284.0
Epoch 9, file 13: Loss = 432.0, Class. loss = 0.15, Class. acc. = 0.98, Loc.
loss = 455.0
Epoch 9, file 14: Loss = 234.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 247.0
Epoch 9, file 15: Loss = 386.0, Class. loss = 0.15, Class. acc. = 0.98, Loc.
loss = 407.0
Epoch 9, file 16: Loss = 384.0, Class. loss = 0.23, Class. acc. = 0.92, Loc.
loss = 405.0
Epoch 9, file 17: Loss = 250.0, Class. loss = 0.06, Class. acc. = 0.98, Loc.
loss = 263.0
Epoch 9, file 18: Loss = 356.0, Class. loss = 0.43, Class. acc. = 0.92, Loc.
loss = 375.0
Epoch 9, file 19: Loss = 433.0, Class. loss = 0.27, Class. acc. = 0.94, Loc.
loss = 456.0
Epoch 9, file 20: Loss = 354.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 373.0
Epoch 9, file 21: Loss = 289.0, Class. loss = 0.61, Class. acc. = 0.98, Loc.
loss = 304.0
Epoch 9, valid.: Loss = 862.0, Class. loss = 1.07, Class. acc. = 0.85, Loc. loss
= 907.0
```

Plot the training history and Test the model

`custom_training_history` takes the output metics (accuracy) and plots for classification and localisation so we can see how the accuracy improved over epochs

```
Y_class_test_cnn, Y_loc_test_cnn = vgg16_2head_fc.predict(X_test_btln, verbose = 1)
```

Generates the prediction labels which are given to our plotting function `plot_data_class_loc_caller`

To show actual and predicted classification and localisation of deformation from the interferograms.

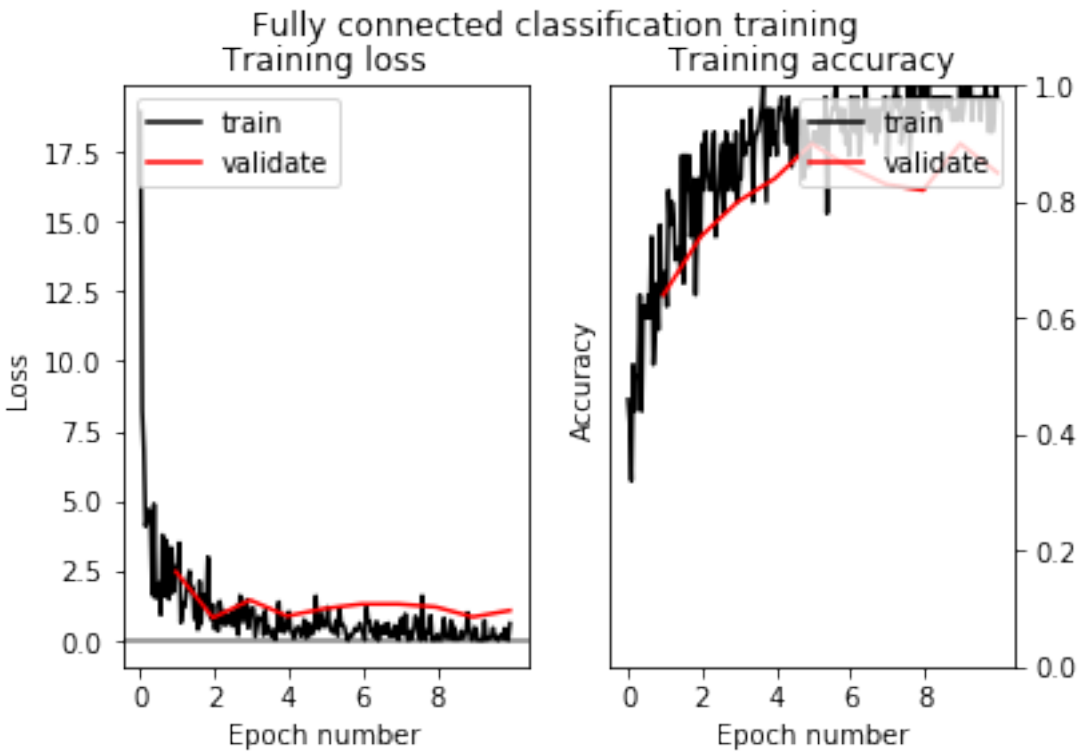
[23]:
```
def custom_training_history(metrics, n_epochs, title = None):
    """Plot training line graphs for loss and accuracy.  Loss on the left,␣
 ↪accuracy on the right.
    Inputs
```

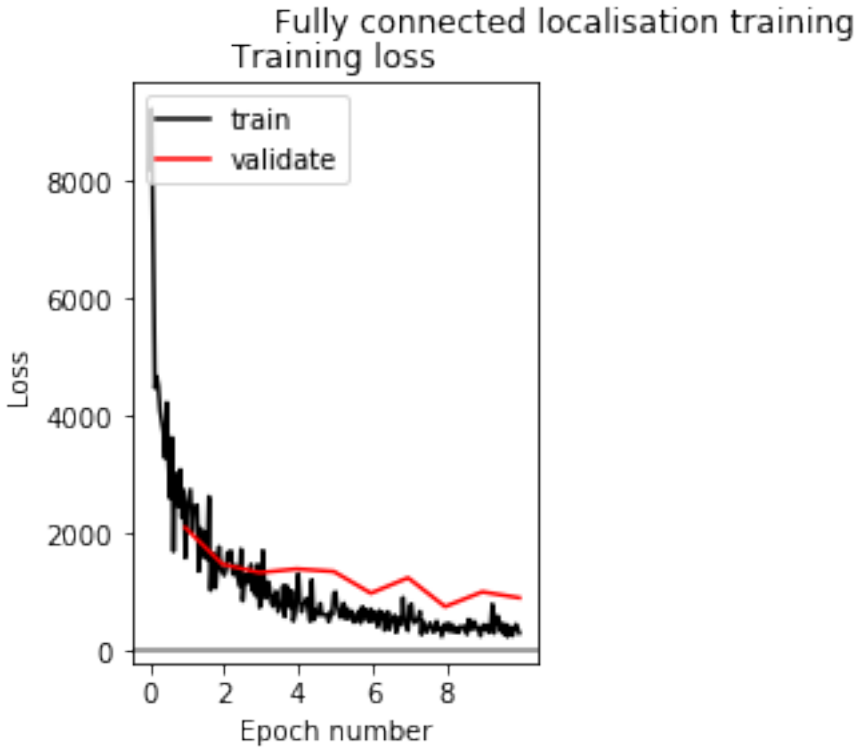
```
        metrics | r2 array | (n_files * n_epochs) x 2 or 4 matrix,  train␣
↪loss/validate loss |
        train accuracy/validate accuracy.
        If no accuracy, only 2 columns
        n_epochs | int | number of epochs model was trained for
        title | string | title
    Returns:
        Figure
    """


    if metrics.shape[1] == 4:        # detemrine if we have accuracy as well as␣
↪loss
        accuracy_flag = True
    else:
        accuracy_flag = False


    n_files = metrics.shape[0] / n_epochs
    # Figure output
    fig1, axes = plt.subplots(1,2)
    fig1.canvas.set_window_title(title)
    fig1.suptitle(title)
    xvals = np.arange(0,metrics.shape[0])
    # fewer validation data; find which ones to plot
    validation_plot = np.ravel(np.argwhere(metrics[:,1] > 1e-10))
    axes[0].plot(xvals, metrics[:,0], c = 'k')                                  ␣
↪     # training loss
    axes[0].plot(xvals[validation_plot], metrics[validation_plot,1], c = 'r')  ␣
↪     # validation loss
    axes[0].set_ylabel('Loss')
    axes[0].legend(['train', 'validate'], loc='upper left')
    axes[0].axhline(y=0, color='k', alpha=0.5)

    if accuracy_flag:
        axes[1].plot(xvals, metrics[:,2], c = 'k')                              ␣
↪         # training accuracy
        axes[1].plot(xvals[validation_plot], metrics[validation_plot,3], c =␣
↪'r')         # validation accuracy
        axes[1].set_ylim([0,1])
        axes[1].set_ylabel('Accuracy')
        axes[1].yaxis.tick_right()
        axes[1].legend(['train', 'validate'], loc='upper right')

    #
    titles = ['Training loss', 'Training accuracy']
```

```
    for i in range(2):
        axes[i].set_title(titles[i])
        # change so a tick only after each epoch (and not each file)
        axes[i].set_xticks(np.arange(0,metrics.shape[0],2*n_files))
        axes[i].set_xticklabels(np.arange(0,n_epochs, 2))   # number ticks
        axes[i].set_xlabel('Epoch number')

    if not accuracy_flag:
        axes[1].set_visible(False)
```

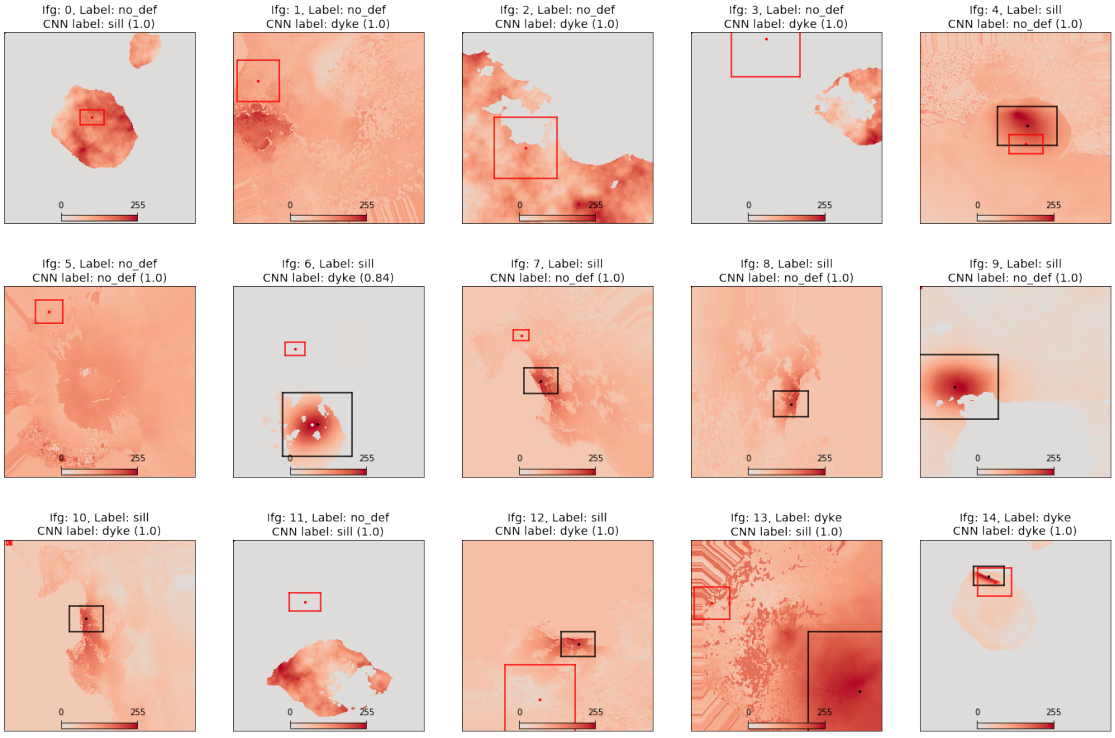
[24]:
```
# plot of the training process for classification
custom_training_history(metrics_class_fc, n_epochs_fc, title = 'Fully connected␣
 ↪classification training')
# plot of the training process for localisation
custom_training_history(metrics_localisation_fc, n_epochs_fc, title = 'Fully␣
 ↪connected localisation training')
# save the weights of the model we have trained
vgg16_2head_fc.save_weights(f'data/train_fully_connected_model/vgg16_2head_fc.
 ↪h5')


