```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'new-plant-diseases-dataset:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F78313%2F182633%2Fbundle%2Farchi

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading new-plant-diseases-dataset, 2897709187 bytes compressed
[==================================================] 2897709187 bytes downloaded
```

```
Downloaded and uncompressed: new-plant-diseases-dataset
Data source import complete.
```

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save &
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid/Corn_(mai
/kaggle/input/new-plant-diseases-dataset/test/test/AppleCedarRust2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight4.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/CornCommonRust1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/AppleScab2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/AppleScab3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus4.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight5.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/CornCommonRust2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/AppleCedarRust1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoHealthy1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus5.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoHealthy1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/CornCommonRust3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight4.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoHealthy4.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/AppleCedarRust3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/AppleCedarRust4.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/AppleScab1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoHealthy2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoHealthy2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight1.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight5.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus6.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus2.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoHealthy3.JPG
/kaggle/input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight6.JPG
```

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
%matplotlib inline
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.applications import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras import Model, layers
from keras.models import load_model, model_from_json


train_datagen = ImageDataGenerator(
    shear_range=10,
    zoom_range=0.4,
    horizontal_flip=True,
    rotation_range=20,  # Random rotation within the range of [-20, 20] degrees
    width_shift_range=0.1,  # Randomly shift images horizontally by up to 10% of the width
    height_shift_range=0.1,  # Randomly shift images vertically by up to 10% of the height
    preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
    '/input/new-plant-diseases-dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train',
