In [ ]:

```

```

In [2]:

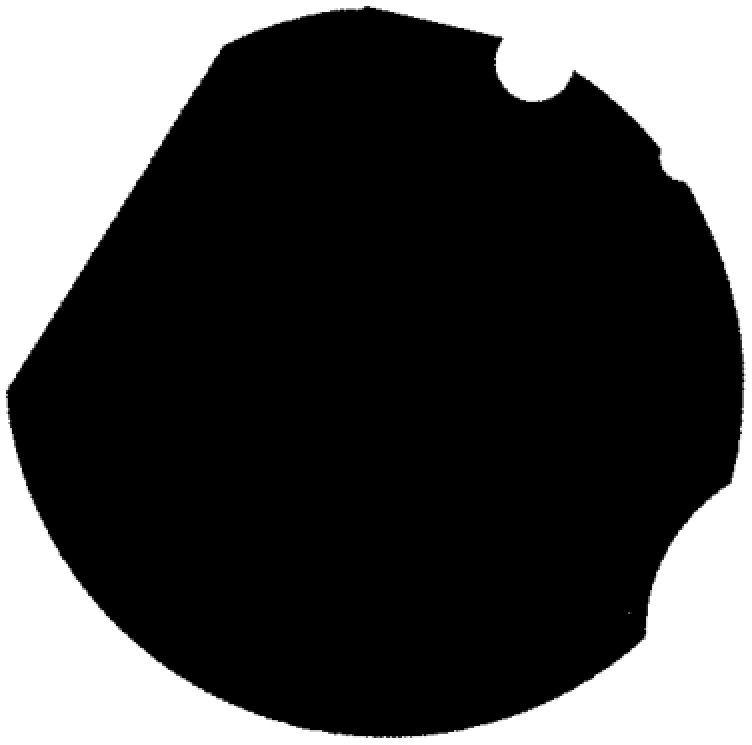
```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import glob
import matplotlib.pyplot as plt
from PIL import Image
import random
import os
import numpy as np
from matplotlib import cm
from  matplotlib.colors import LinearSegmentedColormap
import seaborn as sns
from keras_preprocessing.image import load_img
from keras_preprocessing.image import img_to_array
from keras_preprocessing.image import array_to_img
from scipy.stats import norm
import pandas as pd
from tensorflow.keras import datasets, layers, models
```

In [3]:

```python
# load the mask image
maskim = Image.open('./mask.png')
maskdata = np.array(maskim, dtype='i2')

plt.figure()
plt.axis('off')
plt.imshow(maskim, cmap='gray_r')
```

Out[3]:

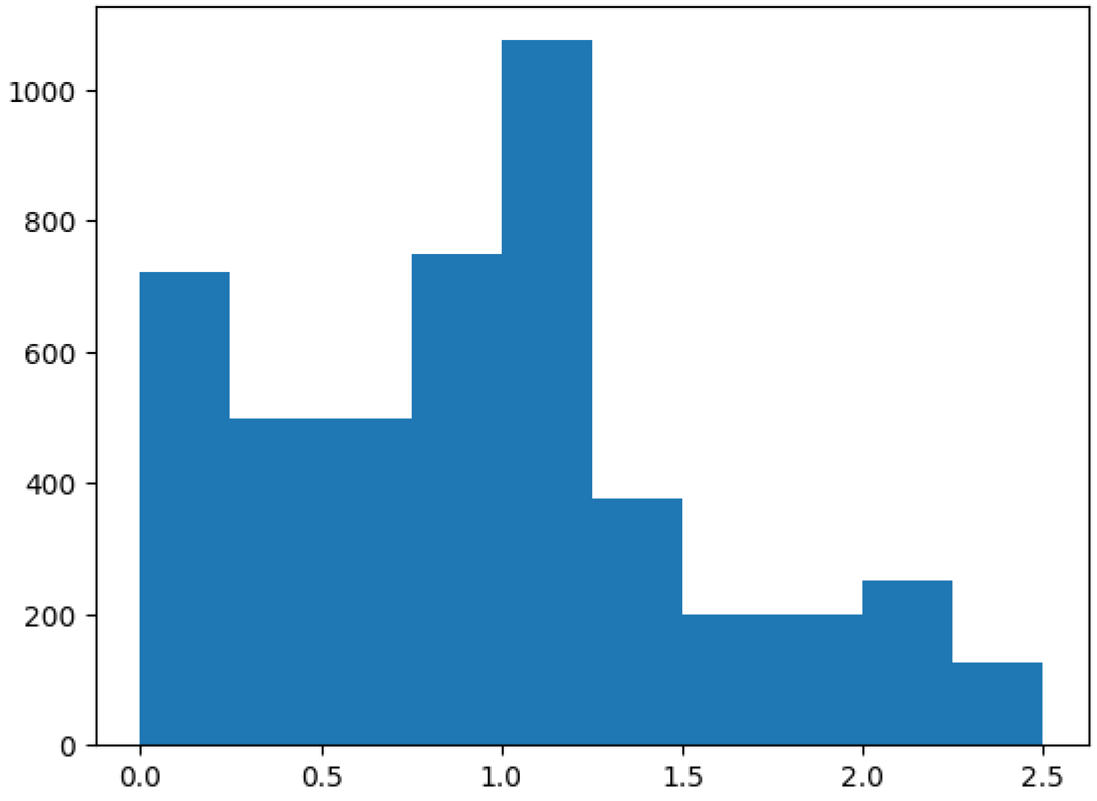
```
<matplotlib.image.AxesImage at 0x14daabc88>
```



```
In [4]: directory = "../kernels/"
        df = pd.read_csv(directory + "attenuations.csv")
        KW = 12
        KH = 12
        batch_size=64#len(df)
        df_size = len(df)
        image_size = (KW,KH)

        i = 0
        images = []
        attenuations = []
        for index, row in df.iterrows():
            filename = directory + row[2]
            attenuation = row[1]
            if(attenuation < 0):
                attenuation = 0
            if(attenuation > 2.5):
                attenuation = 2.5
            if(attenuation < 0.2):
                i += 1
            #if(i % 20 == 1 or attenuation > 0.2):
            attenuations.append(attenuation)
            img = load_img(filename, target_size=image_size)
            img_array = img_to_array(img)
            images.append(img_array)

        DF = pd.DataFrame(pd.Series(images))
        DF.columns = ["image"]
        DF["attenuation"] = attenuations

        plt.hist(DF["attenuation"])
        plt.savefig('attenuation_distibution.png')
        print(len(images))


        dataset = tf.data.Dataset.from_tensor_slices((list(DF['image'].values), DF['

        test_split = int(df_size * 0.2)
        #dataset.shuffle(batch_size)
        test_dataset = dataset.take(test_split)
        train_dataset = dataset.skip(test_split).take(df_size-test_split)
        test_dataset = test_dataset.batch(test_split).cache().prefetch(buffer_size=t
        train_dataset = train_dataset.batch(df_size - test_split).cache().prefetch(b
```

```
4698
```

```
2023-11-16 12:41:43.014607: I tensorflow/compiler/jit/xla_cpu_device.cc:41] N
ot creating XLA devices, tf_xla_enable_xla_devices not set
2023-11-16 12:41:43.016955: I tensorflow/core/platform/cpu_feature_guard.cc:1
42] This TensorFlow binary is optimized with oneAPI Deep Neural Network Libra
ry (oneDNN) to use the following CPU instructions in performance-critical ope
rations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate c
ompiler flags.
```