# Test the model
# forward pass of the testing data bottleneck features through the fully␣
 ↪connected part of the model
Y_class_test_cnn, Y_loc_test_cnn = vgg16_2head_fc.predict(X_test_btln, verbose␣
 ↪= 1)
```

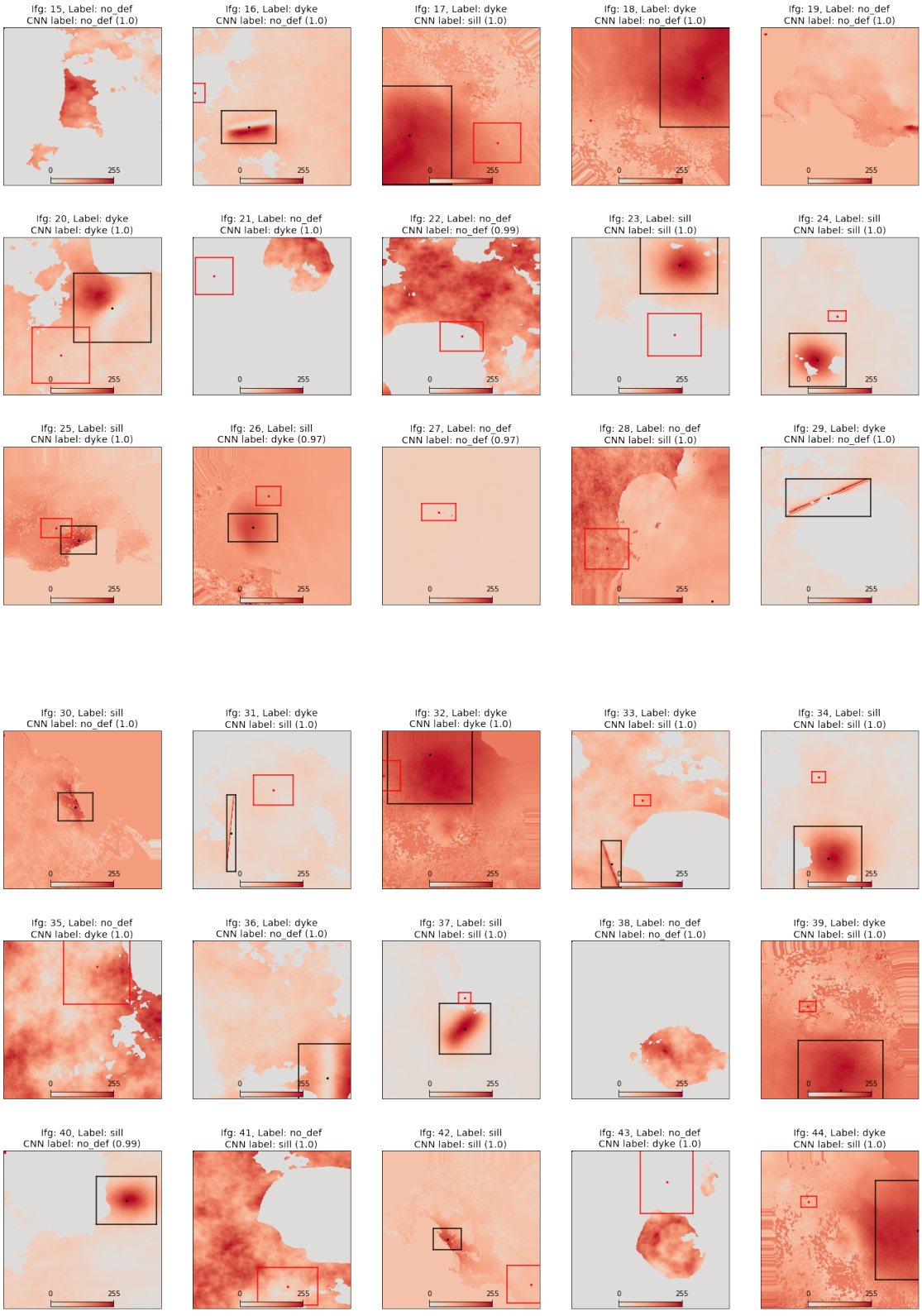
```
4/4 [==============================] - 0s 65ms/step
```

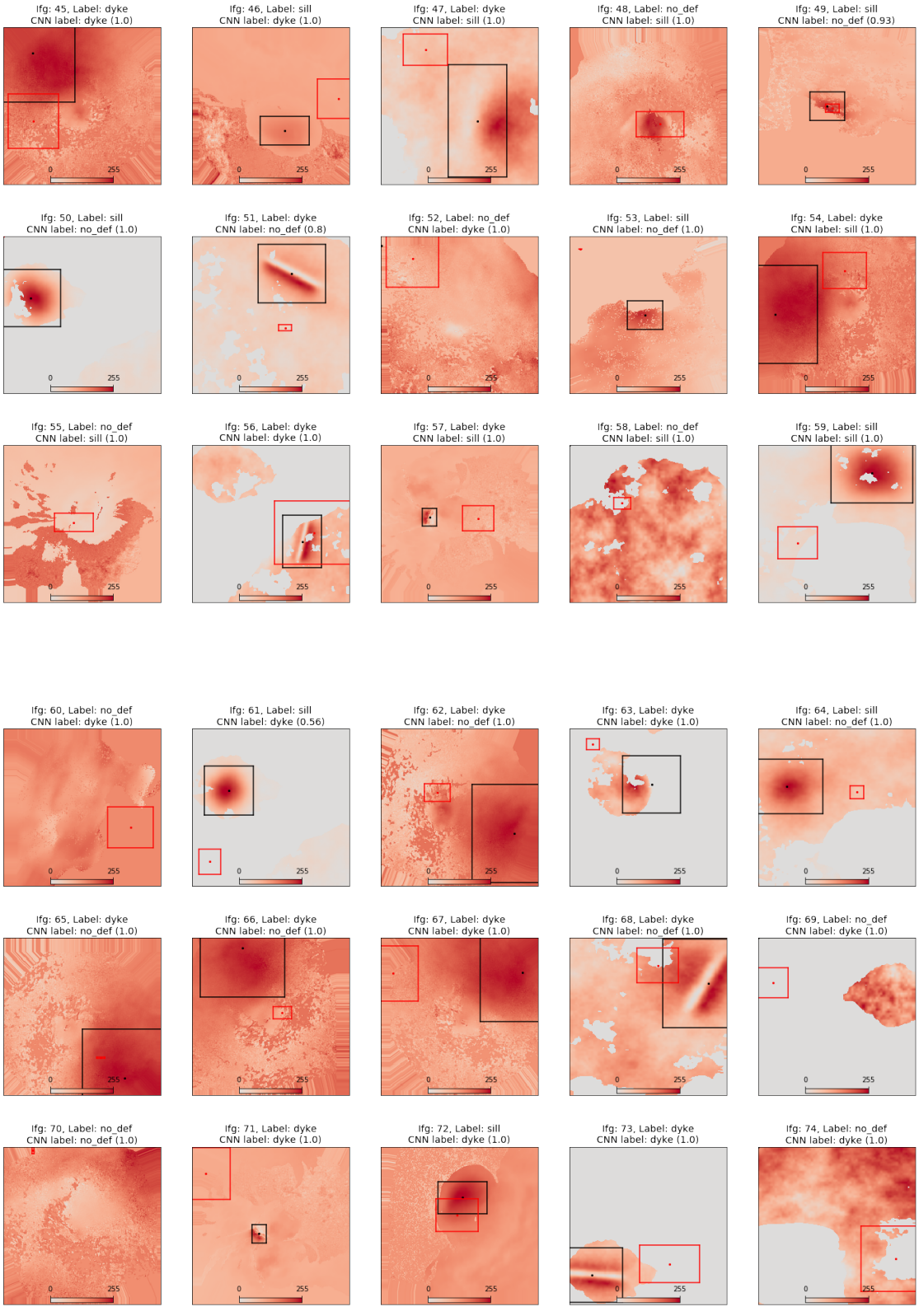

Fully connected classification training
Training loss
Training accuracy
train
validate
Loss
Accuracy
Epoch number
Epoch number

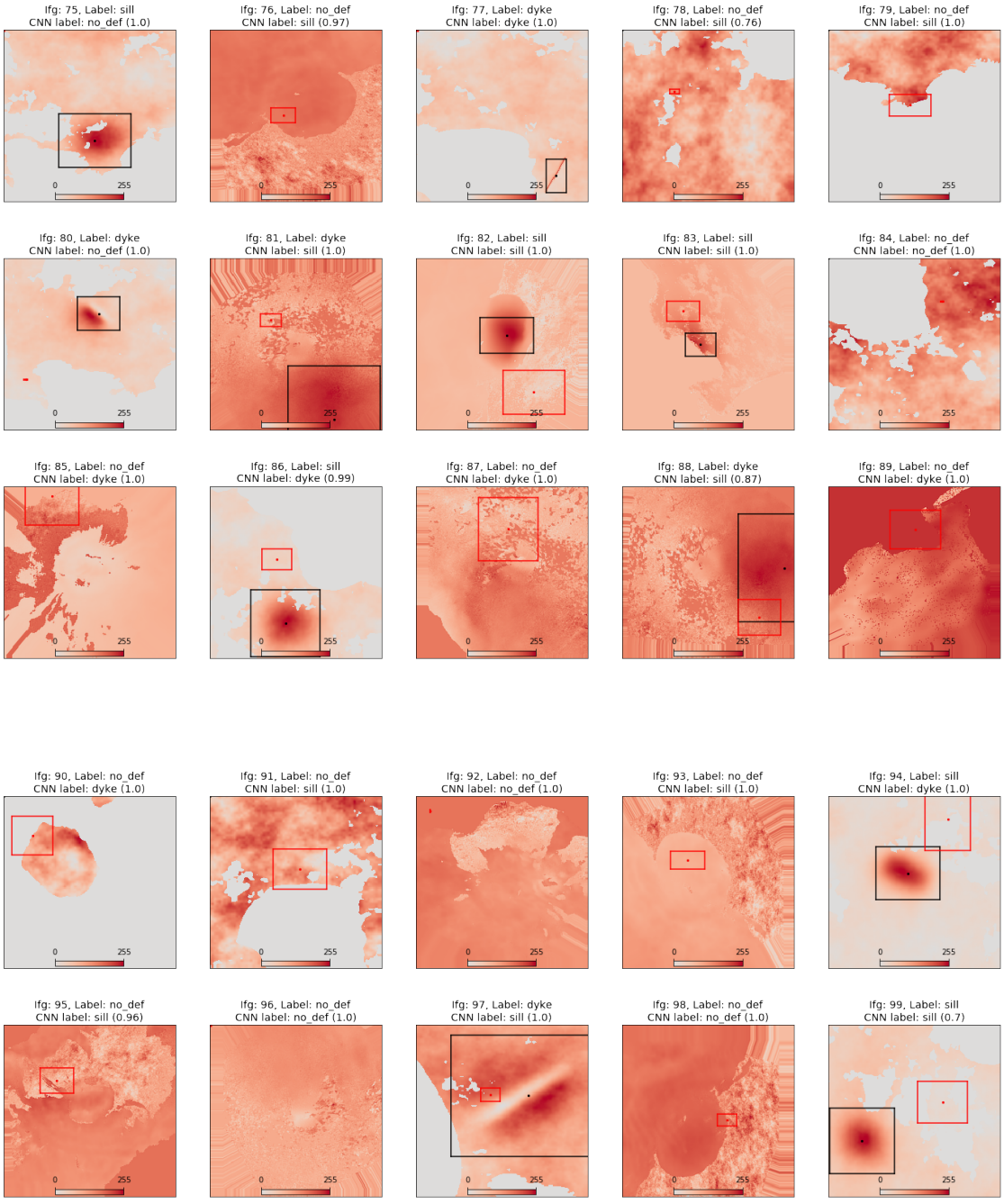

Fully connected localisation training
Training loss
train
validate
Loss
Epoch number

[25]:
```
plot_data_class_loc_caller(X_test, classes = Y_class_test, classes_predicted =␣
↪Y_class_test_cnn,
                           # plot all the testing data
                           locs = Y_loc_test, locs_predicted = Y_loc_test_cnn,
                           source_names =␣
↪synthetic_ifgs_settings['defo_sources'],
                           window_title = 'Testing data')
```




Ifg: 15, Label: no_def
CNN label: no_def (1.0)
Ifg: 16, Label: dyke
CNN label: no_def (1.0)
Ifg: 17, Label: dyke
CNN label: sill (1.0)
Ifg: 18, Label: dyke
CNN label: no_def (1.0)
Ifg: 19, Label: no_def
CNN label: no_def (1.0)
Ifg: 20, Label: dyke
CNN label: dyke (1.0)
Ifg: 21, Label: no_def
CNN label: dyke (1.0)
Ifg: 22, Label: no_def
CNN label: no_def (0.99)
Ifg: 23, Label: sill
CNN label: sill (1.0)
Ifg: 24, Label: sill
CNN label: sill (1.0)
Ifg: 25, Label: sill
CNN label: dyke (1.0)
Ifg: 26, Label: sill
CNN label: dyke (0.97)
Ifg: 27, Label: no_def
CNN label: no_def (0.97)
Ifg: 28, Label: no_def
CNN label: sill (1.0)
Ifg: 29, Label: dyke
CNN label: no_def (1.0)
Ifg: 30, Label: sill
CNN label: no_def (1.0)
Ifg: 31, Label: dyke
CNN label: sill (1.0)
Ifg: 32, Label: dyke
CNN label: dyke (1.0)
Ifg: 33, Label: dyke
CNN label: sill (1.0)
Ifg: 34, Label: sill
CNN label: sill (1.0)
Ifg: 35, Label: no_def
CNN label: dyke (1.0)
Ifg: 36, Label: dyke
CNN label: no_def (1.0)
Ifg: 37, Label: sill
CNN label: sill (1.0)
Ifg: 38, Label: no_def
CNN label: no_def (1.0)
Ifg: 39, Label: dyke
CNN label: sill (1.0)
Ifg: 40, Label: sill
CNN label: no_def (0.99)
Ifg: 41, Label: no_def
CNN label: sill (1.0)
Ifg: 42, Label: sill
CNN label: sill (1.0)
Ifg: 43, Label: no_def
CNN label: dyke (1.0)
Ifg: 44, Label: dyke
CNN label: sill (1.0)
0
255


Ifg: 45, Label: dyke
CNN label: dyke (1.0)
Ifg: 46, Label: sill
CNN label: dyke (1.0)
Ifg: 47, Label: dyke
CNN label: sill (1.0)
Ifg: 48, Label: no_def
CNN label: sill (1.0)
Ifg: 49, Label: sill
CNN label: no_def (0.93)
Ifg: 50, Label: sill
CNN label: no_def (1.0)
Ifg: 51, Label: dyke
CNN label: no_def (0.8)
Ifg: 52, Label: no_def
CNN label: dyke (1.0)
Ifg: 53, Label: sill
CNN label: no_def (1.0)
Ifg: 54, Label: dyke
CNN label: sill (1.0)
Ifg: 55, Label: no_def
CNN label: sill (1.0)
Ifg: 56, Label: dyke
CNN label: dyke (1.0)
Ifg: 57, Label: dyke
CNN label: sill (1.0)
Ifg: 58, Label: no_def
CNN label: sill (1.0)
Ifg: 59, Label: sill
CNN label: sill (1.0)
Ifg: 60, Label: no_def
CNN label: dyke (1.0)
Ifg: 61, Label: sill
CNN label: dyke (0.56)
Ifg: 62, Label: dyke
CNN label: no_def (1.0)
Ifg: 63, Label: dyke
CNN label: dyke (1.0)
Ifg: 64, Label: sill
CNN label: no_def (1.0)
Ifg: 65, Label: dyke
CNN label: no_def (1.0)
Ifg: 66, Label: dyke
CNN label: no_def (1.0)
Ifg: 67, Label: dyke
CNN label: dyke (1.0)
Ifg: 68, Label: dyke
CNN label: no_def (1.0)
Ifg: 69, Label: no_def
CNN label: dyke (1.0)
Ifg: 70, Label: no_def
CNN label: no_def (1.0)
Ifg: 71, Label: dyke
CNN label: dyke (1.0)
Ifg: 72, Label: sill
CNN label: dyke (1.0)
Ifg: 73, Label: dyke
CNN label: dyke (1.0)
Ifg: 74, Label: no_def
CNN label: dyke (1.0)



Fine Tuning

The model performs reasonably well but the above plot shows it could do with some fine-tuning

1. The model's overall loss is now a combination of the classification and localisation loss, which must be balanced using a hyperparameter commonly termed loss weighting. Experimenting with this found that a value of 0.95 for the classification loss and 0.05 for the localisation loss provided a good balance between the two outputs as the localisation loss is significantly

larger than the classification loss.