    batch_size=32,
    class_mode='binary',
    target_size=(224,224))

validation_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input)

validation_generator = validation_datagen.flow_from_directory(
    '/input/new-plant-diseases-dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/valid',
    shuffle=False,
    class_mode='binary',
    target_size=(224,224))

 Found 70295 images belonging to 38 classes.
 Found 17572 images belonging to 38 classes.


conv_base = ResNet50(include_top=False,
                     weights='imagenet')

for layer in conv_base.layers:
    layer.trainable = False

x = conv_base.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128, activation='relu')(x)
predictions = layers.Dense(38, activation='softmax')(x)
model = Model(conv_base.input, predictions)

optimizer = keras.optimizers.Adam()
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])



model.fit(x = train_generator,
          validation_data = validation_generator,
          steps_per_epoch = 16,
          epochs=5)


 Epoch 1/5
 16/16 [==============================] - 66s 4s/step - loss: 0.1770 - accuracy: 0.9375 - val_loss: 0.1621 - val_accuracy: 0.9457
 Epoch 2/5
 16/16 [==============================] - 65s 4s/step - loss: 0.1578 - accuracy: 0.9434 - val_loss: 0.1645 - val_accuracy: 0.9451
 Epoch 3/5
 16/16 [==============================] - 64s 4s/step - loss: 0.1739 - accuracy: 0.9375 - val_loss: 0.1795 - val_accuracy: 0.9404
 Epoch 4/5
 16/16 [==============================] - 65s 4s/step - loss: 0.1977 - accuracy: 0.9551 - val_loss: 0.1796 - val_accuracy: 0.9414
 Epoch 5/5
 16/16 [==============================] - 89s 6s/step - loss: 0.1876 - accuracy: 0.9375 - val_loss: 0.1705 - val_accuracy: 0.9419
 <keras.src.callbacks.History at 0x7900d6c221d0>
```

```
# architecture and weights to HDF5
model.save('models/keras/model.h5')

# architecture to JSON, weights to HDF5
model.save_weights('models/keras/weights.h5')
with open('models/keras/architecture.json', 'w') as f:
    f.write(model.to_json())
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `
  saving_api.save_model(
```

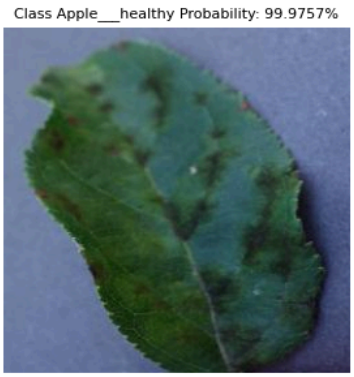
```
# architecture and weights from HDF5
model = load_model('models/keras/model.h5')

# architecture from JSON, weights from HDF5
with open('models/keras/architecture.json') as f:
    model = model_from_json(f.read())
model.load_weights('models/keras/weights.h5')


validation_img_paths = ["/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus5.JPG",
                        "/input/new-plant-diseases-dataset/test/test/TomatoYellowCurlVirus2.JPG",
                        "/input/new-plant-diseases-dataset/test/test/AppleCedarRust1.JPG",
                        "/input/new-plant-diseases-dataset/test/test/AppleScab2.JPG",
                        "/input/new-plant-diseases-dataset/test/test/CornCommonRust3.JPG",
                        "/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight2.JPG",
