In [1]:

```
import pandas as pd
import ipywidgets as widgets
from ipywidgets import interact

import utils

print('All packages imported successfully!')
```

```
All packages imported successfully!
```

```
metadata = utils.get_dataframe_from_file_structure()

# Display the first five rows of the dataset
metadata.head()
```

	subset	label	lat	lon	path	filename
0	train	visible_damage	29.771164000000002	-95.638338	train/visible_damage/-95.638338_29.77116400000...	-95.638338_29.771164000000002.jpeg
1	train	visible_damage	29.824802000000002	-95.089349	train/visible_damage/-95.089349_29.82480200000...	-95.089349_29.824802000000002.jpeg
2	train	visible_damage	29.981403000000004	-95.119694	train/visible_damage/-95.119694_29.98140300000...	-95.119694_29.981403000000004.jpeg
3	train	visible_damage	29.757379999999998	-95.59024000000001	train/visible_damage/-95.59024000000001_29.757...	-95.59024000000001_29.757379999999998.jpeg
4	train	visible_damage	28.579998	-96.994213	train/visible_damage/-96.994213_28.579998.jpeg	-96.994213_28.579998.jpeg

```
# Create dataframe that summarizes image classification in each subset
df = pd.pivot_table(metadata, index='subset', columns='label', values='filename', aggfunc='count')

# Add new column with total number of images in each subset
df['total'] = df['visible_damage'] + df['no_damage']

# Show dataframe
df
```

label	no_damage	visible_damage	total
subset			
test	1000	1000	2000
train	5000	5000	10000
validation	1000	1000	2000

```
dataset = 'train'
# Match images in the no damage and damage dataset based on the location coordinates in the file name
matches = list(set(metadata.query(f'subset == "{dataset}" & label == "visible_damage"')['filename'])
               .intersection(metadata.query(f'subset == "{dataset}" & label == "no_damage"')['filename']))

# Load index slider to navigate between the paired images
file_index_widget = widgets.IntSlider(min=0, max=len(matches)-1, value=10, description='Image Num')

# Load visualizer to match paired images
interact(utils.interactive_plot_pair(f'./data/{dataset}/', matches), file_index=file_index_widget);
```

Image Num 1255

No damage

Visible damage

```
import os
import logging
import tensorflow as tf

# Configure Python to ignore Tensorflow warnings
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'    # Ignore tf warning messages
logging.getLogger("tensorflow").setLevel(logging.ERROR)
tf.autograph.set_verbosity(0)

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam

import matplotlib.pyplot as plt

import utils

print('All packages imported successfully!')
IMAGE_SIZE = (150, 150)
```

```
All packages imported successfully!
```

```
utils.data_aug_flip(image)
plt.show()
```

Random Flip
- ○ horizontal
- ◉ vertical
- ○ horizontal_and_vertical

Original

Flipped

```
utils.data_aug_zoom(image)
plt.show()
```

Zoom: 1.2

Original

Zoomed

```
utils.data_aug_rot(image)
plt.show()
```

Rotation (deg): 20.0

Original

Rotated

```
utils.data_aug_brightness(image)
plt.show()
```

Brightness factor: 1.3

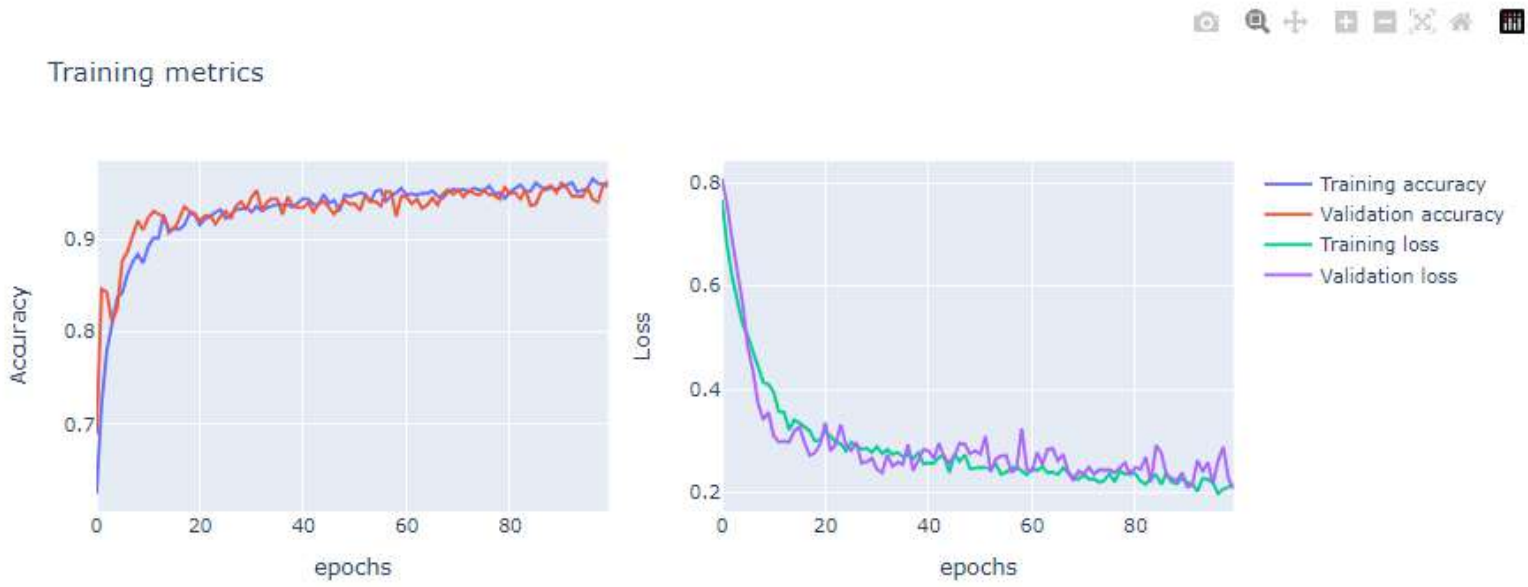
Original

Brightness

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_16 (Conv2D)          (None, 148, 148, 32)      896

 max_pooling2d_16 (MaxPoolin  (None, 74, 74, 32)       0
 g2D)

 batch_normalization_4 (Batc  (None, 74, 74, 32)       128
 hNormalization)

 conv2d_17 (Conv2D)          (None, 72, 72, 64)        18496

 max_pooling2d_17 (MaxPoolin  (None, 36, 36, 64)       0
 g2D)

 conv2d_18 (Conv2D)          (None, 34, 34, 128)       73856
```

```
utils.plot_training_history('./models/cnn_history')
```



```python
# Code to train the model for three more epochs (this can run for a few minutes)
epochs = 3
history_finetune = model.fit(
    train_generator,
    epochs=epochs
)
```

```
Epoch 1/3
8/8 [==============================] - 51s 6s/step - loss: 0.1825 - accuracy: 0.9730
Epoch 2/3
8/8 [==============================] - 48s 6s/step - loss: 0.1823 - accuracy: 0.9780
Epoch 3/3
8/8 [==============================] - 49s 6s/step - loss: 0.1770 - accuracy: 0.9800
```

```
utils.display_confusion_matrix(y_labels, y_predict_prob_1)
```