```
block5_loss_weights = [0.05, 0.95]
```

2. As a new style of training is occurring after the 10th epoch the learning rate needs to be carefully selected (too quick and the fine-tuning in both the convolutional blocks of VGG16 and our fully connected classification and localisation heads can get destroyed

```
block5_lr = 1.0e-6
```

3. Swithcing the optimizer to stochastic gradient descent (SGD)

```
block5_optimiser = optimizers.SGD(lr=block5_lr, momentum=0.9)
```

This might take a while

[26]:
```
# Fine-tune the 5th block and the fully connected part of the network):

# as per fc_loss_weights, but by changing these more emphasis can be placed on␣
 ↪either the clasification
# or localisation loss.
block5_loss_weights = [0.05, 0.95]

# We have to set a learning rate manually as an adaptive approach (e.g. NADAM)␣
 ↪will be high initially,
# and therefore make large updates that will wreck the model (as we're just␣
 ↪fine-tuning a model so have
# something good to start with)
block5_lr = 1.5e-8

# the number of epochs to fine-tune for
# (ie. the number of times all the training data are passed through the model)
n_epochs_block5 = 10

np.random.seed(0)                    # 0 used in the example

#%% Fine-tune the 5th convolutional block and the fully connected network.

# VGG16 is used for its convolutional layers and weights (but no fully␣
 ↪connected part as we define out own )
vgg16_block_1to5 = VGG16(weights='imagenet', include_top=False, input_shape =␣
 ↪(224,224,3))
# build the fully connected part of the model, and get the two model outputs
output_class, output_loc = define_two_head_model(vgg16_block_1to5.output,
                                                 ␣
 ↪len(synthetic_ifgs_settings['defo_sources']))
```

```
vgg16_2head = Model(inputs=vgg16_block_1to5.input, outputs=[output_class,␣
↪output_loc])                                                    # define the full␣
↪model
vgg16_2head.load_weights(f'data/train_fully_connected_model/vgg16_2head_fc.h5',␣
↪by_name = True)                                  # load the weights for the␣
↪fully connected part which were trained in step 06 (by_name flag so that it␣
↪doesn't matter that the models are different sizes))

for layer in vgg16_2head.layers[:15]:
                                                                                 ␣
↪                                                  # freeze blocks 1-4 (ie,␣
↪we are only fine tuneing the 5th block and the fully connected part of the␣
↪network)
    layer.trainable = False


# set the optimizer used in this training part.
# Note have to set a learning rate manualy as an adaptive one (eg Nadam)
# would wreck model weights in the first few passes before it reduced.
block5_optimiser = optimizers.SGD(lr=block5_lr, momentum=0.9)

vgg16_2head.compile(optimizer = block5_optimiser, metrics=['accuracy'],
                    # recompile as we've changed which layers can be trained/␣
↪optimizer etc.
                    loss=[loss_class, loss_loc], loss_weights =␣
↪block5_loss_weights)

try:
    plot_model(vgg16_2head, to_file='vgg16_2head.png', show_shapes = True,␣
↪show_layer_names = True)
    # try to make a graphviz style image showing the complete model
except:
    print(f"Failed to create a .png of the model, but continuing anyway.  ")
    vgg16_2head_fc.summary()
    # this can easily fail, however, so simply alert the user and continue.


print('\n\nFine-tuning the 5th convolutional block and the fully connected␣
↪network.')