                        "/input/new-plant-diseases-dataset/test/test/PotatoEarlyBlight4.JPG",
                        "/input/new-plant-diseases-dataset/test/test/TomatoHealthy3.JPG",
                        "/input/new-plant-diseases-dataset/test/test/TomatoHealthy4.JPG",
                        "/input/new-plant-diseases-dataset/test/test/CornCommonRust1.JPG"]
img_list = [Image.open(img_path) for img_path in validation_img_paths]


img_size = 224  # or whatever size you're using for resizing the images

# Assuming you have a model already defined and compiled

names = ["Apple_scab", "Apple___Black_rot",
         "Cedar_apple_rust", "Apple___healthy",
         "Blueberry___healthy",
         "Cherry_(including_sour)___Powdery_mildew",
         "Cherry_(including_sour)___healthy",
         "Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot",
         "Corn_(maize)___Common_rust_",
         "Corn_(maize)___Northern_Leaf_Blight",
         "Corn_(maize)___healthy",
         "Grape___Black_rot",
         "Grape___Esca_(Black_Measles)",
         "Grape___Leaf_blight_(Isariopsis_Leaf_Spot)",
         "Grape___healthy",
         "Orange___Haunglongbing_(Citrus_greening)",
         "Peach___Bacterial_spot",
         "Peach___healthy",
         "Pepper,_bell___Bacterial_spot",
         "Pepper,_bell___healthy",
         "Potato___Early_blight",
         "Potato___Late_blight",
         "Potato___healthy",
         "Raspberry___healthy",
         "Soybean___healthy",
         "Squash___Powdery_mildew",
         "Strawberry___Leaf_scorch",
         "Strawberry___healthy",
         "Tomato___Bacterial_spot",
         "Tomato___Early_blight",
         "Tomato___Late_blight",
         "Tomato___Leaf_Mold",
         "Tomato___Septoria_leaf_spot",
         "Tomato___Spider_mites Two-spotted_spider_mite",
         "Tomato___Target_Spot",
         "Tomato___Tomato_Yellow_Leaf_Curl_Virus",
         "Tomato___Tomato_mosaic_virus",
         "Tomato___healthy"
]
# Preprocess the validation images
validation_batch = np.stack([preprocess_input(np.array(img.resize((img_size, img_size))))
```

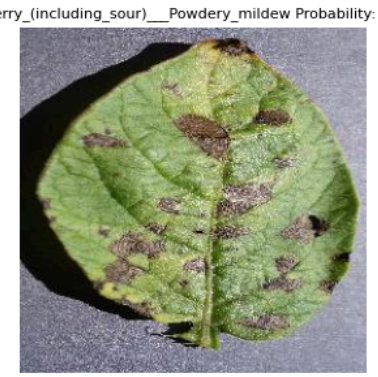
```
                                for img in img_list])

    # Predict probabilities for each class
    pred_probs = model.predict(validation_batch)

    print(pred_probs)

    length = len(pred_probs)
    maxProbs = []
    for i in range(length):
      max_value = max(pred_probs[i])
      max_index = np.argmax(pred_probs[i])

      print(i, max_value, max_index)
      maxProbs.append([max_value,max_index])


    # Plot the images with their respective class probabilities
    fig, axs = plt.subplots(5, 3, figsize=(12, 20))  # 3 rows, 5 columns
    axs = axs.ravel()  # Flatten the axs array for easy indexing
    for i, img in enumerate(img_list):
        ax = axs[i]
        ax.axis('off')

        # Assuming pred_probs[i] contains the predicted probabilities for the i-th image
        class_probabilities = pred_probs[i]
        print(class_probabilities)