In [ ]:

In [5]:
```
model = models.Sequential()
model.add(layers.Conv2D(12, (3, 3), activation='relu', strides=2, padding="s
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', strides=2, padding="s
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', strides=2, padding="s
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(units=1))
```

In [ ]:

In [6]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 6, 6, 12)          336

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 3, 3, 12)         0

_________________________________________________________________
conv2d_1 (Conv2D)           (None, 2, 2, 64)          6976

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 1, 1, 64)         0

_________________________________________________________________
conv2d_2 (Conv2D)           (None, 1, 1, 64)          36928

_________________________________________________________________
flatten (Flatten)           (None, 64)                0

_________________________________________________________________
dense (Dense)               (None, 128)               8320

_________________________________________________________________
dense_1 (Dense)             (None, 64)                8256

_________________________________________________________________
dense_2 (Dense)             (None, 32)                2080

_________________________________________________________________
dense_3 (Dense)             (None, 1)                 33
=================================================================
Total params: 62,929
Trainable params: 62,929
Non-trainable params: 0

_________________________________________________________________
```

In [7]:
```python
epochs = 100

model.compile(
    optimizer=keras.optimizers.Adam(1e-4),
    loss='mean_squared_error'
)
```

In [8]:
```python
history = model.fit(
    train_dataset,
    epochs=epochs,
)
```

```
Epoch 1/100
2023-11-16 12:41:44.615674: I tensorflow/compiler/mlir/mlir_graph_optimizatio
n_pass.cc:116] None of the MLIR optimization passes are enabled (registered
2)
```

```
1/1 [==============================] - 1s 717ms/step - loss: 1.0756
Epoch 2/100
1/1 [==============================] - 0s 72ms/step - loss: 0.6159
Epoch 3/100
1/1 [==============================] - 0s 72ms/step - loss: 0.6578
Epoch 4/100
1/1 [==============================] - 0s 84ms/step - loss: 0.7188
Epoch 5/100
1/1 [==============================] - 0s 79ms/step - loss: 0.6493
Epoch 6/100
1/1 [==============================] - 0s 77ms/step - loss: 0.5349
Epoch 7/100
1/1 [==============================] - 0s 87ms/step - loss: 0.4515
Epoch 8/100
1/1 [==============================] - 0s 83ms/step - loss: 0.4257
Epoch 9/100
1/1 [==============================] - 0s 75ms/step - loss: 0.4400
Epoch 10/100
1/1 [==============================] - 0s 80ms/step - loss: 0.4577
Epoch 11/100
1/1 [==============================] - 0s 81ms/step - loss: 0.4536
Epoch 12/100
1/1 [==============================] - 0s 83ms/step - loss: 0.4256
Epoch 13/100
1/1 [==============================] - 0s 80ms/step - loss: 0.3867
Epoch 14/100
1/1 [==============================] - 0s 77ms/step - loss: 0.3518
Epoch 15/100
1/1 [==============================] - 0s 77ms/step - loss: 0.3311
Epoch 16/100
1/1 [==============================] - 0s 83ms/step - loss: 0.3250
Epoch 17/100
1/1 [==============================] - 0s 87ms/step - loss: 0.3266
Epoch 18/100
1/1 [==============================] - 0s 94ms/step - loss: 0.3265
Epoch 19/100
1/1 [==============================] - 0s 71ms/step - loss: 0.3194
Epoch 20/100
1/1 [==============================] - 0s 83ms/step - loss: 0.3055
Epoch 21/100
1/1 [==============================] - 0s 83ms/step - loss: 0.2895
Epoch 22/100
1/1 [==============================] - 0s 93ms/step - loss: 0.2767
Epoch 23/100
1/1 [==============================] - 0s 91ms/step - loss: 0.2702
Epoch 24/100
1/1 [==============================] - 0s 76ms/step - loss: 0.2690
Epoch 25/100
1/1 [==============================] - 0s 81ms/step - loss: 0.2695
Epoch 26/100
1/1 [==============================] - 0s 89ms/step - loss: 0.2678
Epoch 27/100
1/1 [==============================] - 0s 88ms/step - loss: 0.2621
Epoch 28/100
1/1 [==============================] - 0s 87ms/step - loss: 0.2535
Epoch 29/100
```

```
1/1 [==============================] - 0s 84ms/step - loss: 0.2447
Epoch 30/100
1/1 [==============================] - 0s 93ms/step - loss: 0.2383
Epoch 31/100
1/1 [==============================] - 0s 78ms/step - loss: 0.2351
Epoch 32/100
1/1 [==============================] - 0s 86ms/step - loss: 0.2341
Epoch 33/100
1/1 [==============================] - 0s 79ms/step - loss: 0.2331
Epoch 34/100
1/1 [==============================] - 0s 78ms/step - loss: 0.2307
Epoch 35/100
1/1 [==============================] - 0s 83ms/step - loss: 0.2266
Epoch 36/100
1/1 [==============================] - 0s 90ms/step - loss: 0.2219
Epoch 37/100
1/1 [==============================] - 0s 86ms/step - loss: 0.2178
Epoch 38/100
1/1 [==============================] - 0s 90ms/step - loss: 0.2150
Epoch 39/100
1/1 [==============================] - 0s 82ms/step - loss: 0.2134
Epoch 40/100
1/1 [==============================] - 0s 85ms/step - loss: 0.2124
Epoch 41/100
1/1 [==============================] - 0s 83ms/step - loss: 0.2112
Epoch 42/100
1/1 [==============================] - 0s 88ms/step - loss: 0.2094
Epoch 43/100
1/1 [==============================] - 0s 87ms/step - loss: 0.2070
Epoch 44/100
1/1 [==============================] - 0s 81ms/step - loss: 0.2047
Epoch 45/100
1/1 [==============================] - 0s 82ms/step - loss: 0.2028
Epoch 46/100
1/1 [==============================] - 0s 80ms/step - loss: 0.2015
Epoch 47/100
1/1 [==============================] - 0s 88ms/step - loss: 0.2006
Epoch 48/100
1/1 [==============================] - 0s 90ms/step - loss: 0.2000
Epoch 49/100
1/1 [==============================] - 0s 88ms/step - loss: 0.1992
Epoch 50/100
1/1 [==============================] - 0s 87ms/step - loss: 0.1980
Epoch 51/100
1/1 [==============================] - 0s 86ms/step - loss: 0.1965
Epoch 52/100
1/1 [==============================] - 0s 88ms/step - loss: 0.1951
Epoch 53/100
1/1 [==============================] - 0s 78ms/step - loss: 0.1938
Epoch 54/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1929
Epoch 55/100
1/1 [==============================] - 0s 84ms/step - loss: 0.1921
Epoch 56/100
1/1 [==============================] - 0s 75ms/step - loss: 0.1914
Epoch 57/100
```