[vgg16_2head, metrics_class_5th,
 metrics_localisation_5th, metrics_combined_loss_5th] =␣
↪train_double_network(vgg16_2head, data_files_train,
                                                      n_epochs_block5,␣
↪['class_dense3_loss', 'loc_dense6_loss'],
                                                      X_validate,␣
↪Y_class_validate, Y_loc_validate,
```

```
                                                                    ␣
 ↪len(synthetic_ifgs_settings['defo_sources']))
custom_training_history(metrics_class_5th, n_epochs_block5, title = '5th block␣
 ↪classification training')
custom_training_history(metrics_localisation_5th, n_epochs_block5, title = '5th␣
 ↪block localisation training')
try:
    os.mkdir((Path(f"./data/train_full_model")))
except:
    pass
vgg16_2head.save(f'data/train_full_model/01_vgg16_2head_block5_trained.h5')
np.savez(f'data/train_full_model/training_history.npz', metrics_class_fc =␣
 ↪metrics_class_fc,
                                                                    ␣
 ↪metrics_localisation_fc = metrics_localisation_fc,
                                                                    ␣
 ↪metrics_combined_loss_fc = metrics_combined_loss_fc,
                                                                     metrics_class_5th =␣
 ↪metrics_class_5th,
                                                                    ␣
 ↪metrics_localisation_5th = metrics_localisation_5th,
                                                                    ␣
 ↪metrics_combined_loss_5th = metrics_combined_loss_5th)
```

```
Fine-tuning the 5th convolutional block and the fully connected network.
Epoch 0, file 0: Loss = 236.0, Class. loss = 0.21, Class. acc. = 0.96, Loc. loss
= 249.0
Epoch 0, file 1: Loss = 257.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
271.0
Epoch 0, file 2: Loss = 268.0, Class. loss = 0.48, Class. acc. = 0.96, Loc. loss
= 283.0
Epoch 0, file 3: Loss = 183.0, Class. loss = 0.06, Class. acc. = 0.96, Loc. loss
= 193.0
Epoch 0, file 4: Loss = 192.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
202.0
Epoch 0, file 5: Loss = 180.0, Class. loss = 0.07, Class. acc. = 0.96, Loc. loss
= 189.0
Epoch 0, file 6: Loss = 953.0, Class. loss = 1.81, Class. acc. = 0.82, Loc. loss
= 1003.0
Epoch 0, file 7: Loss = 754.0, Class. loss = 1.29, Class. acc. = 0.8, Loc. loss
= 794.0
Epoch 0, file 8: Loss = 707.0, Class. loss = 2.0, Class. acc. = 0.8, Loc. loss =
744.0
Epoch 0, file 9: Loss = 1346.0, Class. loss = 1.49, Class. acc. = 0.84, Loc.
loss = 1417.0
```

```
Epoch 0, file 10: Loss = 260.0, Class. loss = 0.09, Class. acc. = 0.98, Loc.
loss = 274.0
Epoch 0, file 11: Loss = 202.0, Class. loss = 0.07, Class. acc. = 0.96, Loc.
loss = 213.0
Epoch 0, file 12: Loss = 286.0, Class. loss = 0.11, Class. acc. = 0.96, Loc.
loss = 301.0
Epoch 0, file 13: Loss = 316.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 333.0
Epoch 0, file 14: Loss = 331.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 349.0
Epoch 0, file 15: Loss = 228.0, Class. loss = 0.36, Class. acc. = 0.98, Loc.
loss = 240.0
Epoch 0, file 16: Loss = 286.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 301.0
Epoch 0, file 17: Loss = 184.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 194.0
Epoch 0, file 18: Loss = 205.0, Class. loss = 0.23, Class. acc. = 0.96, Loc.
loss = 216.0
Epoch 0, file 19: Loss = 167.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 176.0
Epoch 0, file 20: Loss = 235.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 248.0
Epoch 0, file 21: Loss = 167.0, Class. loss = 0.23, Class. acc. = 0.98, Loc.
loss = 175.0
Epoch 0, valid.: Loss = 5776.0, Class. loss = 28.83, Class. acc. = 0.29, Loc.
loss = 6078.0
Epoch 1, file 0: Loss = 298.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
313.0
Epoch 1, file 1: Loss = 266.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
279.0
Epoch 1, file 2: Loss = 292.0, Class. loss = 0.87, Class. acc. = 0.92, Loc. loss
= 307.0
Epoch 1, file 3: Loss = 160.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
168.0
Epoch 1, file 4: Loss = 180.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
189.0
Epoch 1, file 5: Loss = 135.0, Class. loss = 0.18, Class. acc. = 0.96, Loc. loss
= 142.0
Epoch 1, file 6: Loss = 890.0, Class. loss = 2.23, Class. acc. = 0.76, Loc. loss
= 937.0
Epoch 1, file 7: Loss = 614.0, Class. loss = 1.19, Class. acc. = 0.82, Loc. loss
= 646.0
Epoch 1, file 8: Loss = 525.0, Class. loss = 1.12, Class. acc. = 0.88, Loc. loss
= 552.0
Epoch 1, file 9: Loss = 1290.0, Class. loss = 1.42, Class. acc. = 0.86, Loc.
loss = 1358.0
Epoch 1, file 10: Loss = 214.0, Class. loss = 0.34, Class. acc. = 0.96, Loc.
loss = 226.0
```

```
Epoch 1, file 11: Loss = 167.0, Class. loss = 0.13, Class. acc. = 0.98, Loc.
loss = 176.0
Epoch 1, file 12: Loss = 332.0, Class. loss = 0.05, Class. acc. = 0.98, Loc.
loss = 349.0
Epoch 1, file 13: Loss = 299.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 315.0
Epoch 1, file 14: Loss = 282.0, Class. loss = 0.08, Class. acc. = 0.96, Loc.
loss = 297.0
Epoch 1, file 15: Loss = 187.0, Class. loss = 0.21, Class. acc. = 0.98, Loc.
loss = 197.0
Epoch 1, file 16: Loss = 239.0, Class. loss = 0.02, Class. acc. = 0.98, Loc.
loss = 251.0
Epoch 1, file 17: Loss = 141.0, Class. loss = 0.09, Class. acc. = 0.96, Loc.
loss = 148.0
Epoch 1, file 18: Loss = 222.0, Class. loss = 0.31, Class. acc. = 0.98, Loc.
loss = 233.0
Epoch 1, file 19: Loss = 198.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 208.0
Epoch 1, file 20: Loss = 214.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 225.0
Epoch 1, file 21: Loss = 167.0, Class. loss = 0.11, Class. acc. = 0.98, Loc.
loss = 176.0
Epoch 1, valid.: Loss = 5784.0, Class. loss = 28.77, Class. acc. = 0.29, Loc.
loss = 6087.0
Epoch 2, file 0: Loss = 176.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
186.0
Epoch 2, file 1: Loss = 199.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
210.0
Epoch 2, file 2: Loss = 235.0, Class. loss = 0.46, Class. acc. = 0.96, Loc. loss
= 248.0
Epoch 2, file 3: Loss = 141.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 149.0
Epoch 2, file 4: Loss = 162.0, Class. loss = 0.05, Class. acc. = 0.96, Loc. loss
= 170.0
Epoch 2, file 5: Loss = 125.0, Class. loss = 0.75, Class. acc. = 0.94, Loc. loss
= 131.0
Epoch 2, file 6: Loss = 868.0, Class. loss = 1.79, Class. acc. = 0.8, Loc. loss
= 913.0
Epoch 2, file 7: Loss = 543.0, Class. loss = 0.75, Class. acc. = 0.84, Loc. loss
= 572.0
Epoch 2, file 8: Loss = 500.0, Class. loss = 1.38, Class. acc. = 0.84, Loc. loss
= 526.0
Epoch 2, file 9: Loss = 1249.0, Class. loss = 1.58, Class. acc. = 0.88, Loc.
loss = 1315.0
Epoch 2, file 10: Loss = 196.0, Class. loss = 0.14, Class. acc. = 0.96, Loc.
loss = 206.0
Epoch 2, file 11: Loss = 204.0, Class. loss = 0.3, Class. acc. = 0.98, Loc. loss
= 215.0
```

```
Epoch 2, file 12: Loss = 253.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 266.0
Epoch 2, file 13: Loss = 291.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 306.0
Epoch 2, file 14: Loss = 254.0, Class. loss = 0.03, Class. acc. = 0.98, Loc.
loss = 267.0
Epoch 2, file 15: Loss = 200.0, Class. loss = 0.44, Class. acc. = 0.96, Loc.
loss = 210.0
Epoch 2, file 16: Loss = 229.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 241.0
Epoch 2, file 17: Loss = 149.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 156.0
Epoch 2, file 18: Loss = 234.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 246.0
Epoch 2, file 19: Loss = 187.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 197.0
Epoch 2, file 20: Loss = 209.0, Class. loss = 0.06, Class. acc. = 0.98, Loc.
loss = 220.0
Epoch 2, file 21: Loss = 148.0, Class. loss = 0.18, Class. acc. = 0.98, Loc.
loss = 156.0
Epoch 2, valid.: Loss = 5750.0, Class. loss = 28.33, Class. acc. = 0.29, Loc.
loss = 6051.0
Epoch 3, file 0: Loss = 226.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
238.0
Epoch 3, file 1: Loss = 199.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
209.0
Epoch 3, file 2: Loss = 226.0, Class. loss = 0.05, Class. acc. = 0.98, Loc. loss
= 238.0
Epoch 3, file 3: Loss = 101.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
106.0
Epoch 3, file 4: Loss = 141.0, Class. loss = 0.02, Class. acc. = 1.0, Loc. loss
= 148.0
Epoch 3, file 5: Loss = 138.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 145.0
Epoch 3, file 6: Loss = 820.0, Class. loss = 2.45, Class. acc. = 0.76, Loc. loss
= 863.0
Epoch 3, file 7: Loss = 621.0, Class. loss = 0.84, Class. acc. = 0.86, Loc. loss
= 654.0
Epoch 3, file 8: Loss = 463.0, Class. loss = 1.09, Class. acc. = 0.82, Loc. loss
= 488.0
Epoch 3, file 9: Loss = 1157.0, Class. loss = 1.74, Class. acc. = 0.8, Loc. loss
= 1218.0
Epoch 3, file 10: Loss = 164.0, Class. loss = 0.36, Class. acc. = 0.96, Loc.
loss = 173.0
Epoch 3, file 11: Loss = 182.0, Class. loss = 0.22, Class. acc. = 0.98, Loc.
loss = 192.0
Epoch 3, file 12: Loss = 271.0, Class. loss = 0.46, Class. acc. = 0.9, Loc. loss
= 285.0
```