        # Assuming the model predicts probabilities for 38 classes
        length = len(maxProbs)

        ax.set_title("Class {} Probability: {:.4f}%".format(names[i] ,maxProbs[i][0] * 100), fontsize=8)

        ax.imshow(img)

    plt.show()
```

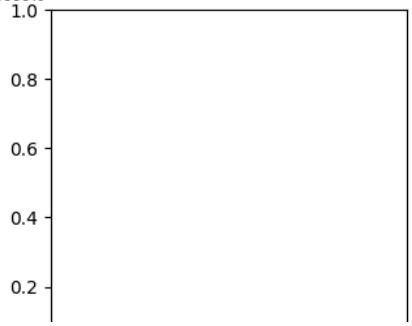
```
1/1 [==============================] - 0s 28ms/step
[[1.98516707e-11 3.31414719e-14 2.26010521e-09 1.12302409e-12
  4.33272732e-12 2.50655660e-07 1.79684804e-12 2.25962220e-13
  5.06167103e-14 1.31036709e-11 8.93654595e-11 1.54815657e-11
  6.07126988e-11 8.81263324e-13 3.37964691e-11 6.07500494e-09
  2.46100251e-09 4.28017561e-12 5.11177545e-09 1.24994270e-09
  2.85803516e-11 6.11530479e-11 1.65152265e-14 1.20678573e-13
  4.65450771e-14 2.41652542e-09 7.06271088e-13 2.56734878e-09
  3.05344656e-05 1.62154561e-06 1.87415790e-05 6.25999178e-07
  1.83616442e-06 1.00838527e-07 1.65637015e-08 9.99946237e-01
  9.66401501e-08 4.45484455e-10]
 [9.34027974e-14 6.50657623e-18 2.31620895e-16 4.49965152e-12
  9.52412303e-13 9.61564856e-12 7.69933976e-15 4.51187021e-13
  1.07317626e-14 3.64622351e-12 1.15138914e-11 2.65894975e-14
  7.53630318e-12 8.97421243e-16 6.66524725e-18 5.02426989e-10
  4.30934016e-13 1.73563746e-12 1.50282209e-09 1.39368128e-10
  6.46301756e-17 1.33359070e-11 1.47394098e-12 3.73161614e-12
  2.12498734e-15 8.33059885e-14 1.78912942e-14 4.52447663e-10
  6.87325974e-10 7.57127196e-08 1.06489559e-07 9.63930142e-06
  3.32224422e-11 1.15434639e-04 3.35967876e-09 9.99874830e-01
  2.33753443e-08 3.27283443e-11]
 [9.81081527e-08 6.63377335e-08 9.99988794e-01 2.39170433e-10
  6.39980135e-06 1.21129782e-08 2.89388158e-09 7.22228050e-12
  1.77932524e-09 7.12830877e-14 2.73442900e-11 1.38982218e-12
  3.90257675e-08 6.54886978e-10 2.35350459e-08 4.82892347e-07
  2.02871695e-11 6.46498410e-10 3.74910428e-06 9.36148759e-09
  1.64447733e-09 6.77309736e-11 2.00332043e-10 5.67238589e-10
  9.21100286e-14 1.23302518e-12 2.34776192e-11 4.99364994e-09
  9.30266836e-11 2.48663312e-10 1.19171411e-07 8.03766984e-11
  8.79774209e-09 2.11249542e-11 5.80404134e-08 1.12127694e-07
  3.27102825e-08 1.50792281e-10]
 [9.99757230e-01 4.05664796e-05 6.88864966e-05 1.45846834e-05
  2.48809458e-12 1.05656022e-06 2.95327562e-09 4.32139331e-07
  1.15003367e-07 2.08131308e-08 3.86938703e-10 2.71129907e-12
  4.38525674e-11 7.41708872e-09 1.36938363e-08 3.17885025e-07
  6.86030180e-05 8.79007516e-08 5.89768661e-06 4.08837009e-09
  1.67121481e-08 1.69365109e-07 1.99526520e-10 1.02811275e-11
  3.22791998e-08 1.13860887e-09 5.11656828e-11 2.28680520e-07
  4.14526730e-05 1.15868093e-09 1.07375966e-08 8.19281283e-08
  5.12402778e-08 2.20124918e-09 1.73826194e-07 2.00697514e-10
  2.22548750e-11 5.94173148e-12]
 [1.37030609e-10 2.58261685e-12 3.45918977e-07 1.45691679e-15
  6.72536332e-12 4.09548480e-13 4.17622127e-13 4.95079255e-10
  9.99998689e-01 2.44721274e-11 2.80613770e-12 8.09656824e-16
  4.48144169e-12 7.01023180e-12 3.09929361e-12 1.11597209e-09
  7.87197876e-11 3.46322498e-10 1.76138428e-07 1.97628891e-09