```
1/1 [==============================] - 0s 83ms/step - loss: 0.1905
Epoch 58/100
1/1 [==============================] - 0s 77ms/step - loss: 0.1893
Epoch 59/100
1/1 [==============================] - 0s 81ms/step - loss: 0.1882
Epoch 60/100
1/1 [==============================] - 0s 74ms/step - loss: 0.1871
Epoch 61/100
1/1 [==============================] - 0s 78ms/step - loss: 0.1863
Epoch 62/100
1/1 [==============================] - 0s 72ms/step - loss: 0.1856
Epoch 63/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1850
Epoch 64/100
1/1 [==============================] - 0s 74ms/step - loss: 0.1843
Epoch 65/100
1/1 [==============================] - 0s 74ms/step - loss: 0.1835
Epoch 66/100
1/1 [==============================] - 0s 78ms/step - loss: 0.1827
Epoch 67/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1820
Epoch 68/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1814
Epoch 69/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1808
Epoch 70/100
1/1 [==============================] - 0s 77ms/step - loss: 0.1803
Epoch 71/100
1/1 [==============================] - 0s 82ms/step - loss: 0.1797
Epoch 72/100
1/1 [==============================] - 0s 81ms/step - loss: 0.1790
Epoch 73/100
1/1 [==============================] - 0s 82ms/step - loss: 0.1783
Epoch 74/100
1/1 [==============================] - 0s 71ms/step - loss: 0.1777
Epoch 75/100
1/1 [==============================] - 0s 72ms/step - loss: 0.1772
Epoch 76/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1766
Epoch 77/100
1/1 [==============================] - 0s 72ms/step - loss: 0.1761
Epoch 78/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1755
Epoch 79/100
1/1 [==============================] - 0s 99ms/step - loss: 0.1748
Epoch 80/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1742
Epoch 81/100
1/1 [==============================] - 0s 77ms/step - loss: 0.1737
Epoch 82/100
1/1 [==============================] - 0s 79ms/step - loss: 0.1731
Epoch 83/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1726
Epoch 84/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1721
Epoch 85/100
```

```
1/1 [==============================] - 0s 78ms/step - loss: 0.1715
Epoch 86/100
1/1 [==============================] - 0s 77ms/step - loss: 0.1710
Epoch 87/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1705
Epoch 88/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1699
Epoch 89/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1695
Epoch 90/100
1/1 [==============================] - 0s 75ms/step - loss: 0.1690
Epoch 91/100
1/1 [==============================] - 0s 74ms/step - loss: 0.1685
Epoch 92/100
1/1 [==============================] - 0s 78ms/step - loss: 0.1681
Epoch 93/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1676
Epoch 94/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1672
Epoch 95/100
1/1 [==============================] - 0s 72ms/step - loss: 0.1668
Epoch 96/100
1/1 [==============================] - 0s 72ms/step - loss: 0.1664
Epoch 97/100
1/1 [==============================] - 0s 75ms/step - loss: 0.1659
Epoch 98/100
1/1 [==============================] - 0s 73ms/step - loss: 0.1655
Epoch 99/100
1/1 [==============================] - 0s 70ms/step - loss: 0.1651
Epoch 100/100
1/1 [==============================] - 0s 76ms/step - loss: 0.1647
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

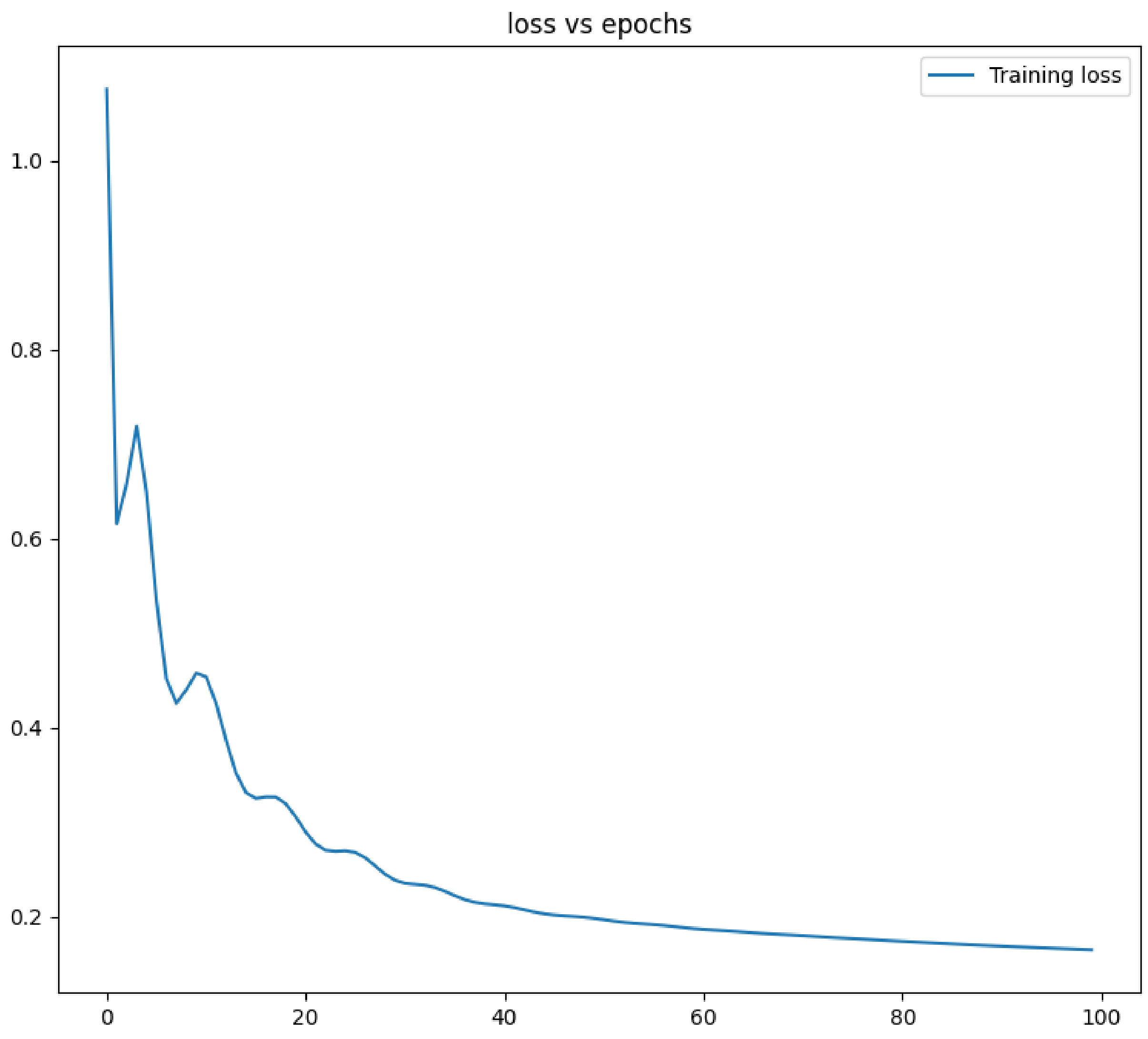
In [ ]:

In [9]:
```
#save model
model.save('./model/')
```

```
2023-11-16 12:41:54.022530: W tensorflow/python/util/util.cc:348] Sets are no
t currently considered sequences, but this may change in the future, so consi
der avoiding using them.
INFO:tensorflow:Assets written to: ./model/assets
```

```
In [10]: #load a previous model:
         model = tf.keras.models.load_model('./model/')
```

```
In [11]: #model performance visualization
         f = plt.figure(figsize=(20,8))

         #loss
         plt2 = f.add_subplot(122)
         plt2.plot(history.history['loss'], label = str('Training loss'))
         plt.legend()
         plt.title('loss vs epochs')

         plt.show()
```



```
In [12]: model.evaluate(test_dataset)

        1/1 [==============================] - 0s 123ms/step - loss: 1.3559

Out[12]: 1.3559308052062988
```

```
In [ ]:
```