```
Epoch 3, file 13: Loss = 195.0, Class. loss = 0.18, Class. acc. = 0.98, Loc.
loss = 205.0
Epoch 3, file 14: Loss = 232.0, Class. loss = 0.54, Class. acc. = 0.94, Loc.
loss = 244.0
Epoch 3, file 15: Loss = 159.0, Class. loss = 0.43, Class. acc. = 0.94, Loc.
loss = 167.0
Epoch 3, file 16: Loss = 270.0, Class. loss = 0.05, Class. acc. = 0.98, Loc.
loss = 284.0
Epoch 3, file 17: Loss = 164.0, Class. loss = 0.25, Class. acc. = 0.96, Loc.
loss = 173.0
Epoch 3, file 18: Loss = 220.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 232.0
Epoch 3, file 19: Loss = 178.0, Class. loss = 0.1, Class. acc. = 0.96, Loc. loss
= 187.0
Epoch 3, file 20: Loss = 218.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 229.0
Epoch 3, file 21: Loss = 195.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 205.0
Epoch 3, valid.: Loss = 5745.0, Class. loss = 28.21, Class. acc. = 0.29, Loc.
loss = 6046.0
Epoch 4, file 0: Loss = 219.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
231.0
Epoch 4, file 1: Loss = 221.0, Class. loss = 0.16, Class. acc. = 0.96, Loc. loss
= 232.0
Epoch 4, file 2: Loss = 216.0, Class. loss = 0.39, Class. acc. = 0.96, Loc. loss
= 228.0
Epoch 4, file 3: Loss = 140.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 147.0
Epoch 4, file 4: Loss = 144.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
151.0
Epoch 4, file 5: Loss = 140.0, Class. loss = 0.2, Class. acc. = 0.96, Loc. loss
= 148.0
Epoch 4, file 6: Loss = 832.0, Class. loss = 1.71, Class. acc. = 0.72, Loc. loss
= 875.0
Epoch 4, file 7: Loss = 530.0, Class. loss = 1.23, Class. acc. = 0.82, Loc. loss
= 558.0
Epoch 4, file 8: Loss = 427.0, Class. loss = 1.58, Class. acc. = 0.8, Loc. loss
= 449.0
Epoch 4, file 9: Loss = 1137.0, Class. loss = 1.73, Class. acc. = 0.82, Loc.
loss = 1197.0
Epoch 4, file 10: Loss = 171.0, Class. loss = 0.36, Class. acc. = 0.98, Loc.
loss = 180.0
Epoch 4, file 11: Loss = 227.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 239.0
Epoch 4, file 12: Loss = 232.0, Class. loss = 0.05, Class. acc. = 0.98, Loc.
loss = 245.0
Epoch 4, file 13: Loss = 218.0, Class. loss = 0.09, Class. acc. = 0.98, Loc.
loss = 229.0
```

```
Epoch 4, file 14: Loss = 236.0, Class. loss = 0.34, Class. acc. = 0.96, Loc.
loss = 248.0
Epoch 4, file 15: Loss = 163.0, Class. loss = 0.76, Class. acc. = 0.94, Loc.
loss = 172.0
Epoch 4, file 16: Loss = 155.0, Class. loss = 0.12, Class. acc. = 0.98, Loc.
loss = 163.0
Epoch 4, file 17: Loss = 168.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 177.0
Epoch 4, file 18: Loss = 161.0, Class. loss = 0.52, Class. acc. = 0.94, Loc.
loss = 170.0
Epoch 4, file 19: Loss = 198.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 208.0
Epoch 4, file 20: Loss = 211.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 222.0
Epoch 4, file 21: Loss = 165.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 174.0
Epoch 4, valid.: Loss = 5778.0, Class. loss = 28.26, Class. acc. = 0.29, Loc.
loss = 6080.0
Epoch 5, file 0: Loss = 204.0, Class. loss = 0.12, Class. acc. = 0.98, Loc. loss
= 215.0
Epoch 5, file 1: Loss = 209.0, Class. loss = 0.02, Class. acc. = 0.98, Loc. loss
= 220.0
Epoch 5, file 2: Loss = 215.0, Class. loss = 0.08, Class. acc. = 0.98, Loc. loss
= 227.0
Epoch 5, file 3: Loss = 165.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
173.0
Epoch 5, file 4: Loss = 145.0, Class. loss = 0.02, Class. acc. = 1.0, Loc. loss
= 153.0
Epoch 5, file 5: Loss = 140.0, Class. loss = 0.05, Class. acc. = 0.98, Loc. loss
= 147.0
Epoch 5, file 6: Loss = 824.0, Class. loss = 2.21, Class. acc. = 0.74, Loc. loss
= 867.0
Epoch 5, file 7: Loss = 531.0, Class. loss = 0.74, Class. acc. = 0.9, Loc. loss
= 559.0
Epoch 5, file 8: Loss = 401.0, Class. loss = 1.38, Class. acc. = 0.88, Loc. loss
= 422.0
Epoch 5, file 9: Loss = 1154.0, Class. loss = 1.79, Class. acc. = 0.82, Loc.
loss = 1215.0
Epoch 5, file 10: Loss = 214.0, Class. loss = 0.06, Class. acc. = 0.98, Loc.
loss = 225.0
Epoch 5, file 11: Loss = 198.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 208.0
Epoch 5, file 12: Loss = 232.0, Class. loss = 0.06, Class. acc. = 0.98, Loc.
loss = 244.0
Epoch 5, file 13: Loss = 201.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 212.0
Epoch 5, file 14: Loss = 269.0, Class. loss = 0.02, Class. acc. = 1.0, Loc. loss
= 283.0
```

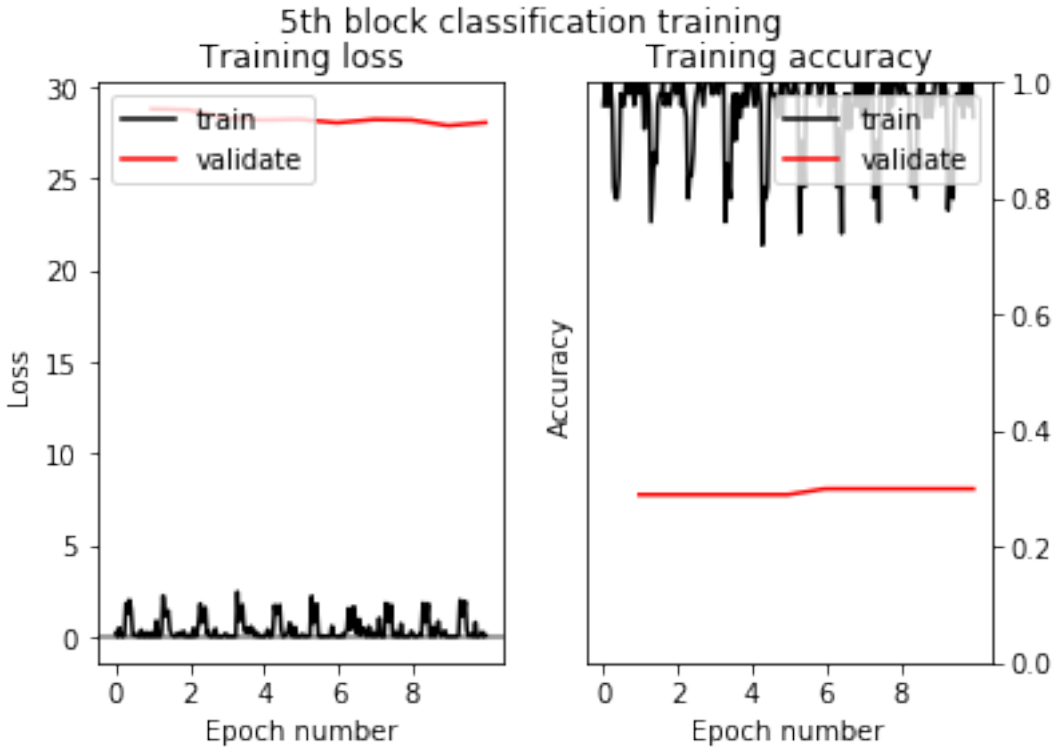
```
Epoch 5, file 15: Loss = 163.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 172.0
Epoch 5, file 16: Loss = 159.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 168.0
Epoch 5, file 17: Loss = 148.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 156.0
Epoch 5, file 18: Loss = 162.0, Class. loss = 0.58, Class. acc. = 0.96, Loc.
loss = 170.0
Epoch 5, file 19: Loss = 139.0, Class. loss = 0.09, Class. acc. = 0.98, Loc.
loss = 146.0
Epoch 5, file 20: Loss = 222.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 234.0
Epoch 5, file 21: Loss = 131.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 137.0
Epoch 5, valid.: Loss = 5781.0, Class. loss = 28.06, Class. acc. = 0.3, Loc.
loss = 6084.0
Epoch 6, file 0: Loss = 212.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
223.0
Epoch 6, file 1: Loss = 194.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
204.0
Epoch 6, file 2: Loss = 191.0, Class. loss = 0.12, Class. acc. = 0.94, Loc. loss
= 201.0
Epoch 6, file 3: Loss = 129.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
136.0
Epoch 6, file 4: Loss = 126.0, Class. loss = 0.2, Class. acc. = 0.96, Loc. loss
= 132.0
Epoch 6, file 5: Loss = 126.0, Class. loss = 0.41, Class. acc. = 0.98, Loc. loss
= 133.0
Epoch 6, file 6: Loss = 843.0, Class. loss = 1.54, Class. acc. = 0.82, Loc. loss
= 888.0
Epoch 6, file 7: Loss = 615.0, Class. loss = 0.37, Class. acc. = 0.92, Loc. loss
= 648.0
Epoch 6, file 8: Loss = 449.0, Class. loss = 0.89, Class. acc. = 0.86, Loc. loss
= 473.0
Epoch 6, file 9: Loss = 1124.0, Class. loss = 1.64, Class. acc. = 0.74, Loc.
loss = 1183.0
Epoch 6, file 10: Loss = 206.0, Class. loss = 0.37, Class. acc. = 0.96, Loc.
loss = 217.0
Epoch 6, file 11: Loss = 179.0, Class. loss = 0.15, Class. acc. = 0.98, Loc.
loss = 189.0
Epoch 6, file 12: Loss = 249.0, Class. loss = 0.96, Class. acc. = 0.92, Loc.
loss = 262.0
Epoch 6, file 13: Loss = 203.0, Class. loss = 0.08, Class. acc. = 0.98, Loc.
loss = 214.0
Epoch 6, file 14: Loss = 221.0, Class. loss = 0.49, Class. acc. = 0.92, Loc.
loss = 233.0
Epoch 6, file 15: Loss = 186.0, Class. loss = 0.46, Class. acc. = 0.96, Loc.
loss = 195.0
```

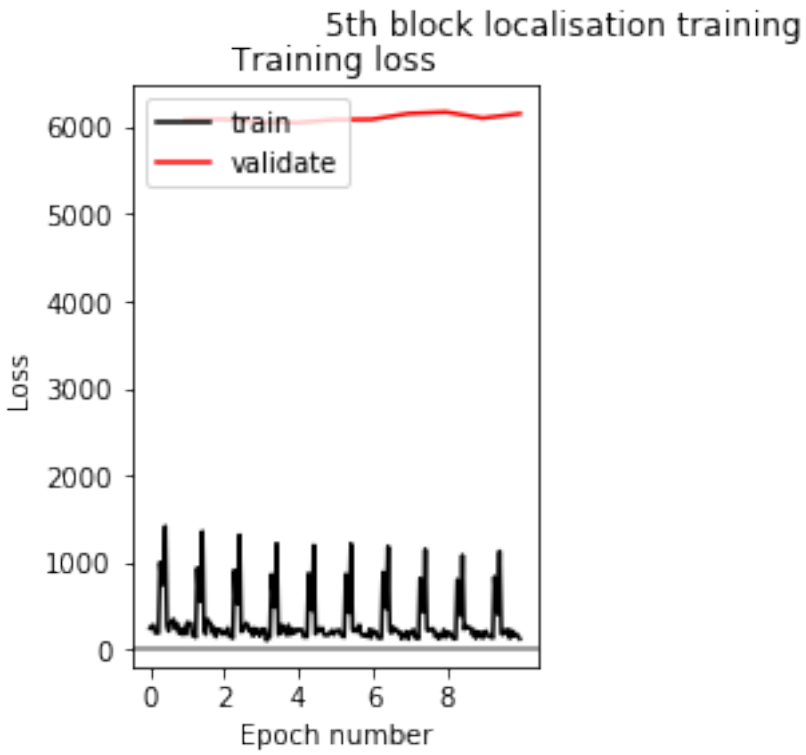
```
Epoch 6, file 16: Loss = 195.0, Class. loss = 0.06, Class. acc. = 0.96, Loc.
loss = 205.0
Epoch 6, file 17: Loss = 118.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 124.0
Epoch 6, file 18: Loss = 162.0, Class. loss = 0.55, Class. acc. = 0.96, Loc.
loss = 170.0
Epoch 6, file 19: Loss = 181.0, Class. loss = 0.08, Class. acc. = 0.96, Loc.
loss = 190.0
Epoch 6, file 20: Loss = 199.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 209.0
Epoch 6, file 21: Loss = 115.0, Class. loss = 0.23, Class. acc. = 0.94, Loc.
loss = 121.0
Epoch 6, valid.: Loss = 5844.0, Class. loss = 28.25, Class. acc. = 0.3, Loc.
loss = 6150.0
Epoch 7, file 0: Loss = 186.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
196.0
Epoch 7, file 1: Loss = 193.0, Class. loss = 0.14, Class. acc. = 0.98, Loc. loss
= 203.0
Epoch 7, file 2: Loss = 219.0, Class. loss = 1.01, Class. acc. = 0.94, Loc. loss
= 231.0
Epoch 7, file 3: Loss = 130.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
137.0
Epoch 7, file 4: Loss = 138.0, Class. loss = 0.08, Class. acc. = 0.98, Loc. loss
= 145.0
Epoch 7, file 5: Loss = 119.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
126.0
Epoch 7, file 6: Loss = 776.0, Class. loss = 1.82, Class. acc. = 0.8, Loc. loss
= 816.0
Epoch 7, file 7: Loss = 552.0, Class. loss = 0.91, Class. acc. = 0.84, Loc. loss
= 581.0
Epoch 7, file 8: Loss = 412.0, Class. loss = 0.83, Class. acc. = 0.9, Loc. loss
= 434.0
Epoch 7, file 9: Loss = 1093.0, Class. loss = 1.74, Class. acc. = 0.76, Loc.
loss = 1151.0
Epoch 7, file 10: Loss = 214.0, Class. loss = 0.1, Class. acc. = 0.98, Loc. loss
= 226.0
Epoch 7, file 11: Loss = 205.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 216.0
Epoch 7, file 12: Loss = 218.0, Class. loss = 0.25, Class. acc. = 0.98, Loc.
loss = 229.0
Epoch 7, file 13: Loss = 231.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 243.0
Epoch 7, file 14: Loss = 218.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 229.0
Epoch 7, file 15: Loss = 142.0, Class. loss = 0.7, Class. acc. = 0.94, Loc. loss
= 149.0
Epoch 7, file 16: Loss = 177.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 186.0
```