  7.70845443e-10 3.39667031e-12 4.59037981e-11 7.95871480e-14
  3.89595968e-19 3.64555337e-13 2.18289555e-14 2.04163021e-13
  2.59504874e-13 9.24994664e-11 7.74591626e-07 9.59210836e-12
  7.09095543e-13 1.85701649e-12 1.73172310e-11 1.63192620e-13
  1.96989858e-13 8.35819261e-11]
 [3.06630170e-07 1.14924049e-07 3.22127164e-07 5.93170424e-13
  5.00304615e-12 5.98890253e-08 5.03830528e-12 4.67458179e-08
  2.38132605e-04 3.93136107e-10 1.14983066e-12 1.61738515e-12
  3.29947514e-10 2.62591802e-05 1.85333179e-10 6.52640331e-10
  6.28271124e-08 7.09774461e-11 2.23996904e-04 4.89439920e-08
  9.99095321e-01 3.95029056e-04 7.85901648e-06 3.53074688e-12
  6.88310303e-13 4.81059803e-10 9.90101285e-07 2.80713175e-10
  5.79693094e-08 2.04866137e-06 8.24238189e-09 3.98842426e-09
  2.39247993e-06 3.24115973e-07 6.49005142e-06 3.40634014e-11
  1.90341343e-08 5.05283504e-09]
 [1.92597369e-08 4.70171853e-11 1.30747608e-08 5.80732565e-13
  2.51825600e-12 6.93324687e-09 2.22279687e-13 7.96621435e-10
  5.72501847e-07 4.31839026e-12 5.78521199e-14 4.32630853e-14
  5.25329346e-12 8.21980734e-07 1.74829194e-12 1.81174242e-09
  1.72834504e-07 8.51439058e-12 1.03474244e-01 1.70923585e-07
  8.96512270e-01 1.18815569e-06 4.12456831e-12 4.42398980e-18
  2.55676239e-15 3.26516938e-11 3.05058617e-11 2.64311056e-11
  3.26463623e-06 7.08912069e-07 1.96161380e-07 4.21518642e-08
  5.92676724e-06 6.11176034e-08 4.26419263e-07 1.30073474e-09
  1.29021089e-13 1.11988092e-11]
 [1.46377535e-10 7.57027427e-12 5.59420263e-17 1.06643488e-13
  1.31929689e-10 5.37188458e-13 1.20056002e-14 2.37097592e-10
  3.94755062e-11 4.07326489e-14 4.88841953e-11 3.76486425e-17
  1.55681648e-11 1.21499056e-13 7.93793362e-13 1.24585866e-14
  1.84777243e-17 2.85196085e-11 6.95654701e-10 3.53068742e-07
  1.76354653e-10 3.03089043e-12 4.94613305e-05 4.73756279e-09
  2.10327422e-10 2.82215478e-08 5.93873258e-13 7.25903760e-07
  6.91832225e-13 1.62976868e-08 1.33083699e-06 6.60875457e-07
  4.74955328e-08 3.54044721e-04 9.71195381e-03 7.58269891e-09
  3.82408994e-04 9.89498973e-01]
 [2.47072003e-13 1.88282341e-11 4.30799872e-16 4.13178436e-10
  1.84247488e-13 7.05579150e-13 1.72500750e-10 1.44259171e-12
  5.93171623e-11 1.24951427e-12 3.80431153e-09 5.15132034e-16
```

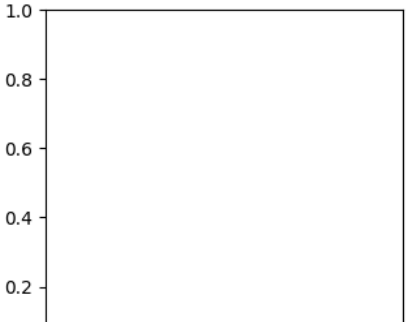
```
  6.45675943e-11 7.8797382e-15 1.3495180e-11 2.3711387e-16
  1.90119323e-13 3.84203940e-06 1.10969258e-13 1.49523999e-07
  6.84319680e-13 1.11199722e-14 9.45433172e-08 1.96236360e-09
  1.72377179e-09 5.19387306e-08 7.53441493e-12 4.63023326e-07
  7.47429896e-12 1.43011016e-08 1.65257745e-08 1.87581605e-07
  3.14584803e-08 6.96522984e-05 2.06157956e-02 6.38810693e-09
  6.63752726e-05 9.79243279e-01]
 [3.25233243e-12 1.09700440e-14 1.98116575e-11 2.31376102e-16
  4.64186727e-14 5.29033782e-15 2.78261030e-12 9.24628107e-09
  1.00000000e+00 2.40789033e-09 2.12593883e-11 4.17669987e-19
  4.51857984e-15 3.23631435e-14 8.99799500e-16 1.18654131e-09
  1.17947576e-14 6.29901166e-12 3.76905884e-09 1.74401127e-09