In [ ]:

```

```

In [13]:

```
cmap = plt.colormaps['jet'].copy()
cmap.set_bad('white')       # color of mask on heatmap
cmap.set_under('white')     # color of mask on cbar
```

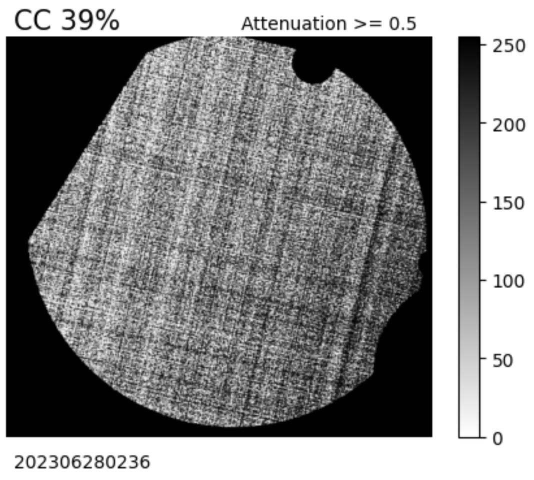
In [14]:

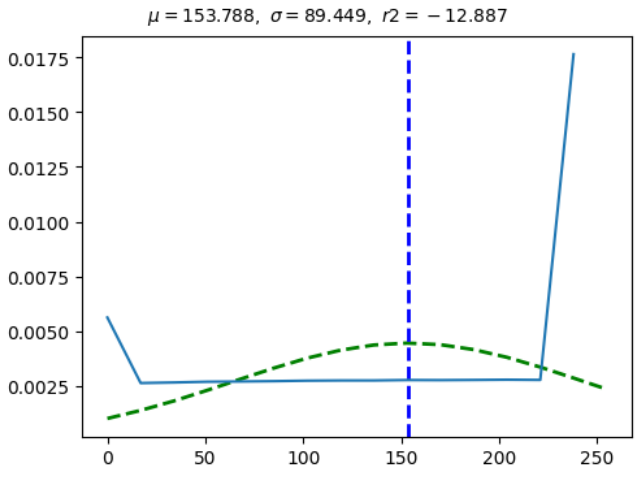
```
IW = 512
IH = 512

size = KW / 2 #side length of kernel from center. 16x16 for size=8
maxSqrSize = 145 #length from center to edge
midx = IW/2
midy = IH/2
```

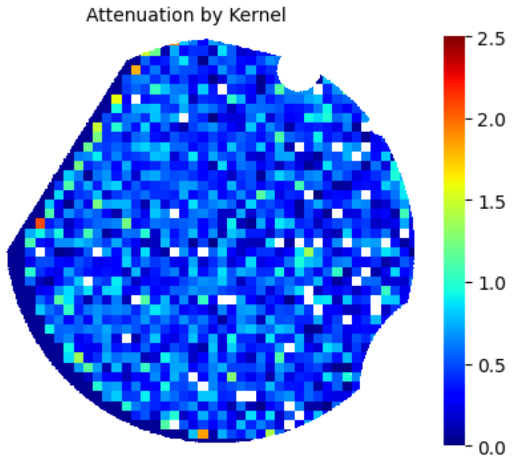
In [15]:

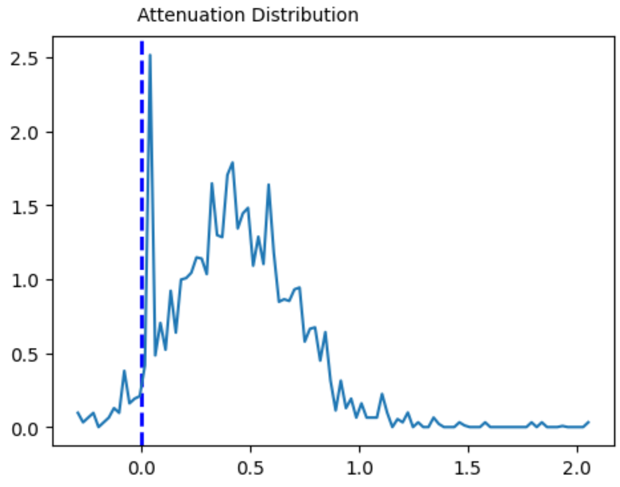
```
#load image

# make sure to check images from the mix,clear, and cloudy folders
folder="../images/"
i = 0
for file in glob.glob(folder + '/*.png'):

# how many tests do you want to look at
#for i in range(1):

#    while True:
#        a=random.choice(os.listdir(folder))
#        if "20200616_2253" in a:
#        if "20200619_2023" in a:
#        if "20230726_2242" in a:
#        if "20230718_0331" in a:
#        if "20230322_2048" in a:
#        if "20220927_0324" in a:
#        if "20220625_2352" in a:
#        if "20220501_0044" in a:
#        if "20220420_2102" in a:
#            break
#    a=random.choice(os.listdir(folder))
#    print (a)

#    file = folder+'/'+a
#    print(file)

#im = Image.open("./train/cloudy/202302042354a.png")

    filedate = file.split('/')[2].split('asiva')[1].split('.')[-2]

#    print (filedate, filetime)
    im = Image.open(file)

    imc = im
    x_locs = np.arange(size, IW, size * 2)
    y_locs = np.arange(size, IH, size * 2)

    heatmap = np.empty([IH, IW])
    heatmap_masked = []

    for y in y_locs:
        for x in x_locs:
            imk = imc.crop((x-size, y-size, x+size, y+size))
            imk.save("tmp.png")
            img = load_img("tmp.png", target_size=image_size)
            img_array = img_to_array(img)
            img_array = tf.expand_dims(img_array, 0)  # Create batch axis

            predictions = model.predict(img_array, verbose = 0)
            cloudy_score = predictions[0]

            size = int(size)
            y = int(y)
            x = int(x)

            for k_y in range(y-size, y+size):
                for k_x in range(x-size, x+size):
                    if (k_y >= IH):
                        continue
                    if (k_x >= IW):
                        continue
                    if (cloudy_score <= 0):
                        heatmap[k_y, k_x] = 0
                    if (cloudy_score >=2.5):
                        heatmap[k_y, k_x] = 2.5

                    if (maskdata[k_y, k_x] != 1):
                        heatmap[k_y, k_x] = cloudy_score
                        heatmap_masked.append(cloudy_score)
                    else:
                        heatmap[k_y, k_x] = 0 # was -1

    heat = heatmap#np.uint8(heatmap)

#calculate percentage cloud coverage. Iterate through image
#and for a given threshold value set value above to 1 and values below to 0

    heatmap_filtered = np.ma.array(heat, mask=maskdata > 0.0)

    imarr = np.array(im)

    data = []
    hdata = []

    th = 0.5
    cloud_count = 0
    total = 0

    for i in range(len(heatmap_filtered)):
        for j in range(len(heatmap_filtered[0])):
            if(maskdata[i, j] < 1):
                total = total + 1
                data.append(imarr[i,j])
                hdata.append(heatmap_filtered[i,j])
                if(heatmap_filtered[i, j] > (th)):
                    cloud_count = cloud_count + 1
    #             else:
    #                 print (cloud_count, i,j,heatmap_filtered[i,j])
    print(max(hdata))
    #stats
    mu = np.average(data)
    sigma = np.std(data)

    # normalized distribution
    histogram, bins = np.histogram(data, bins=15, density=True)

    # gaussian
    y = norm.pdf(bins, mu, sigma)

    # truncate last element to match up arrays
    y0 = np.delete(y, -1)

    # fundamental stats
    absError = histogram - y0
    SE = np.square(absError) # squared errors
    MSE = np.mean(SE) # mean squared errors
    RMSE = np.sqrt(MSE) # Root Mean Squared Error, RMSE
    Rsquared = 1.0 - (np.var(absError) / np.var(y0))

    fig = plt.figure(figsize=(12, 4))

    ax = fig.add_subplot(121)

    ia = ax.imshow(im, cmap='gray_r')
    plt.colorbar(ia)

    ax.text(10, -10, 'CC ' + str(int(cloud_count / total * 100)) + '%', font
    ax.text(300, -10, 'Attenuation >= ' + str((th)))
    ax.text(10, 550, filedate)
    ax.axis('off')
    plt.axis('off')

    ax1 = fig.add_subplot(122)
    ax1.plot(bins, y, 'g', linewidth=2, linestyle='--')
    ax1.axvline(mu, color='b', linewidth=2, linestyle='--')
    ax.text(900,-20, r'$\ \mu=%.3f,\ \sigma=%.3f,\ r2=%.3f$' % (mu, sigma, R
    ax1.plot(bins[0:-1], histogram)

    #stats
    mu = np.average(hdata)
    sigma = np.std(hdata)

    # normalized distribution
    histogram, bins = np.histogram(hdata, bins=100, density=True)

    # gaussian
    y = norm.pdf(bins, mu, sigma)

    # truncate last element to match up arrays
    y0 = np.delete(y, -1)

    # fundamental stats
    absError = histogram - y0
    SE = np.square(absError) # squared errors
    MSE = np.mean(SE) # mean squared errors
    RMSE = np.sqrt(MSE) # Root Mean Squared Error, RMSE
    Rsquared = 1.0 - (np.var(absError) / np.var(y0))

    fig.savefig('./img/' + filedate + '_0.png')

    fig2 = plt.figure(figsize=(12, 4))

    ax2 = fig2.add_subplot(121)
    ax2.text(100,-20, 'Attenuation by Kernel')

    hm = sns.heatmap(heat, cmap=cmap, vmin=0.0, vmax=2.5, cbar=True, annot=F

    plt.axis('off')
    ax3 = fig2.add_subplot(122)
    ax3.axvline(int(th), color='b', linewidth=2, linestyle='--')
    ax2.text(900,-20, 'Attenuation Distribution')
    ax3.plot(bins[0:100:1], histogram)

    #save probability values
    f = open("./model/" + "/probabilities.txt", "w")
    for item in hdata:
        f.write(str(item) + "\n")
    f.close()

    plt.show()

    fig2.savefig('./img' + filedate + '_1.png')

    #imw = Image.open('./img/' + filedate + '_d.png')

    fig3 = plt.figure(figsize=(12, 4))

    fx1 = fig3.add_subplot(121)
    ia = fx1.imshow(im, cmap='gray_r')
    hm = sns.heatmap(heat, cmap=cmap, vmin=0.0, vmax=2.5, cbar=False, annot=

    fx1.axis('off')


    fx2 = fig3.add_subplot(122)
    fx2.axis('off')
    #fx2.imshow(imw)

    plt.axis('off')

    fig3.savefig('./img/' + filedate + '_3.png')

    print (filedate, int(cloud_count / total * 100))
```

```
2.078126907348633
```