```
Epoch 7, file 17: Loss = 118.0, Class. loss = 0.04, Class. acc. = 0.96, Loc.
loss = 124.0
Epoch 7, file 18: Loss = 153.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 161.0
Epoch 7, file 19: Loss = 144.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 151.0
Epoch 7, file 20: Loss = 219.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 230.0
Epoch 7, file 21: Loss = 133.0, Class. loss = 0.3, Class. acc. = 0.98, Loc. loss
= 140.0
Epoch 7, valid.: Loss = 5862.0, Class. loss = 28.22, Class. acc. = 0.3, Loc.
loss = 6169.0
Epoch 8, file 0: Loss = 192.0, Class. loss = 0.17, Class. acc. = 0.98, Loc. loss
= 202.0
Epoch 8, file 1: Loss = 162.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
170.0
Epoch 8, file 2: Loss = 185.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
195.0
Epoch 8, file 3: Loss = 131.0, Class. loss = 0.05, Class. acc. = 0.98, Loc. loss
= 138.0
Epoch 8, file 4: Loss = 122.0, Class. loss = 0.1, Class. acc. = 0.98, Loc. loss
= 129.0
Epoch 8, file 5: Loss = 117.0, Class. loss = 0.03, Class. acc. = 0.98, Loc. loss
= 123.0
Epoch 8, file 6: Loss = 764.0, Class. loss = 1.82, Class. acc. = 0.82, Loc. loss
= 804.0
Epoch 8, file 7: Loss = 549.0, Class. loss = 0.77, Class. acc. = 0.86, Loc. loss
= 578.0
Epoch 8, file 8: Loss = 368.0, Class. loss = 1.01, Class. acc. = 0.88, Loc. loss
= 388.0
Epoch 8, file 9: Loss = 1031.0, Class. loss = 1.8, Class. acc. = 0.8, Loc. loss
= 1085.0
Epoch 8, file 10: Loss = 223.0, Class. loss = 0.04, Class. acc. = 0.98, Loc.
loss = 234.0
Epoch 8, file 11: Loss = 263.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 277.0
Epoch 8, file 12: Loss = 237.0, Class. loss = 0.53, Class. acc. = 0.98, Loc.
loss = 249.0
Epoch 8, file 13: Loss = 238.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 250.0
Epoch 8, file 14: Loss = 237.0, Class. loss = 0.32, Class. acc. = 0.94, Loc.
loss = 249.0
Epoch 8, file 15: Loss = 162.0, Class. loss = 0.83, Class. acc. = 0.94, Loc.
loss = 171.0
Epoch 8, file 16: Loss = 209.0, Class. loss = 0.15, Class. acc. = 0.98, Loc.
loss = 220.0
Epoch 8, file 17: Loss = 122.0, Class. loss = 0.02, Class. acc. = 0.98, Loc.
loss = 129.0
```

```
Epoch 8, file 18: Loss = 195.0, Class. loss = 0.14, Class. acc. = 0.94, Loc.
loss = 205.0
Epoch 8, file 19: Loss = 156.0, Class. loss = 0.33, Class. acc. = 0.96, Loc.
loss = 164.0
Epoch 8, file 20: Loss = 187.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 197.0
Epoch 8, file 21: Loss = 125.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 132.0
Epoch 8, valid.: Loss = 5795.0, Class. loss = 27.91, Class. acc. = 0.3, Loc.
loss = 6099.0
Epoch 9, file 0: Loss = 184.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss =
193.0
Epoch 9, file 1: Loss = 204.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 215.0
Epoch 9, file 2: Loss = 148.0, Class. loss = 0.03, Class. acc. = 0.96, Loc. loss
= 156.0
Epoch 9, file 3: Loss = 143.0, Class. loss = 0.02, Class. acc. = 1.0, Loc. loss
= 150.0
Epoch 9, file 4: Loss = 147.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 154.0
Epoch 9, file 5: Loss = 128.0, Class. loss = 0.07, Class. acc. = 0.98, Loc. loss
= 134.0
Epoch 9, file 6: Loss = 795.0, Class. loss = 1.99, Class. acc. = 0.78, Loc. loss
= 837.0
Epoch 9, file 7: Loss = 553.0, Class. loss = 1.12, Class. acc. = 0.84, Loc. loss
= 582.0
Epoch 9, file 8: Loss = 386.0, Class. loss = 1.29, Class. acc. = 0.88, Loc. loss
= 406.0
Epoch 9, file 9: Loss = 1073.0, Class. loss = 1.92, Class. acc. = 0.8, Loc. loss
= 1129.0
Epoch 9, file 10: Loss = 175.0, Class. loss = 0.32, Class. acc. = 0.96, Loc.
loss = 184.0
Epoch 9, file 11: Loss = 164.0, Class. loss = 0.03, Class. acc. = 0.98, Loc.
loss = 173.0
Epoch 9, file 12: Loss = 211.0, Class. loss = 0.06, Class. acc. = 0.98, Loc.
loss = 222.0
Epoch 9, file 13: Loss = 190.0, Class. loss = 0.13, Class. acc. = 0.96, Loc.
loss = 200.0
Epoch 9, file 14: Loss = 220.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 232.0
Epoch 9, file 15: Loss = 124.0, Class. loss = 0.78, Class. acc. = 0.94, Loc.
loss = 131.0
Epoch 9, file 16: Loss = 209.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 220.0
Epoch 9, file 17: Loss = 146.0, Class. loss = 0.01, Class. acc. = 1.0, Loc. loss
= 153.0
Epoch 9, file 18: Loss = 172.0, Class. loss = 0.07, Class. acc. = 0.96, Loc.
loss = 181.0
```