  2.16835300e-11 8.37027116e-13 2.15940332e-09 1.00513341e-13
  1.02060563e-17 5.50057733e-15 8.84300195e-15 1.05546405e-10
  5.38680970e-13 1.34631661e-12 2.40332607e-08 1.35936415e-10
  2.97146917e-14 1.72319071e-12 2.81732450e-11 1.49110672e-12
  1.65443555e-15 7.89354519e-12]]
0 0.99994624 35
1 0.99987483 35
2 0.9999888 2
3 0.99975723 0
4 0.9999987 8
5 0.9990953 20
6 0.89651227 20
7 0.989499 37
8 0.9792433 37
9 1.0 8
[1.9851671e-11 3.3141472e-14 2.2601052e-09 1.1230241e-12 4.3327273e-12
 2.5065566e-07 1.7968480e-12 2.2596222e-13 5.0616710e-14 1.3103671e-11
 8.9365459e-11 1.5481566e-11 6.0712699e-11 8.8126332e-13 3.3796469e-11
 6.0750049e-09 2.4610025e-09 4.2801756e-12 5.1117754e-09 1.2499427e-09
 2.8580352e-11 6.1153048e-11 1.6515226e-14 1.2067857e-13 4.6545077e-14
 2.4165254e-09 7.0627109e-13 2.5673488e-09 3.0534466e-05 1.6215456e-06
 1.8741579e-05 6.2599918e-07 1.8361644e-06 1.0083853e-07 1.6563702e-08
 9.9994624e-01 9.6640150e-08 4.4548445e-10]
[9.3402797e-14 6.5065762e-18 2.3162089e-16 4.4996515e-12 9.5241230e-13
 9.6156486e-12 7.6993398e-15 4.5118702e-13 1.0731763e-14 3.6462235e-12
 1.1513891e-11 2.6589498e-14 7.5363032e-12 8.9742124e-16 6.6652473e-18
 5.0242699e-10 4.3093402e-13 1.7356375e-12 1.5028221e-09 1.3936813e-10
 6.4630176e-17 1.3335907e-11 1.4739410e-12 3.7316161e-12 2.1249873e-15
 8.3305989e-14 1.7891294e-14 4.5244766e-10 6.8732597e-10 7.5712720e-08
 1.0648956e-07 9.6393014e-06 3.3222442e-11 1.1543464e-04 3.3596788e-09
 9.9987483e-01 2.3375344e-08 3.2728344e-11]
[9.81081527e-08 6.63377335e-08 9.99988794e-01 2.39170433e-10
 6.39980135e-06 1.21129782e-08 2.89388158e-09 7.22228050e-12
 1.77932524e-09 7.12830877e-14 2.73442900e-11 1.38982218e-12
 3.90257675e-08 6.54886978e-10 2.35350459e-08 4.82892347e-07
 2.02871695e-11 6.46498410e-10 3.74910428e-06 9.36148759e-09
 1.64447733e-09 6.77309736e-11 2.00332043e-10 5.67238589e-10
 9.21100286e-14 1.23302518e-12 2.34776192e-11 4.99364994e-09
 9.30266836e-11 2.48663312e-10 1.19171411e-07 8.03766984e-11
 8.79774209e-09 2.11249542e-11 5.80404134e-08 1.12127694e-07
 3.27102825e-08 1.50792281e-10]
[9.99757230e-01 4.05664796e-05 6.88864966e-05 1.45846834e-05
 2.48809458e-12 1.05656022e-06 2.95327562e-09 4.32139331e-07
 1.15003367e-07 2.08131308e-08 3.86938703e-10 2.71129907e-12
 4.38525674e-11 7.41708872e-09 1.36938363e-08 3.17885025e-07
 6.86030180e-05 8.79007516e-08 5.89768661e-06 4.08837009e-09
 1.67121481e-08 1.69365109e-07 1.99526520e-10 1.02811275e-11
 3.22791998e-08 1.13860887e-09 5.11656828e-11 2.28680520e-07
 4.14526730e-05 1.15868093e-09 1.07375966e-08 8.19281283e-08
 5.12402778e-08 2.20124918e-09 1.73826194e-07 2.00697514e-10
 2.22548750e-11 5.94173148e-12]
[1.3703061e-10 2.5826168e-12 3.4591898e-07 1.4569168e-15 6.7253633e-12
 4.0954848e-13 4.1762213e-13 4.9507926e-10 9.9999869e-01 2.4472127e-11
 2.8061377e-12 8.0965682e-16 4.4814417e-12 7.0102318e-12 3.0992936e-12
 1.1159721e-09 7.8719788e-11 3.4632250e-10 1.7613843e-07 1.9762889e-09
 7.7084544e-10 3.3966703e-12 4.5903798e-11 7.9587148e-14 3.8959597e-19