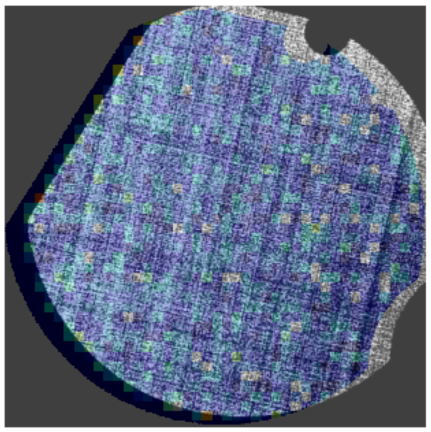


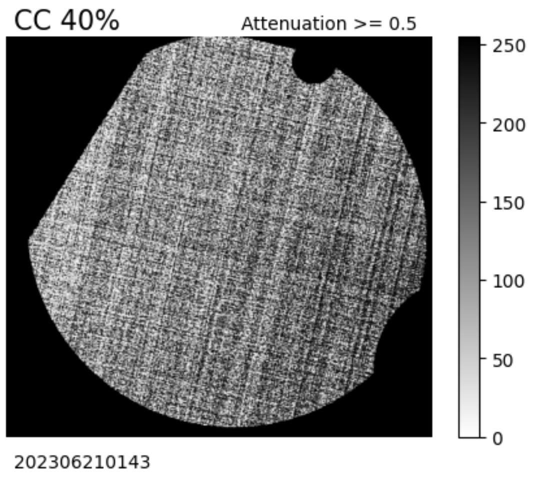


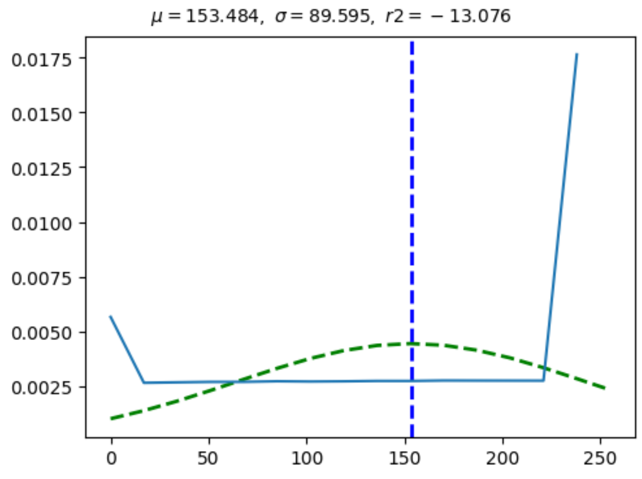


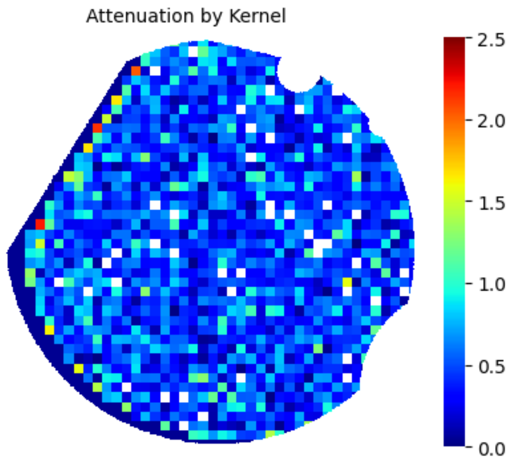


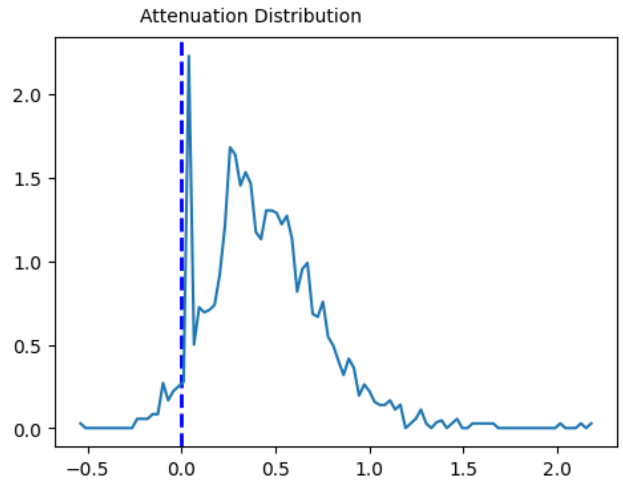
```
202306280236 39
2.20774245262146
```