```
Epoch 9, file 19: Loss = 172.0, Class. loss = 0.26, Class. acc. = 0.96, Loc.
loss = 182.0
Epoch 9, file 20: Loss = 141.0, Class. loss = 0.0, Class. acc. = 1.0, Loc. loss
= 148.0
Epoch 9, file 21: Loss = 119.0, Class. loss = 0.07, Class. acc. = 0.94, Loc.
loss = 125.0
Epoch 9, valid.: Loss = 5844.0, Class. loss = 28.07, Class. acc. = 0.3, Loc.
loss = 6150.0
```

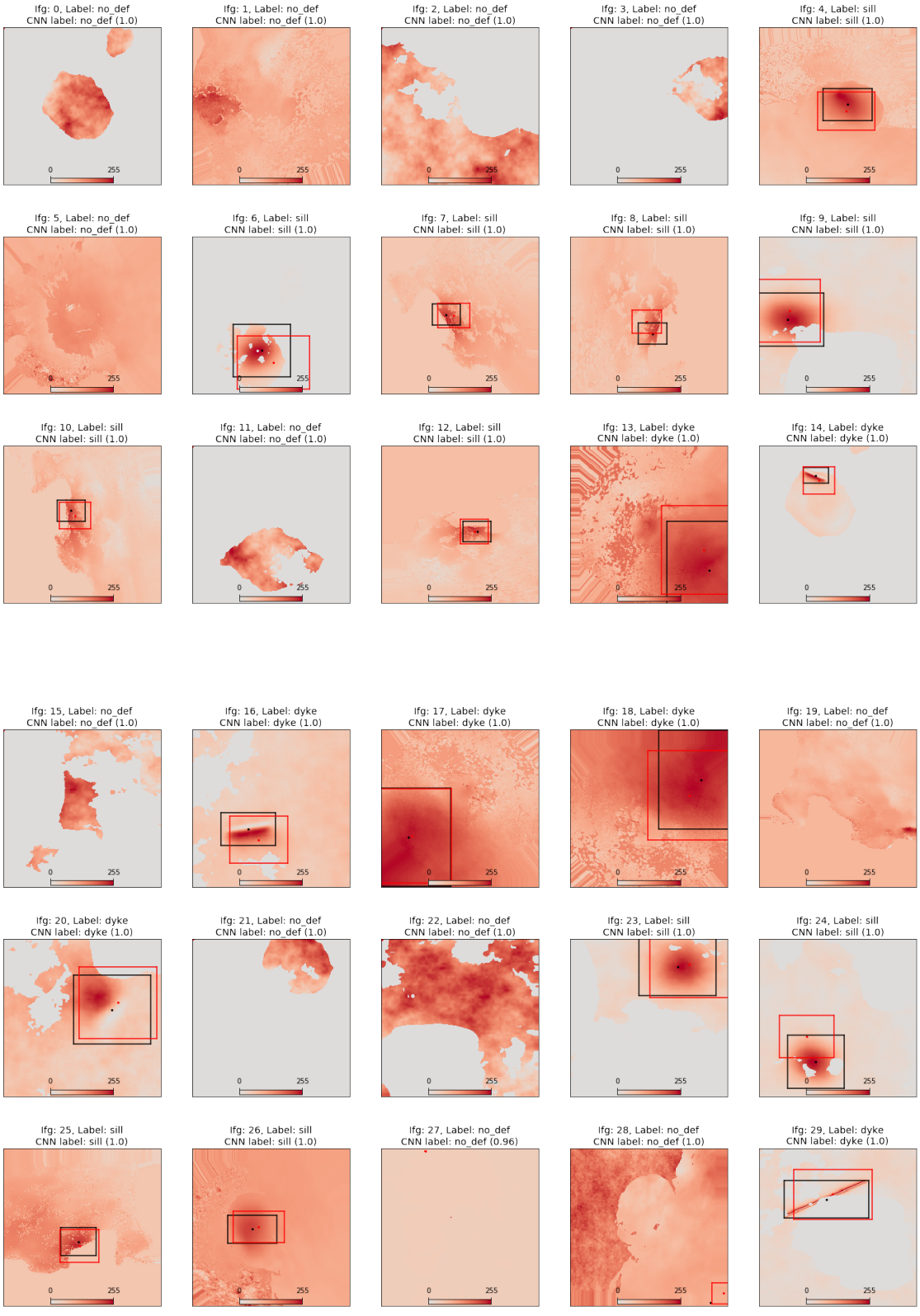


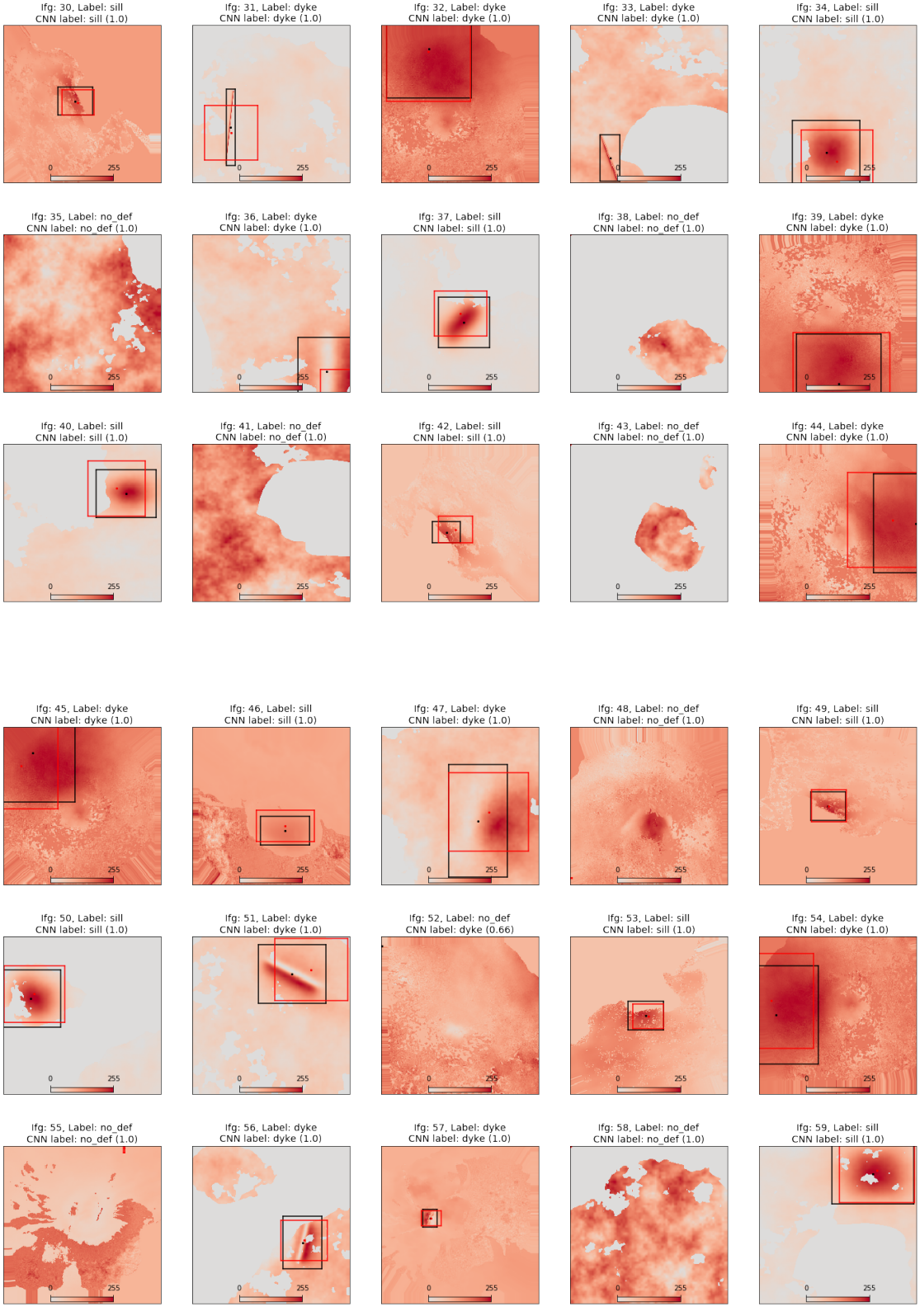


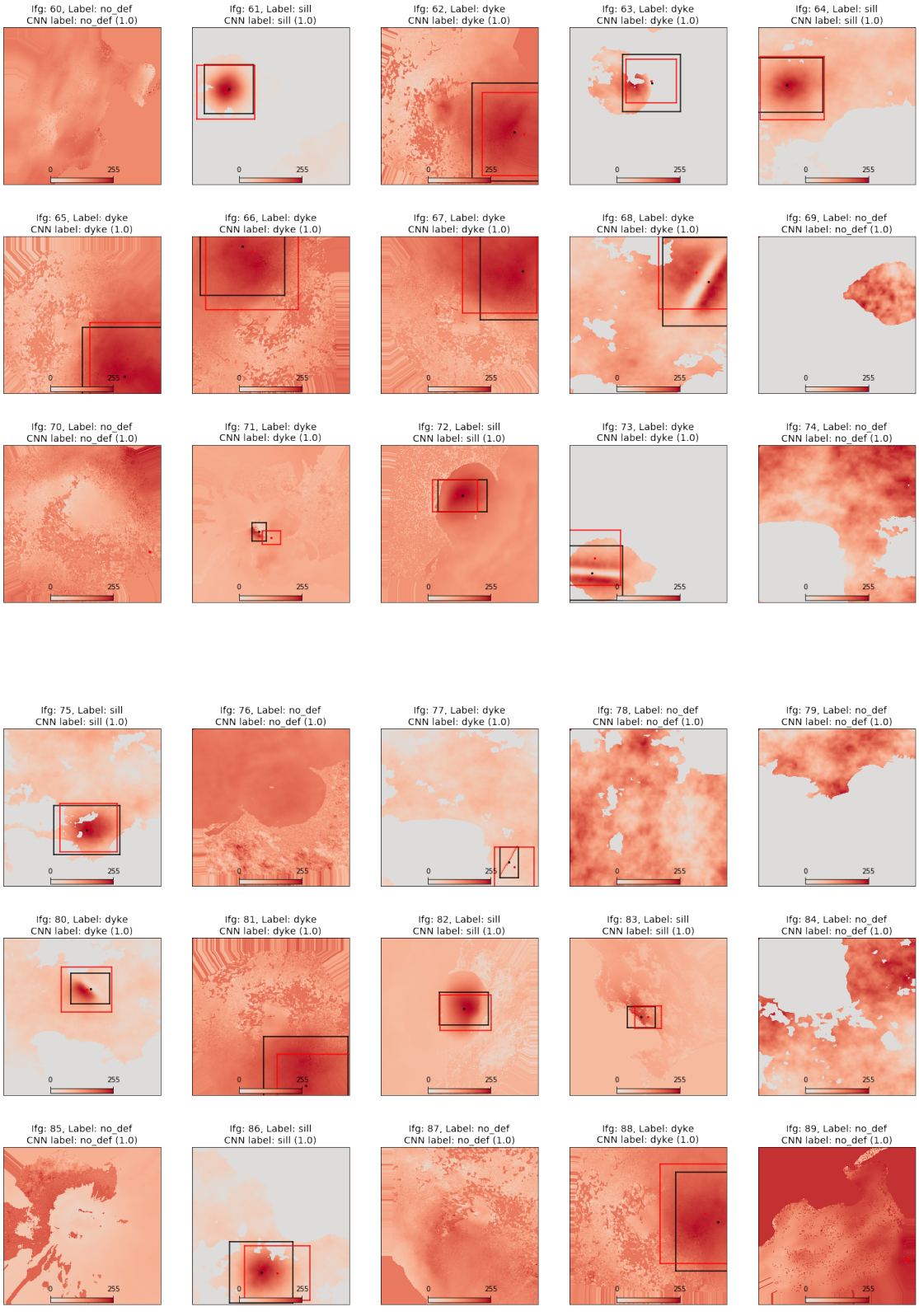
Plot the results of the fine tuned model

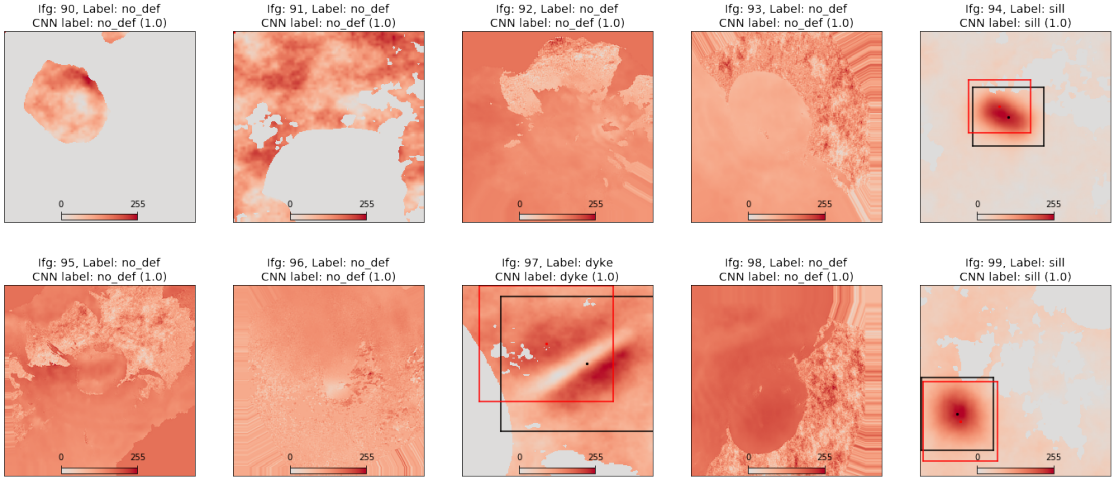
[27]:
```
#%% Test with synthetic and real data