 3.6455534e-13 2.1828956e-14 2.0416302e-13 2.5950487e-13 9.2499466e-11
 7.7459163e-07 9.5921084e-12 7.0909554e-13 1.8570165e-12 1.7317231e-11
 1.6319262e-13 1.9698986e-13 8.3581926e-11]
[3.0663017e-07 1.1492405e-07 3.2212716e-07 5.9317042e-13 5.0030462e-12
 5.9889025e-08 5.0383053e-12 4.6745818e-08 2.3813260e-04 3.9313611e-10
 1.1498307e-12 1.6173852e-12 3.2994751e-10 2.6259180e-05 1.8533318e-10
 6.5264033e-10 6.2827112e-08 7.0977446e-11 2.2399690e-04 4.8943992e-08
 9.9909532e-01 3.9502906e-04 7.8590165e-06 3.5307469e-12 6.8831030e-13
 4.8105980e-10 9.9010128e-07 2.8071317e-10 5.7969309e-08 2.0486614e-06
 8.2423819e-09 3.9884243e-09 2.3924799e-06 3.2411597e-07 6.4900514e-06
 3.4063401e-11 1.9034134e-08 5.0528350e-09]
[1.92597369e-08 4.70171853e-11 1.30747608e-08 5.80732565e-13
 2.51825600e-12 6.93324687e-09 2.22279687e-13 7.96621435e-10
 5.72501847e-07 4.31839026e-12 5.78521199e-14 4.32630853e-14
 5.25329346e-12 8.21980734e-07 1.74829194e-12 1.81174242e-09
 1.72834504e-07 8.51439058e-12 1.03474244e-01 1.70923585e-07
 8.96512270e-01 1.18815569e-06 4.12456831e-12 4.42398980e-18
```

```
 2.55676239e-15 3.26516938e-11 3.05058617e-11 2.64311056e-11
 3.26463623e-06 7.08912069e-07 1.96161380e-07 4.21518642e-08
 5.92676724e-06 6.11176034e-08 4.26419263e-07 1.30073474e-09
 1.29021089e-13 1.11988092e-11]
[1.4637754e-10 7.5702743e-12 5.5942026e-17 1.0664349e-13 1.3192969e-10
 5.3718846e-13 1.2005600e-14 2.3709759e-10 3.9475506e-11 4.0732649e-14
 4.8884195e-11 3.7648643e-17 1.5568165e-11 1.2149906e-13 7.9379336e-13
 1.2458587e-14 1.8477724e-17 2.8519609e-11 6.9565470e-10 3.5306874e-07
 1.7635465e-10 3.0308904e-12 4.9461330e-05 4.7375628e-09 2.1032742e-10
 2.8221548e-08 5.9387326e-13 7.2590376e-07 6.9183223e-13 1.6297687e-08
 1.3308370e-06 6.6087546e-07 4.7495533e-08 3.5404472e-04 9.7119538e-03
 7.5826989e-09 3.8240899e-04 9.8949897e-01]
[2.4707200e-13 1.8828234e-11 4.3079987e-16 4.1317844e-10 1.8424749e-13
 7.0557915e-13 1.7250075e-10 1.4425917e-12 5.9317162e-11 1.2495143e-12
 3.8043115e-09 5.1513203e-16 6.4567594e-11 7.8797383e-15 1.3495181e-11
 2.3711388e-16 1.9011932e-13 3.8420394e-06 1.1096926e-13 1.4952400e-07
 6.8431968e-13 1.1119972e-14 9.4543317e-08 1.9623636e-09 1.7237718e-09
 5.1938731e-08 7.5344149e-12 4.6302333e-07 7.4742990e-12 1.4301102e-08
 1.6525775e-08 1.8758161e-07 3.1458480e-08 6.9652298e-05 2.0615796e-02
 6.3881069e-09 6.6375273e-05 9.7924328e-01]
[3.25233243e-12 1.09700440e-14 1.98116575e-11 2.31376102e-16
 4.64186727e-14 5.29033782e-15 2.78261030e-12 9.24628107e-09
 1.00000000e+00 2.40789033e-09 2.12593883e-11 4.17669987e-19
 4.51857984e-15 3.23631435e-14 8.99799500e-16 1.18654131e-09
 1.17947576e-14 6.29901166e-12 3.76905884e-09 1.74401127e-09
 2.16835300e-11 8.37027116e-13 2.15940332e-09 1.00513341e-13
 1.02060563e-17 5.50057733e-15 8.84300195e-15 1.05546405e-10
 5.38680970e-13 1.34631661e-12 2.40332607e-08 1.35936415e-10
 2.97146917e-14 1.72319071e-12 2.81732450e-11 1.49110672e-12
 1.65443555e-15 7.89354519e-12]
```






1.0
0.8
0.6
0.4
0.2


1.0
0.8
0.6
0.4
0.2