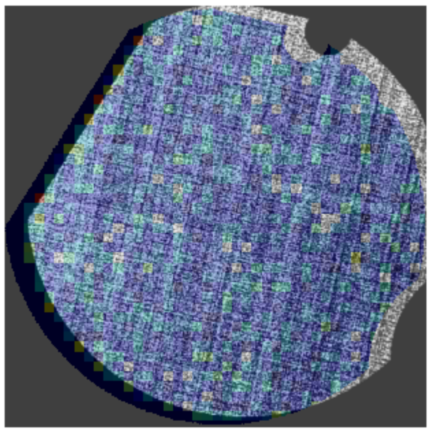


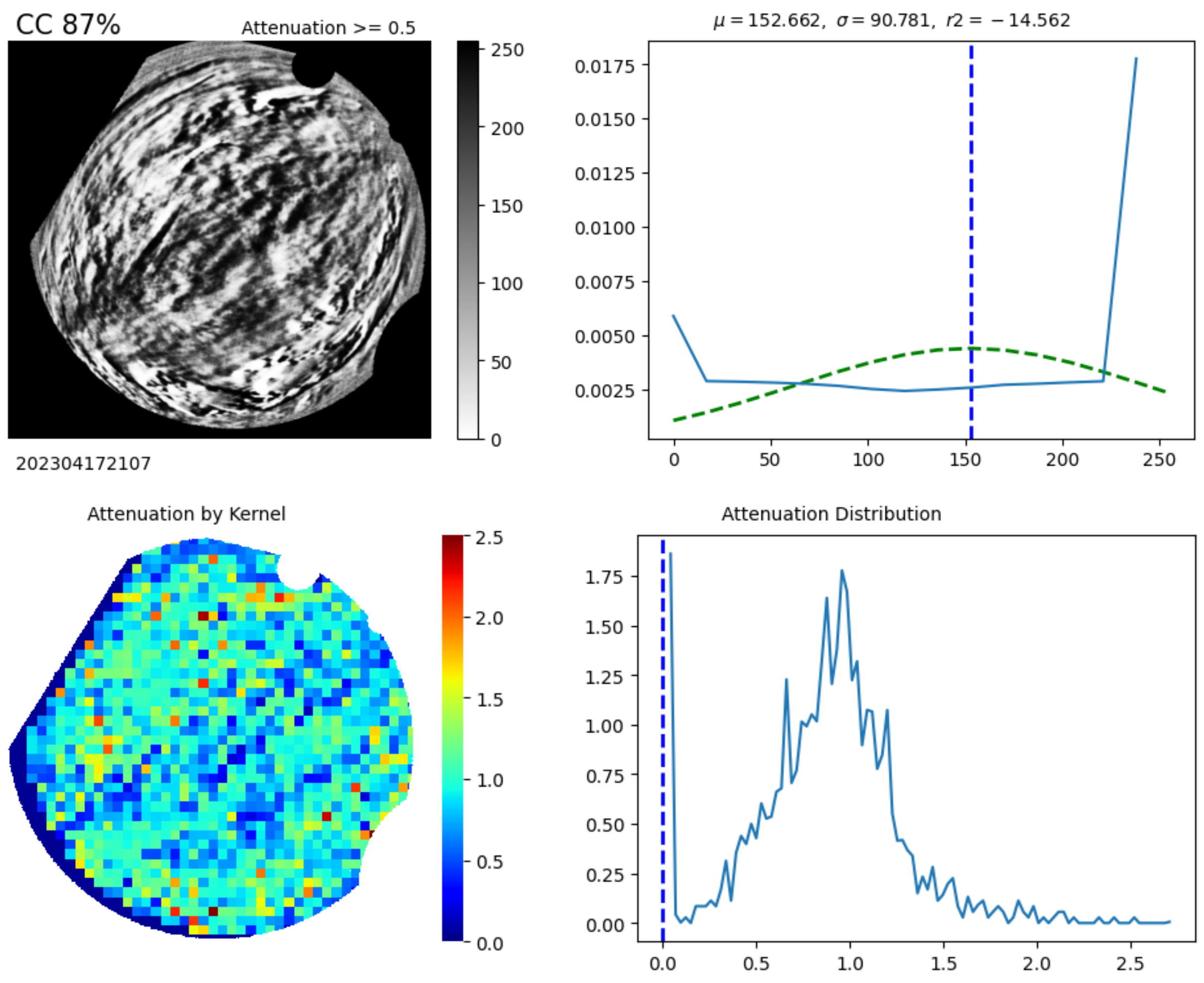






```
202306210143 40
2.7366840839385986
```





```
202304172107 87
```

```
Exception ignored in: <function IteratorResourceDeleter.__del__ at 0x113c03e1
8>
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-
packages/tensorflow/python/data/ops/iterator_ops.py", line 532, in __del__
    if self._eager_mode:
KeyboardInterrupt
```

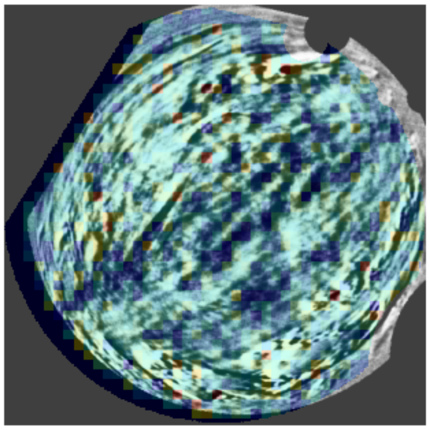
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/ops/array_ops.py in gather(***failed resolving arguments*
**)
   4812     # without introducing a circular dependency.
-> 4813     return params.sparse_read(indices, name=name)
   4814   except AttributeError:

AttributeError: 'Tensor' object has no attribute 'sparse_read'

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
/var/folders/1f/q2kb4sw53m7_smq0287qp_0r0000gn/T/ipykernel_9809/1450839862.py
in <module>
     50             img_array = tf.expand_dims(img_array, 0)  # Create batch
axis
     51 
---> 52             predictions = model.predict(img_array, verbose = 0)
     53             cloudy_score = predictions[0]
     54 

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/keras/engine/training.py in predict(self, x, batch_size,
verbose, steps, callbacks, max_queue_size, workers, use_multiprocessing)
   1606           use_multiprocessing=use_multiprocessing,
   1607           model=self,
-> 1608           steps_per_execution=self._steps_per_execution)
   1609 
   1610       # Container that configures and calls `tf.keras.Callback`s.

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/keras/engine/data_adapter.py in __init__(self, x, y, samp
le_weight, batch_size, steps_per_epoch, initial_epoch, epochs, shuffle, class
_weight, max_queue_size, workers, use_multiprocessing, model, steps_per_execu
tion)
   1110         use_multiprocessing=use_multiprocessing,
   1111         distribution_strategy=ds_context.get_strategy(),
-> 1112         model=model)
   1113 
   1114     strategy = ds_context.get_strategy()

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/keras/engine/data_adapter.py in __init__(self, x, y, samp
le_weights, sample_weight_modes, batch_size, epochs, steps, shuffle, **kwarg
s)
    353     indices_dataset = indices_dataset.flat_map(slice_batch_indices)
    354 
--> 355     dataset = self.slice_inputs(indices_dataset, inputs)
    356 
    357     if shuffle == "batch":

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/keras/engine/data_adapter.py in slice_inputs(self, indice
s_dataset, inputs)
```

```
    386 
    387     dataset = dataset.map(
--> 388         grab_batch, num_parallel_calls=dataset_ops.AUTOTUNE)
    389 
    390     # Default optimizations are disabled to avoid the overhead of (un
necessary)

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/data/ops/dataset_ops.py in map(self, map_func, num_parall
el_calls, deterministic)
   1810           num_parallel_calls,
   1811           deterministic,
-> 1812           preserve_cardinality=True)
   1813 
   1814   def flat_map(self, map_func):

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/data/ops/dataset_ops.py in __init__(self, input_dataset,
map_func, num_parallel_calls, deterministic, use_inter_op_parallelism, preser
ve_cardinality, use_legacy_function)
   4244         self._transformation_name(),
   4245         dataset=input_dataset,
-> 4246         use_legacy_function=use_legacy_function)
   4247     if deterministic is None:
   4248       self._deterministic = "default"

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/data/ops/dataset_ops.py in __init__(self, func, transform
ation_name, dataset, input_classes, input_shapes, input_types, input_structur
e, add_to_graph, use_legacy_function, defun_kwargs)
   3523       with tracking.resource_tracker_scope(resource_tracker):
   3524         # TODO(b/141462134): Switch to using garbage collection.
-> 3525         self._function = wrapper_fn.get_concrete_function()
   3526         if add_to_graph:
   3527           self._function.add_to_graph(ops.get_default_graph())

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/eager/function.py in get_concrete_function(self, *args, *
*kwargs)
   3050     """
   3051     graph_function = self._get_concrete_function_garbage_collected(
-> 3052         *args, **kwargs)
   3053     graph_function._garbage_collector.release()  # pylint: disable=pr
otected-access
   3054     return graph_function

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/eager/function.py in _get_concrete_function_garbage_colle
cted(self, *args, **kwargs)
   3017       args, kwargs = None, None
   3018     with self._lock:
-> 3019       graph_function, _ = self._maybe_define_function(args, kwargs)
   3020       seen_names = set()
   3021       captured = object_identity.ObjectIdentitySet(

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
```

```
s/tensorflow/python/eager/function.py in _maybe_define_function(self, args, k
wargs)
   3359
   3360           self._function_cache.missed.add(call_context_key)
-> 3361           graph_function = self._create_graph_function(args, kwargs)
   3362           self._function_cache.primary[cache_key] = graph_function
   3363

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/eager/function.py in _create_graph_function(self, args, k
wargs, override_flat_arg_shapes)
   3204             arg_names=arg_names,
   3205             override_flat_arg_shapes=override_flat_arg_shapes,
-> 3206             capture_by_value=self._capture_by_value),
   3207         self._function_attributes,
   3208         function_spec=self.function_spec,

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/framework/func_graph.py in func_graph_from_py_func(name,
python_func, args, kwargs, signature, func_graph, autograph, autograph_option
s, add_control_dependencies, arg_names, op_return_value, collections, capture
_by_value, override_flat_arg_shapes)
    988         _, original_func = tf_decorator.unwrap(python_func)
    989
--> 990       func_outputs = python_func(*func_args, **func_kwargs)
    991
    992       # invariant: `func_outputs` contains only Tensors, CompositeTen
sors,

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/data/ops/dataset_ops.py in wrapper_fn(*args)
   3516           attributes=defun_kwargs)
   3517       def wrapper_fn(*args):  # pylint: disable=missing-docstring
-> 3518         ret = _wrapper_helper(*args)
   3519         ret = structure.to_tensor_list(self._output_structure, ret)
   3520         return [ops.convert_to_tensor(t) for t in ret]

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/data/ops/dataset_ops.py in _wrapper_helper(*args)
   3451         nested_args = (nested_args,)
   3452
-> 3453       ret = autograph.tf_convert(func, ag_ctx)(*nested_args)
   3454       # If `func` returns a list of tensors, `nest.flatten()` and
   3455       # `ops.convert_to_tensor()` would conspire to attempt to stack

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/autograph/impl/api.py in wrapper(*args, **kwargs)
    665       try:
    666         with conversion_ctx:
--> 667           return converted_call(f, args, kwargs, options=options)
    668       except Exception as e:  # pylint:disable=broad-except
    669         if hasattr(e, 'ag_error_metadata'):

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/autograph/impl/api.py in converted_call(f, args, kwargs,
caller_fn_scope, options)
```

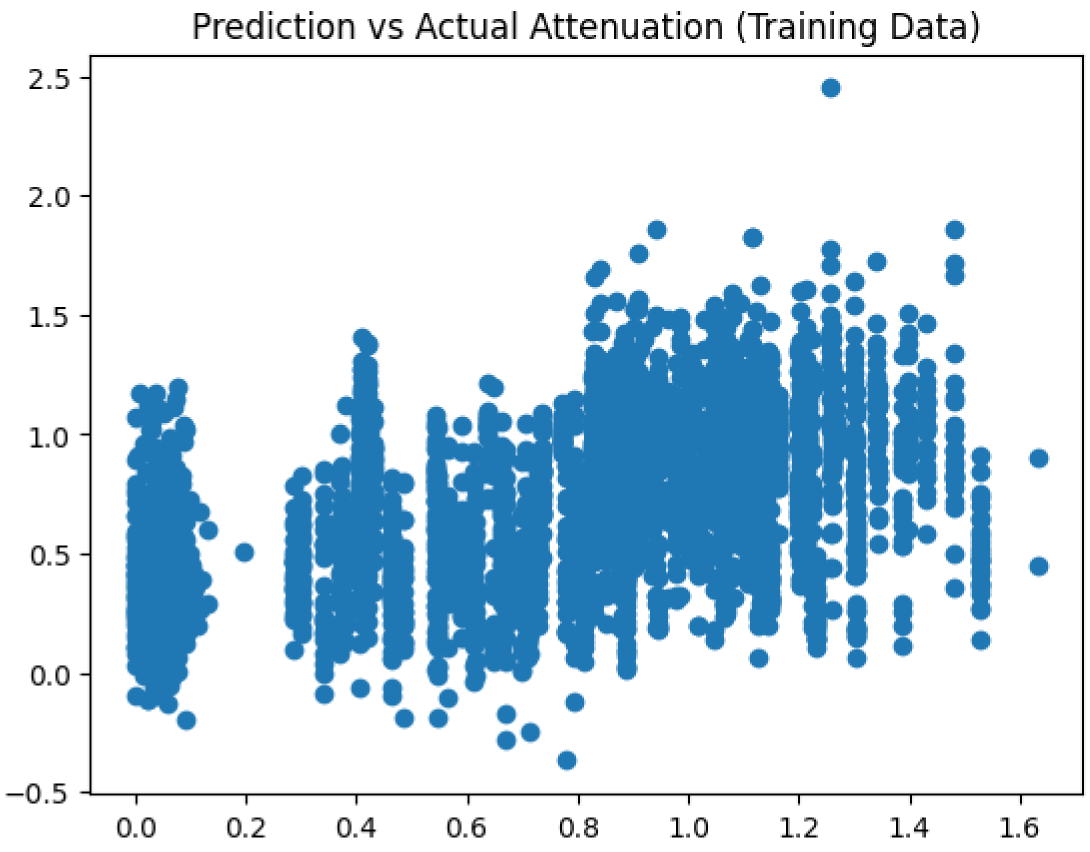
```
    394 
    395   if not options.user_requested and conversion.is_allowlisted(f):
--> 396     return _call_unconverted(f, args, kwargs, options)
    397 
    398   # internal_convert_user_code is for example turned off when issuing
a dynamic

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/autograph/impl/api.py in _call_unconverted(f, args, kwarg
s, options, update_cache)
    476 
    477   if kwargs is not None:
--> 478     return f(*args, **kwargs)
    479   return f(*args)
    480 

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/keras/engine/data_adapter.py in grab_batch(i, data)
    383 
    384     def grab_batch(i, data):
--> 385       return nest.map_structure(lambda d: array_ops.gather(d, i, axis
=0), data)
    386 
    387     dataset = dataset.map(

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/util/nest.py in map_structure(func, *structure, **kwargs)
    657 
    658   return pack_sequence_as(
--> 659       structure[0], [func(*x) for x in entries],
    660       expand_composites=expand_composites)
    661 

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/util/nest.py in <listcomp>(.0)
    657 
    658   return pack_sequence_as(
--> 659       structure[0], [func(*x) for x in entries],
    660       expand_composites=expand_composites)
    661 

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/keras/engine/data_adapter.py in <lambda>(d)
    383 
    384     def grab_batch(i, data):
--> 385       return nest.map_structure(lambda d: array_ops.gather(d, i, axis
=0), data)
    386 
    387     dataset = dataset.map(

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/util/dispatch.py in wrapper(*args, **kwargs)
    199     """Call target, and fall back on dispatchers if there is a TypeEr
ror."""
    200     try:
--> 201       return target(*args, **kwargs)
```

```
    202     except (TypeError, ValueError):
    203       # Note: convert_to_eager_tensor currently raises a ValueError,
not a

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/ops/array_ops.py in gather(***failed resolving arguments*
**)
   4813     return params.sparse_read(indices, name=name)
   4814   except AttributeError:
-> 4815     return gen_array_ops.gather_v2(params, indices, axis, name=name)
   4816
   4817

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/ops/gen_array_ops.py in gather_v2(params, indices, axis,
batch_dims, name)
   3800   _, _, _op, _outputs = _op_def_library._apply_op_helper(
   3801         "GatherV2", params=params, indices=indices, axis=axis,
-> 3802                     batch_dims=batch_dims, name=name)
   3803   _result = _outputs[:]
   3804   if _execute.must_record_gradient():

/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-package
s/tensorflow/python/framework/op_def_library.py in _apply_op_helper(op_type_n
ame, name, **keywords)
    743     must_colocate_inputs = [val for arg, val in zip(op_def.input_arg,
inputs)
    744                               if arg.is_ref]
--> 745     with _MaybeColocateWith(must_colocate_inputs):
    746       # Add Op to graph
    747       # pylint: disable=protected-access

KeyboardInterrupt:
```



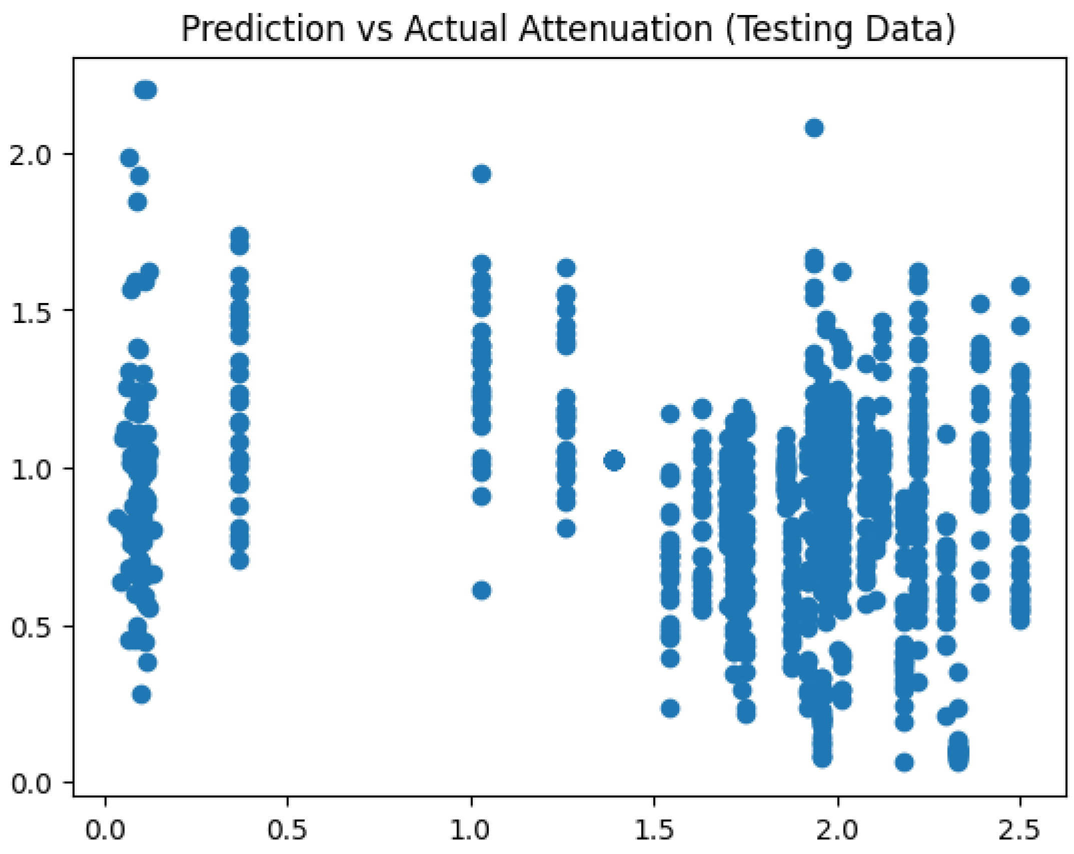
In [16]:
```
#test accuracy of the model in predicting real attenuation values.
pred = model.predict(train_dataset)
test_labels = np.concatenate([y for x, y in train_dataset], axis=0) #print(p
plt.scatter(test_labels, pred)
plt.title("Prediction vs Actual Attenuation (Training Data)")
```

Out[16]:
```
Text(0.5, 1.0, 'Prediction vs Actual Attenuation (Training Data)')
```



```
In [17]: #test accuracy of the model in predicting real attenuation values.
         pred = model.predict(test_dataset)
         test_labels = np.concatenate([y for x, y in test_dataset], axis=0) #print(pr
         plt.scatter(test_labels, pred)
         plt.title("Prediction vs Actual Attenuation (Testing Data)")
```

```
Out[17]: Text(0.5, 1.0, 'Prediction vs Actual Attenuation (Testing Data)')
```


Prediction vs Actual Attenuation (Testing Data)
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```