print('\n Forward pass of the testing data through the network:')

Y_class_test_cnn, Y_loc_test_cnn = vgg16_2head.predict(X_test[:,:,:,:], verbose␣
↪= 1)                                    # predict class labels


plot_data_class_loc_caller(X_test, classes = Y_class_test, classes_predicted =␣
↪Y_class_test_cnn,                    # plot all the testing data
                           locs = Y_loc_test, locs_predicted = Y_loc_test_cnn,
                           source_names =␣
↪synthetic_ifgs_settings['defo_sources'],
                           window_title = 'Testing data (after fine tuning)')
```

```
 Forward pass of the testing data through the network:
4/4 [==============================] - 4s 975ms/step
```


lfg: 0, Label: no_def
CNN label: no_def (1.0)
lfg: 1, Label: no_def
CNN label: no_def (1.0)
lfg: 2, Label: no_def
CNN label: no_def (1.0)
lfg: 3, Label: no_def
CNN label: no_def (1.0)
lfg: 4, Label: sill
CNN label: sill (1.0)
lfg: 5, Label: no_def
CNN label: no_def (1.0)
lfg: 6, Label: sill
CNN label: sill (1.0)
lfg: 7, Label: sill
CNN label: sill (1.0)
lfg: 8, Label: sill
CNN label: sill (1.0)
lfg: 9, Label: sill
CNN label: sill (1.0)
lfg: 10, Label: sill
CNN label: sill (1.0)
lfg: 11, Label: no_def
CNN label: no_def (1.0)
lfg: 12, Label: sill
CNN label: sill (1.0)
lfg: 13, Label: dyke
CNN label: dyke (1.0)
lfg: 14, Label: dyke
CNN label: dyke (1.0)
lfg: 15, Label: no_def
CNN label: no_def (1.0)
lfg: 16, Label: dyke
CNN label: dyke (1.0)
lfg: 17, Label: dyke
CNN label: dyke (1.0)
lfg: 18, Label: dyke
CNN label: dyke (1.0)
lfg: 19, Label: no_def
CNN label: no_def (1.0)
lfg: 20, Label: dyke
CNN label: dyke (1.0)
lfg: 21, Label: no_def
CNN label: no_def (1.0)
lfg: 22, Label: no_def
CNN label: no_def (1.0)
lfg: 23, Label: sill
CNN label: sill (1.0)
lfg: 24, Label: sill
CNN label: sill (1.0)
lfg: 25, Label: sill
CNN label: sill (1.0)
lfg: 26, Label: sill
CNN label: sill (1.0)
lfg: 27, Label: no_def
CNN label: no_def (0.96)
lfg: 28, Label: no_def
CNN label: no_def (1.0)
lfg: 29, Label: dyke
CNN label: dyke (1.0)
0
255


Ifg: 30, Label: sill
CNN label: sill (1.0)
Ifg: 31, Label: dyke
CNN label: dyke (1.0)
Ifg: 32, Label: dyke
CNN label: dyke (1.0)
Ifg: 33, Label: dyke
CNN label: dyke (1.0)
Ifg: 34, Label: sill
CNN label: sill (1.0)
Ifg: 35, Label: no_def
CNN label: no_def (1.0)
Ifg: 36, Label: dyke
CNN label: dyke (1.0)
Ifg: 37, Label: sill
CNN label: sill (1.0)
Ifg: 38, Label: no_def
CNN label: no_def (1.0)
Ifg: 39, Label: dyke
CNN label: dyke (1.0)
Ifg: 40, Label: sill
CNN label: sill (1.0)
Ifg: 41, Label: no_def
CNN label: no_def (1.0)
Ifg: 42, Label: sill
CNN label: sill (1.0)
Ifg: 43, Label: no_def
CNN label: no_def (1.0)
Ifg: 44, Label: dyke
CNN label: dyke (1.0)
Ifg: 45, Label: dyke
CNN label: dyke (1.0)
Ifg: 46, Label: sill
CNN label: sill (1.0)
Ifg: 47, Label: dyke
CNN label: dyke (1.0)
Ifg: 48, Label: no_def
CNN label: no_def (1.0)
Ifg: 49, Label: sill
CNN label: sill (1.0)
Ifg: 50, Label: sill
CNN label: sill (1.0)
Ifg: 51, Label: dyke
CNN label: dyke (1.0)
Ifg: 52, Label: no_def
CNN label: dyke (0.66)
Ifg: 53, Label: sill
CNN label: sill (1.0)
Ifg: 54, Label: dyke
CNN label: dyke (1.0)
Ifg: 55, Label: no_def
CNN label: no_def (1.0)
Ifg: 56, Label: dyke
CNN label: dyke (1.0)
Ifg: 57, Label: dyke
CNN label: dyke (1.0)
Ifg: 58, Label: no_def
CNN label: no_def (1.0)
Ifg: 59, Label: sill
CNN label: sill (1.0)
0
255


lfg: 60, Label: no_def
CNN label: no_def (1.0)
lfg: 61, Label: sill
CNN label: sill (1.0)
lfg: 62, Label: dyke
CNN label: dyke (1.0)
lfg: 63, Label: dyke
CNN label: dyke (1.0)
lfg: 64, Label: sill
CNN label: sill (1.0)
lfg: 65, Label: dyke
CNN label: dyke (1.0)
lfg: 66, Label: dyke
CNN label: dyke (1.0)
lfg: 67, Label: dyke
CNN label: dyke (1.0)
lfg: 68, Label: dyke
CNN label: dyke (1.0)
lfg: 69, Label: no_def
CNN label: no_def (1.0)
lfg: 70, Label: no_def
CNN label: no_def (1.0)
lfg: 71, Label: dyke
CNN label: dyke (1.0)
lfg: 72, Label: sill
CNN label: sill (1.0)
lfg: 73, Label: dyke
CNN label: dyke (1.0)
lfg: 74, Label: no_def
CNN label: no_def (1.0)
lfg: 75, Label: sill
CNN label: sill (1.0)
lfg: 76, Label: no_def
CNN label: no_def (1.0)
lfg: 77, Label: dyke
CNN label: dyke (1.0)
lfg: 78, Label: no_def
CNN label: no_def (1.0)
lfg: 79, Label: no_def
CNN label: no_def (1.0)
lfg: 80, Label: dyke
CNN label: dyke (1.0)
lfg: 81, Label: dyke
CNN label: dyke (1.0)
lfg: 82, Label: sill
CNN label: sill (1.0)
lfg: 83, Label: sill
CNN label: sill (1.0)
lfg: 84, Label: no_def
CNN label: no_def (1.0)
lfg: 85, Label: no_def
CNN label: no_def (1.0)
lfg: 86, Label: sill
CNN label: sill (1.0)
lfg: 87, Label: no_def
CNN label: no_def (1.0)
lfg: 88, Label: dyke
CNN label: dyke (1.0)
lfg: 89, Label: no_def
CNN label: no_def (1.0)
0
255


Ifg: 90, Label: no_def
CNN label: no_def (1.0)
Ifg: 91, Label: no_def
CNN label: no_def (1.0)
Ifg: 92, Label: no_def
CNN label: no_def (1.0)
Ifg: 93, Label: no_def
CNN label: no_def (1.0)
Ifg: 94, Label: sill
CNN label: sill (1.0)
Ifg: 95, Label: no_def
CNN label: no_def (1.0)
Ifg: 96, Label: no_def
CNN label: no_def (1.0)
Ifg: 97, Label: dyke
CNN label: dyke (1.0)
Ifg: 98, Label: no_def
CNN label: no_def (1.0)
Ifg: 99, Label: sill
CNN label: sill (1.0)
0
255

[ ]:

[ ]:

[ ]: