dog_app

July 25, 2021

1 Convolutional Neural Networks

1.1 Project: Write an Algorithm for a Dog Identification App

In this notebook, some template code has already been provided for you, and you will need to implement additional functionality to successfully complete this project. You will not need to modify the included code beyond what is requested. Sections that begin with **'(IMPLEMENTATION)'** in the header indicate that the following block of code will require additional functionality which you must provide. Instructions will be provided for each section, and the specifics of the implementation are marked in the code block with a 'TODO' statement. Please be sure to read the instructions carefully!

> **Note**: Once you have completed all of the code implementations, you need to finalize your work by exporting the Jupyter Notebook as an HTML document. Before exporting the notebook to html, all of the code cells need to have been run so that reviewers can see the final implementation and output. You can then export the notebook by using the menu above and navigating to **File -> Download as -> HTML (.html)**. Include the finished document along with this notebook as your submission.

In addition to implementing code, there will be questions that you must answer which relate to the project and your implementation. Each section where you will answer a question is preceded by a **'Question X'** header. Carefully read each question and provide thorough answers in the following text boxes that begin with **'Answer:'**. Your project submission will be evaluated based on your answers to each of the questions and the implementation you provide.

> **Note:** Code and Markdown cells can be executed using the **Shift + Enter** keyboard shortcut. Markdown cells can be edited by double-clicking the cell to enter edit mode.

The rubric contains *optional* "Stand Out Suggestions" for enhancing the project beyond the minimum requirements. If you decide to pursue the "Stand Out Suggestions", you should include the code in this Jupyter notebook.

Step 0: Import Datasets

Make sure that you've downloaded the required human and dog datasets:

Note: if you are using the Udacity workspace, you *DO NOT* need to re-download these - they can be found in the `/data` folder as noted in the cell below.

- Download the dog dataset. Unzip the folder and place it in this project's home directory, at the location `/dog_images`.
- Download the human dataset. Unzip the folder and place it in the home directory, at location `/lfw`.

Note: If you are using a Windows machine, you are encouraged to use 7zip to extract the folder.

In the code cell below, we save the file paths for both the human (LFW) dataset and dog dataset in the numpy arrays `human_files` and `dog_files`.

```
In [1]: import numpy as np
        from glob import glob

        # load filenames for human and dog images
        human_files = np.array(glob("/data/lfw/*/*"))
        dog_files = np.array(glob("/data/dog_images/*/*/*"))

        # print number of images in each dataset
        print('There are %d total human images.' % len(human_files))
        print('There are %d total dog images.' % len(dog_files))
```

```
There are 13233 total human images.
There are 8351 total dog images.
```

Step 1: Detect Humans

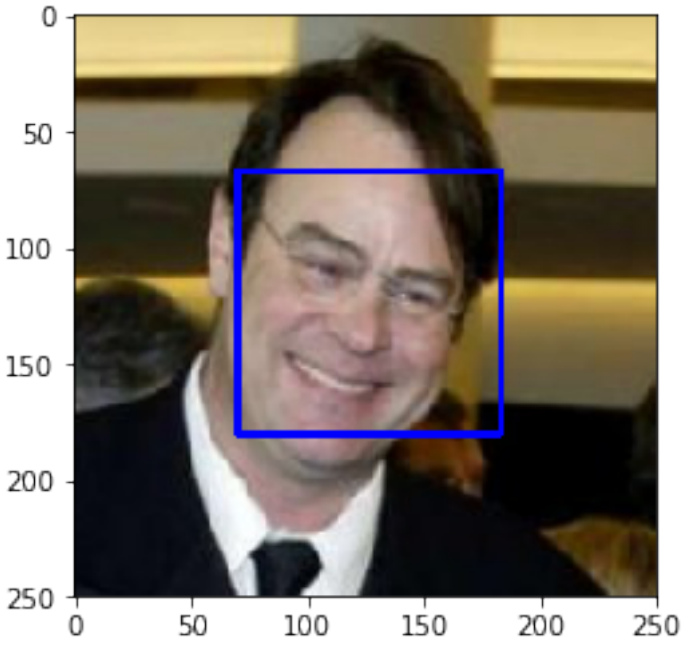
In this section, we use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images.

OpenCV provides many pre-trained face detectors, stored as XML files on github. We have downloaded one of these detectors and stored it in the `haarcascades` directory. In the next code cell, we demonstrate how to use this detector to find human faces in a sample image.

```
In [2]: import cv2
        import matplotlib.pyplot as plt
        %matplotlib inline

        # extract pre-trained face detector
        face_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_alt.xml')

        # load color (BGR) image
        img = cv2.imread(human_files[0])
        # convert BGR image to grayscale
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # find faces in image
        faces = face_cascade.detectMultiScale(gray)

        # print number of faces detected in the image
        print('Number of faces detected:', len(faces))
```

```
        # get bounding box for each detected face
        for (x,y,w,h) in faces:
            # add bounding box to color image
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

        # convert BGR image to RGB for plotting
        cv_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        # display the image, along with bounding box
        plt.imshow(cv_rgb)
        plt.show()
```

```
Number of faces detected: 1
```



Before using any of the face detectors, it is standard procedure to convert the images to grayscale. The `detectMultiScale` function executes the classifier stored in `face_cascade` and takes the grayscale image as a parameter.

In the above code, `faces` is a numpy array of detected faces, where each row corresponds to a detected face. Each detected face is a 1D array with four entries that specifies the bounding box of the detected face. The first two entries in the array (extracted in the above code as `x` and `y`) specify the horizontal and vertical positions of the top left corner of the bounding box. The last two entries in the array (extracted here as `w` and `h`) specify the width and height of the box.

1.1.1 Write a Human Face Detector

We can use this procedure to write a function that returns `True` if a human face is detected in an image and `False` otherwise. This function, aptly named `face_detector`, takes a string-valued file path to an image as input and appears in the code block below.

```
In [3]: # returns "True" if face is detected in image stored at img_path
        def face_detector(img_path):
            img = cv2.imread(img_path)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray)
            return len(faces) > 0
```

1.1.2 (IMPLEMENTATION) Assess the Human Face Detector

Question 1: Use the code cell below to test the performance of the `face_detector` function.
- What percentage of the first 100 images in `human_files` have a detected human face?
- What percentage of the first 100 images in `dog_files` have a detected human face?

Ideally, we would like 100% of human images with a detected face and 0% of dog images with a detected face. You will see that our algorithm falls short of this goal, but still gives acceptable performance. We extract the file paths for the first 100 images from each of the datasets and store them in the numpy arrays `human_files_short` and `dog_files_short`.

Answer: (You can print out your results and/or write your percentages in this cell). See percentages below.

```
In [4]: from tqdm import tqdm

        human_files_short = human_files[:100]
        dog_files_short = dog_files[:100]

        #-#-# Do NOT modify the code above this line. #-#-#

        ## TODO: Test the performance of the face_detector algorithm
        ## on the images in human_files_short and dog_files_short.

        human_faces_true_counter = 0
        dog_faces_true_counter = 0

        for i in human_files_short:
            result = face_detector(i)
            if result == True:
                human_faces_true_counter += 1

        human_faces_true_ratio = human_faces_true_counter/len(human_files_short)

        for i in dog_files_short:
            result = face_detector(i)
            if result == True:
                dog_faces_true_counter += 1
```

```
        dog_faces_true_ratio = dog_faces_true_counter/len(dog_files_short)

        print(f'face_detector identifies human faces in ~{human_faces_true_ratio*100}% of human
        print(f'face_detector identifies human faces in ~{dog_faces_true_ratio*100}% of dog imag
```

```
face_detector identifies human faces in ~98.0% of human images
face_detector identifies human faces in ~17.0% of dog images
```

We suggest the face detector from OpenCV as a potential way to detect human images in your algorithm, but you are free to explore other approaches, especially approaches that make use of deep learning :). Please use the code cell below to design and test your own face detection algorithm. If you decide to pursue this *optional* task, report performance on `human_files_short` and `dog_files_short`.

```
In [ ]: ### (Optional)
        ### TODO: Test performance of anotherface detection algorithm.
        ### Feel free to use as many code cells as needed.
```

Step 2: Detect Dogs

In this section, we use a pre-trained model to detect dogs in images.

1.1.3 Obtain Pre-trained VGG-16 Model

The code cell below downloads the VGG-16 model, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories.

```
In [5]: import torch
        import torchvision.models as models

        # define VGG16 model
        VGG16 = models.vgg16(pretrained=True)

        # check if CUDA is available
        use_cuda = torch.cuda.is_available()

        # move model to GPU if CUDA is available
        if use_cuda:
            VGG16 = VGG16.cuda()
```

```
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to /root/.torch/models/vgg
100%|| 553433881/553433881 [00:09<00:00, 58135871.32it/s]
```

Given an image, this pre-trained VGG-16 model returns a prediction (derived from the 1000 possible categories in ImageNet) for the object that is contained in the image.

1.1.4 (IMPLEMENTATION) Making Predictions with a Pre-trained Model

In the next code cell, you will write a function that accepts a path to an image (such as `'dogImages/train/001.Affenpinscher/Affenpinscher_00001.jpg'`) as input and returns the index corresponding to the ImageNet class that is predicted by the pre-trained VGG-16 model. The output should always be an integer between 0 and 999, inclusive.

Before writing the function, make sure that you take the time to learn how to appropriately pre-process tensors for pre-trained models in the PyTorch documentation.

```
In [6]: from PIL import Image
        import torchvision.transforms as transforms

        def VGG16_predict(img_path):
            '''
            Use pre-trained VGG-16 model to obtain index corresponding to
            predicted ImageNet class for image at specified path

            Args:
                img_path: path to an image

            Returns:
                Index corresponding to VGG-16 model's prediction
            '''

            ## TODO: Complete the function.
            ## Load and pre-process an image from the given img_path
            ## Return the *index* of the predicted class for that image

            img_transform = transforms.Compose([
                transforms.RandomResizedCrop(224),
                transforms.ToTensor()
            ])

            img = Image.open(img_path)
            img = img_transform(img)
            img = img.unsqueeze(0)
            VGG16.eval()
            if use_cuda:
                img = img.cuda()
            with torch.no_grad():
                prediction = VGG16(img)
        #     prediction = prediction.data.numpy().argmax()
            prediction = prediction.cpu().numpy().argmax()

            return prediction # predicted class index
```

1.1.5 (IMPLEMENTATION) Write a Dog Detector

While looking at the dictionary, you will notice that the categories corresponding to dogs appear in an uninterrupted sequence and correspond to dictionary keys 151-268, inclusive, to include all categories from `'Chihuahua'` to `'Mexican hairless'`. Thus, in order to check to see if an image is predicted to contain a dog by the pre-trained VGG-16 model, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).

Use these ideas to complete the `dog_detector` function below, which returns `True` if a dog is detected in an image (and `False` if not).

In [7]:
```
### returns "True" if a dog is detected in the image stored at img_path
def dog_detector(img_path):
    ## TODO: Complete the function.
    dog_indices = list(range(151, 269))
    result = int(VGG16_predict(img_path))
    if result in dog_indices:
        return True
    else:
        return False
```

1.1.6 (IMPLEMENTATION) Assess the Dog Detector

Question 2: Use the code cell below to test the performance of your `dog_detector` function.
- What percentage of the images in `human_files_short` have a detected dog?
- What percentage of the images in `dog_files_short` have a detected dog?

Answer: 1) 1% 2) 74%

In [8]:
```
### TODO: Test the performance of the dog_detector function
### on the images in human_files_short and dog_files_short.

human_true_counter = 0
dog_true_counter = 0

for i in human_files_short:
    result = dog_detector(i)
    if result == True:
        human_true_counter += 1

human_true_ratio = human_true_counter/len(human_files_short)

for i in dog_files_short:
    result = dog_detector(i)
    if result == True:
        dog_true_counter += 1

dog_true_ratio = dog_true_counter/len(dog_files_short)

print(f'VGG-based dog_detector identifies ~{human_true_ratio*100}% of human images as do
print(f'VGG-based dog_detector identifies ~{dog_true_ratio*100}% of dog images as dogs')
```

```
VGG-based dog_detector identifies ~1.0% of human images as dogs
VGG-based dog_detector identifies ~74.0% of dog images as dogs
```

We suggest VGG-16 as a potential network to detect dog images in your algorithm, but you are free to explore other pre-trained networks (such as Inception-v3, ResNet-50, etc). Please use the code cell below to test other pre-trained PyTorch models. If you decide to pursue this *optional* task, report performance on `human_files_short` and `dog_files_short`.

```
In [ ]: ### (Optional)
        ### TODO: Report the performance of another pre-trained network.
        ### Feel free to use as many code cells as needed.
```

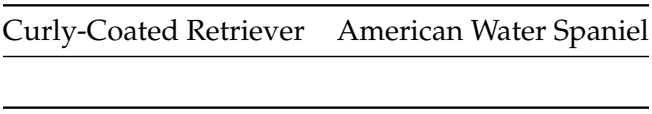
Step 3: Create a CNN to Classify Dog Breeds (from Scratch)

Now that we have functions for detecting humans and dogs in images, we need a way to predict breed from images. In this step, you will create a CNN that classifies dog breeds. You must create your CNN *from scratch* (so, you can't use transfer learning *yet*!), and you must attain a test accuracy of at least 10%. In Step 4 of this notebook, you will have the opportunity to use transfer learning to create a CNN that attains greatly improved accuracy.

We mention that the task of assigning breed to dogs from images is considered exceptionally challenging. To see why, consider that *even a human* would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel.

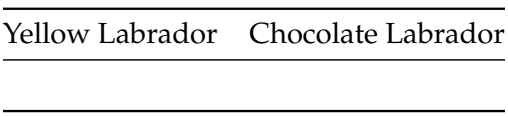
Brittany Welsh Springer Spaniel

It is not difficult to find other dog breed pairs with minimal inter-class variation (for instance, Curly-Coated Retrievers and American Water Spaniels).



Curly-Coated Retriever American Water Spaniel

Likewise, recall that labradors come in yellow, chocolate, and black. Your vision-based algorithm will have to conquer this high intra-class variation to determine how to classify all of these different shades as the same breed.



Yellow Labrador Chocolate Labrador

We also mention that random chance presents an exceptionally low bar: setting aside the fact that the classes are slightly imabalanced, a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Remember that the practice is far ahead of the theory in deep learning. Experiment with many different architectures, and trust your intuition. And, of course, have fun!

1.1.7 (IMPLEMENTATION) Specify Data Loaders for the Dog Dataset

Use the code cell below to write three separate data loaders for the training, validation, and test datasets of dog images (located at `dog_images/train`, `dog_images/valid`, and `dog_images/test`, respectively). You may find this documentation on custom datasets to be a useful resource. If you are interested in augmenting your training and/or validation data, check out the wide variety of transforms!

```
In [9]: #Check Directories
        import os
        print(os.listdir("/data"))
        print(os.listdir("/data/dog_images"))
```

```
['dog_images', 'lfw', 'bottleneck_features']
['train', 'test', 'valid']
```

```
In [10]: #Walk through all directories and files in "dog_images" and gather size info for each i
         import PIL

         image_size_dict = {}

         for (root, dirs, files) in os.walk('/data/dog_images', topdown=True):
             if len(files) > 0:
                 print('Now iterating through files in', root)
                 for i in files:
         #             print(i)
                     image = PIL.Image.open(f'{root}/{i}')
                     width, height = image.size
                     min_dimension = min([width, height])
         #             print(min_dimension)
                     image_size_dict[i] = min_dimension
```

```
Now iterating through files in /data/dog_images/train/103.Mastiff
Now iterating through files in /data/dog_images/train/059.Doberman_pinscher
Now iterating through files in /data/dog_images/train/055.Curly-coated_retriever
Now iterating through files in /data/dog_images/train/031.Borzoi
Now iterating through files in /data/dog_images/train/024.Bichon_frise
Now iterating through files in /data/dog_images/train/049.Chinese_crested
Now iterating through files in /data/dog_images/train/067.Finnish_spitz
Now iterating through files in /data/dog_images/train/130.Welsh_springer_spaniel
Now iterating through files in /data/dog_images/train/019.Bedlington_terrier
Now iterating through files in /data/dog_images/train/115.Papillon
Now iterating through files in /data/dog_images/train/126.Saint_bernard
Now iterating through files in /data/dog_images/train/013.Australian_terrier
Now iterating through files in /data/dog_images/train/116.Parson_russell_terrier
Now iterating through files in /data/dog_images/train/107.Norfolk_terrier
Now iterating through files in /data/dog_images/train/133.Yorkshire_terrier
Now iterating through files in /data/dog_images/train/032.Boston_terrier
```

```
Now iterating through files in /data/dog_images/train/108.Norwegian_buhund
Now iterating through files in /data/dog_images/train/028.Bluetick_coonhound
Now iterating through files in /data/dog_images/train/066.Field_spaniel
Now iterating through files in /data/dog_images/train/129.Tibetan_mastiff
Now iterating through files in /data/dog_images/train/005.Alaskan_malamute
Now iterating through files in /data/dog_images/train/102.Manchester_terrier
Now iterating through files in /data/dog_images/train/034.Boxer
Now iterating through files in /data/dog_images/train/068.Flat-coated_retriever
Now iterating through files in /data/dog_images/train/089.Irish_wolfhound
Now iterating through files in /data/dog_images/train/104.Miniature_schnauzer
Now iterating through files in /data/dog_images/train/035.Boykin_spaniel
Now iterating through files in /data/dog_images/train/080.Greater_swiss_mountain_dog
Now iterating through files in /data/dog_images/train/007.American_foxhound
Now iterating through files in /data/dog_images/train/112.Nova_scotia_duck_tolling_retriever
Now iterating through files in /data/dog_images/train/025.Black_and_tan_coonhound
Now iterating through files in /data/dog_images/train/072.German_shorthaired_pointer
Now iterating through files in /data/dog_images/train/062.English_setter
Now iterating through files in /data/dog_images/train/029.Border_collie
Now iterating through files in /data/dog_images/train/045.Cardigan_welsh_corgi
Now iterating through files in /data/dog_images/train/105.Neapolitan_mastiff
Now iterating through files in /data/dog_images/train/076.Golden_retriever
Now iterating through files in /data/dog_images/train/063.English_springer_spaniel
Now iterating through files in /data/dog_images/train/078.Great_dane
Now iterating through files in /data/dog_images/train/084.Icelandic_sheepdog
Now iterating through files in /data/dog_images/train/023.Bernese_mountain_dog
Now iterating through files in /data/dog_images/train/091.Japanese_chin
Now iterating through files in /data/dog_images/train/011.Australian_cattle_dog
Now iterating through files in /data/dog_images/train/021.Belgian_sheepdog
Now iterating through files in /data/dog_images/train/041.Bullmastiff
Now iterating through files in /data/dog_images/train/098.Leonberger
Now iterating through files in /data/dog_images/train/018.Beauceron
Now iterating through files in /data/dog_images/train/020.Belgian_malinois
Now iterating through files in /data/dog_images/train/016.Beagle
Now iterating through files in /data/dog_images/train/039.Bull_terrier
Now iterating through files in /data/dog_images/train/087.Irish_terrier
Now iterating through files in /data/dog_images/train/064.English_toy_spaniel
Now iterating through files in /data/dog_images/train/123.Pomeranian
Now iterating through files in /data/dog_images/train/097.Lakeland_terrier
Now iterating through files in /data/dog_images/train/127.Silky_terrier
Now iterating through files in /data/dog_images/train/120.Pharaoh_hound
Now iterating through files in /data/dog_images/train/006.American_eskimo_dog
Now iterating through files in /data/dog_images/train/012.Australian_shepherd
Now iterating through files in /data/dog_images/train/070.German_pinscher
Now iterating through files in /data/dog_images/train/095.Kuvasz
Now iterating through files in /data/dog_images/train/131.Wirehaired_pointing_griffon
Now iterating through files in /data/dog_images/train/125.Portuguese_water_dog
Now iterating through files in /data/dog_images/train/071.German_shepherd_dog
Now iterating through files in /data/dog_images/train/003.Airedale_terrier
```

```
Now iterating through files in /data/dog_images/train/043.Canaan_dog
Now iterating through files in /data/dog_images/train/118.Pembroke_welsh_corgi
Now iterating through files in /data/dog_images/train/010.Anatolian_shepherd_dog
Now iterating through files in /data/dog_images/train/033.Bouvier_des_flandres
Now iterating through files in /data/dog_images/train/106.Newfoundland
Now iterating through files in /data/dog_images/train/047.Chesapeake_bay_retriever
Now iterating through files in /data/dog_images/train/009.American_water_spaniel
Now iterating through files in /data/dog_images/train/065.Entlebucher_mountain_dog
Now iterating through files in /data/dog_images/train/002.Afghan_hound
Now iterating through files in /data/dog_images/train/054.Collie
Now iterating through files in /data/dog_images/train/093.Kerry_blue_terrier
Now iterating through files in /data/dog_images/train/061.English_cocker_spaniel
Now iterating through files in /data/dog_images/train/082.Havanese
Now iterating through files in /data/dog_images/train/044.Cane_corso
Now iterating through files in /data/dog_images/train/056.Dachshund
Now iterating through files in /data/dog_images/train/026.Black_russian_terrier
Now iterating through files in /data/dog_images/train/132.Xoloitzcuintli
Now iterating through files in /data/dog_images/train/094.Komondor
Now iterating through files in /data/dog_images/train/022.Belgian_tervuren
Now iterating through files in /data/dog_images/train/114.Otterhound
Now iterating through files in /data/dog_images/train/036.Briard
Now iterating through files in /data/dog_images/train/074.Giant_schnauzer
Now iterating through files in /data/dog_images/train/017.Bearded_collie
Now iterating through files in /data/dog_images/train/110.Norwegian_lundehund
Now iterating through files in /data/dog_images/train/085.Irish_red_and_white_setter
Now iterating through files in /data/dog_images/train/069.French_bulldog
Now iterating through files in /data/dog_images/train/075.Glen_of_imaal_terrier
Now iterating through files in /data/dog_images/train/042.Cairn_terrier
Now iterating through files in /data/dog_images/train/004.Akita
Now iterating through files in /data/dog_images/train/060.Dogue_de_bordeaux
Now iterating through files in /data/dog_images/train/128.Smooth_fox_terrier
Now iterating through files in /data/dog_images/train/083.Ibizan_hound
Now iterating through files in /data/dog_images/train/117.Pekingese
Now iterating through files in /data/dog_images/train/081.Greyhound
Now iterating through files in /data/dog_images/train/051.Chow_chow
Now iterating through files in /data/dog_images/train/040.Bulldog
Now iterating through files in /data/dog_images/train/008.American_staffordshire_terrier
Now iterating through files in /data/dog_images/train/046.Cavalier_king_charles_spaniel
Now iterating through files in /data/dog_images/train/099.Lhasa_apso
Now iterating through files in /data/dog_images/train/090.Italian_greyhound
Now iterating through files in /data/dog_images/train/050.Chinese_shar-pei
Now iterating through files in /data/dog_images/train/086.Irish_setter
Now iterating through files in /data/dog_images/train/037.Brittany
Now iterating through files in /data/dog_images/train/121.Plott
Now iterating through files in /data/dog_images/train/014.Basenji
Now iterating through files in /data/dog_images/train/030.Border_terrier
Now iterating through files in /data/dog_images/train/079.Great_pyrenees
Now iterating through files in /data/dog_images/train/096.Labrador_retriever
```

```
Now iterating through files in /data/dog_images/train/027.Bloodhound
Now iterating through files in /data/dog_images/train/048.Chihuahua
Now iterating through files in /data/dog_images/train/119.Petit_basset_griffon_vendeen
Now iterating through files in /data/dog_images/train/124.Poodle
Now iterating through files in /data/dog_images/train/058.Dandie_dinmont_terrier
Now iterating through files in /data/dog_images/train/052.Clumber_spaniel
Now iterating through files in /data/dog_images/train/038.Brussels_griffon
Now iterating through files in /data/dog_images/train/113.Old_english_sheepdog
Now iterating through files in /data/dog_images/train/057.Dalmatian
Now iterating through files in /data/dog_images/train/053.Cocker_spaniel
Now iterating through files in /data/dog_images/train/122.Pointer
Now iterating through files in /data/dog_images/train/077.Gordon_setter
Now iterating through files in /data/dog_images/train/073.German_wirehaired_pointer
Now iterating through files in /data/dog_images/train/088.Irish_water_spaniel
Now iterating through files in /data/dog_images/train/111.Norwich_terrier
Now iterating through files in /data/dog_images/train/109.Norwegian_elkhound
Now iterating through files in /data/dog_images/train/001.Affenpinscher
Now iterating through files in /data/dog_images/train/015.Basset_hound
Now iterating through files in /data/dog_images/train/101.Maltese
Now iterating through files in /data/dog_images/train/092.Keeshond
Now iterating through files in /data/dog_images/train/100.Lowchen
Now iterating through files in /data/dog_images/test/103.Mastiff
Now iterating through files in /data/dog_images/test/059.Doberman_pinscher
Now iterating through files in /data/dog_images/test/055.Curly-coated_retriever
Now iterating through files in /data/dog_images/test/031.Borzoi
Now iterating through files in /data/dog_images/test/024.Bichon_frise
Now iterating through files in /data/dog_images/test/049.Chinese_crested
Now iterating through files in /data/dog_images/test/067.Finnish_spitz
Now iterating through files in /data/dog_images/test/130.Welsh_springer_spaniel
Now iterating through files in /data/dog_images/test/019.Bedlington_terrier
Now iterating through files in /data/dog_images/test/115.Papillon
Now iterating through files in /data/dog_images/test/126.Saint_bernard
Now iterating through files in /data/dog_images/test/013.Australian_terrier
Now iterating through files in /data/dog_images/test/116.Parson_russell_terrier
Now iterating through files in /data/dog_images/test/107.Norfolk_terrier
Now iterating through files in /data/dog_images/test/133.Yorkshire_terrier
Now iterating through files in /data/dog_images/test/032.Boston_terrier
Now iterating through files in /data/dog_images/test/108.Norwegian_buhund
Now iterating through files in /data/dog_images/test/028.Bluetick_coonhound
Now iterating through files in /data/dog_images/test/066.Field_spaniel
Now iterating through files in /data/dog_images/test/129.Tibetan_mastiff
Now iterating through files in /data/dog_images/test/005.Alaskan_malamute
Now iterating through files in /data/dog_images/test/102.Manchester_terrier
Now iterating through files in /data/dog_images/test/034.Boxer
Now iterating through files in /data/dog_images/test/068.Flat-coated_retriever
Now iterating through files in /data/dog_images/test/089.Irish_wolfhound
Now iterating through files in /data/dog_images/test/104.Miniature_schnauzer
Now iterating through files in /data/dog_images/test/035.Boykin_spaniel
```

```
Now iterating through files in /data/dog_images/test/080.Greater_swiss_mountain_dog
Now iterating through files in /data/dog_images/test/007.American_foxhound
Now iterating through files in /data/dog_images/test/112.Nova_scotia_duck_tolling_retriever
Now iterating through files in /data/dog_images/test/025.Black_and_tan_coonhound
Now iterating through files in /data/dog_images/test/072.German_shorthaired_pointer
Now iterating through files in /data/dog_images/test/062.English_setter
Now iterating through files in /data/dog_images/test/029.Border_collie
Now iterating through files in /data/dog_images/test/045.Cardigan_welsh_corgi
Now iterating through files in /data/dog_images/test/105.Neapolitan_mastiff
Now iterating through files in /data/dog_images/test/076.Golden_retriever
Now iterating through files in /data/dog_images/test/063.English_springer_spaniel
Now iterating through files in /data/dog_images/test/078.Great_dane
Now iterating through files in /data/dog_images/test/084.Icelandic_sheepdog
Now iterating through files in /data/dog_images/test/023.Bernese_mountain_dog
Now iterating through files in /data/dog_images/test/091.Japanese_chin
Now iterating through files in /data/dog_images/test/011.Australian_cattle_dog
Now iterating through files in /data/dog_images/test/021.Belgian_sheepdog
Now iterating through files in /data/dog_images/test/041.Bullmastiff
Now iterating through files in /data/dog_images/test/098.Leonberger
Now iterating through files in /data/dog_images/test/018.Beauceron
Now iterating through files in /data/dog_images/test/020.Belgian_malinois
Now iterating through files in /data/dog_images/test/016.Beagle
Now iterating through files in /data/dog_images/test/039.Bull_terrier
Now iterating through files in /data/dog_images/test/087.Irish_terrier
Now iterating through files in /data/dog_images/test/064.English_toy_spaniel
Now iterating through files in /data/dog_images/test/123.Pomeranian
Now iterating through files in /data/dog_images/test/097.Lakeland_terrier
Now iterating through files in /data/dog_images/test/127.Silky_terrier
Now iterating through files in /data/dog_images/test/120.Pharaoh_hound
Now iterating through files in /data/dog_images/test/006.American_eskimo_dog
Now iterating through files in /data/dog_images/test/012.Australian_shepherd
Now iterating through files in /data/dog_images/test/070.German_pinscher
Now iterating through files in /data/dog_images/test/095.Kuvasz
Now iterating through files in /data/dog_images/test/131.Wirehaired_pointing_griffon
Now iterating through files in /data/dog_images/test/125.Portuguese_water_dog
Now iterating through files in /data/dog_images/test/071.German_shepherd_dog
Now iterating through files in /data/dog_images/test/003.Airedale_terrier
Now iterating through files in /data/dog_images/test/043.Canaan_dog
Now iterating through files in /data/dog_images/test/118.Pembroke_welsh_corgi
Now iterating through files in /data/dog_images/test/010.Anatolian_shepherd_dog
Now iterating through files in /data/dog_images/test/033.Bouvier_des_flandres
Now iterating through files in /data/dog_images/test/106.Newfoundland
Now iterating through files in /data/dog_images/test/047.Chesapeake_bay_retriever
Now iterating through files in /data/dog_images/test/009.American_water_spaniel
Now iterating through files in /data/dog_images/test/065.Entlebucher_mountain_dog
Now iterating through files in /data/dog_images/test/002.Afghan_hound
Now iterating through files in /data/dog_images/test/054.Collie
Now iterating through files in /data/dog_images/test/093.Kerry_blue_terrier
```

```
Now iterating through files in /data/dog_images/test/061.English_cocker_spaniel
Now iterating through files in /data/dog_images/test/082.Havanese
Now iterating through files in /data/dog_images/test/044.Cane_corso
Now iterating through files in /data/dog_images/test/056.Dachshund
Now iterating through files in /data/dog_images/test/026.Black_russian_terrier
Now iterating through files in /data/dog_images/test/132.Xoloitzcuintli
Now iterating through files in /data/dog_images/test/094.Komondor
Now iterating through files in /data/dog_images/test/022.Belgian_tervuren
Now iterating through files in /data/dog_images/test/114.Otterhound
Now iterating through files in /data/dog_images/test/036.Briard
Now iterating through files in /data/dog_images/test/074.Giant_schnauzer
Now iterating through files in /data/dog_images/test/017.Bearded_collie
Now iterating through files in /data/dog_images/test/110.Norwegian_lundehund
Now iterating through files in /data/dog_images/test/085.Irish_red_and_white_setter
Now iterating through files in /data/dog_images/test/069.French_bulldog
Now iterating through files in /data/dog_images/test/075.Glen_of_imaal_terrier
Now iterating through files in /data/dog_images/test/042.Cairn_terrier
Now iterating through files in /data/dog_images/test/004.Akita
Now iterating through files in /data/dog_images/test/060.Dogue_de_bordeaux
Now iterating through files in /data/dog_images/test/128.Smooth_fox_terrier
Now iterating through files in /data/dog_images/test/083.Ibizan_hound
Now iterating through files in /data/dog_images/test/117.Pekingese
Now iterating through files in /data/dog_images/test/081.Greyhound
Now iterating through files in /data/dog_images/test/051.Chow_chow
Now iterating through files in /data/dog_images/test/040.Bulldog
Now iterating through files in /data/dog_images/test/008.American_staffordshire_terrier
Now iterating through files in /data/dog_images/test/046.Cavalier_king_charles_spaniel
Now iterating through files in /data/dog_images/test/099.Lhasa_apso
Now iterating through files in /data/dog_images/test/090.Italian_greyhound
Now iterating through files in /data/dog_images/test/050.Chinese_shar-pei
Now iterating through files in /data/dog_images/test/086.Irish_setter
Now iterating through files in /data/dog_images/test/037.Brittany
Now iterating through files in /data/dog_images/test/121.Plott
Now iterating through files in /data/dog_images/test/014.Basenji
Now iterating through files in /data/dog_images/test/030.Border_terrier
Now iterating through files in /data/dog_images/test/079.Great_pyrenees
Now iterating through files in /data/dog_images/test/096.Labrador_retriever
Now iterating through files in /data/dog_images/test/027.Bloodhound
Now iterating through files in /data/dog_images/test/048.Chihuahua
Now iterating through files in /data/dog_images/test/119.Petit_basset_griffon_vendeen
Now iterating through files in /data/dog_images/test/124.Poodle
Now iterating through files in /data/dog_images/test/058.Dandie_dinmont_terrier
Now iterating through files in /data/dog_images/test/052.Clumber_spaniel
Now iterating through files in /data/dog_images/test/038.Brussels_griffon
Now iterating through files in /data/dog_images/test/113.Old_english_sheepdog
Now iterating through files in /data/dog_images/test/057.Dalmatian
Now iterating through files in /data/dog_images/test/053.Cocker_spaniel
Now iterating through files in /data/dog_images/test/122.Pointer
```

```
Now iterating through files in /data/dog_images/test/077.Gordon_setter
Now iterating through files in /data/dog_images/test/073.German_wirehaired_pointer
Now iterating through files in /data/dog_images/test/088.Irish_water_spaniel
Now iterating through files in /data/dog_images/test/111.Norwich_terrier
Now iterating through files in /data/dog_images/test/109.Norwegian_elkhound
Now iterating through files in /data/dog_images/test/001.Affenpinscher
Now iterating through files in /data/dog_images/test/015.Basset_hound
Now iterating through files in /data/dog_images/test/101.Maltese
Now iterating through files in /data/dog_images/test/092.Keeshond
Now iterating through files in /data/dog_images/test/100.Lowchen
Now iterating through files in /data/dog_images/valid/103.Mastiff
Now iterating through files in /data/dog_images/valid/059.Doberman_pinscher
Now iterating through files in /data/dog_images/valid/055.Curly-coated_retriever
Now iterating through files in /data/dog_images/valid/031.Borzoi
Now iterating through files in /data/dog_images/valid/024.Bichon_frise
Now iterating through files in /data/dog_images/valid/049.Chinese_crested
Now iterating through files in /data/dog_images/valid/067.Finnish_spitz
Now iterating through files in /data/dog_images/valid/130.Welsh_springer_spaniel
Now iterating through files in /data/dog_images/valid/019.Bedlington_terrier
Now iterating through files in /data/dog_images/valid/115.Papillon
Now iterating through files in /data/dog_images/valid/126.Saint_bernard
Now iterating through files in /data/dog_images/valid/013.Australian_terrier
Now iterating through files in /data/dog_images/valid/116.Parson_russell_terrier
Now iterating through files in /data/dog_images/valid/107.Norfolk_terrier
Now iterating through files in /data/dog_images/valid/133.Yorkshire_terrier
Now iterating through files in /data/dog_images/valid/032.Boston_terrier
Now iterating through files in /data/dog_images/valid/108.Norwegian_buhund
Now iterating through files in /data/dog_images/valid/028.Bluetick_coonhound
Now iterating through files in /data/dog_images/valid/066.Field_spaniel
Now iterating through files in /data/dog_images/valid/129.Tibetan_mastiff
Now iterating through files in /data/dog_images/valid/005.Alaskan_malamute
Now iterating through files in /data/dog_images/valid/102.Manchester_terrier
Now iterating through files in /data/dog_images/valid/034.Boxer
Now iterating through files in /data/dog_images/valid/068.Flat-coated_retriever
Now iterating through files in /data/dog_images/valid/089.Irish_wolfhound
Now iterating through files in /data/dog_images/valid/104.Miniature_schnauzer
Now iterating through files in /data/dog_images/valid/035.Boykin_spaniel
Now iterating through files in /data/dog_images/valid/080.Greater_swiss_mountain_dog
Now iterating through files in /data/dog_images/valid/007.American_foxhound
Now iterating through files in /data/dog_images/valid/112.Nova_scotia_duck_tolling_retriever
Now iterating through files in /data/dog_images/valid/025.Black_and_tan_coonhound
Now iterating through files in /data/dog_images/valid/072.German_shorthaired_pointer
Now iterating through files in /data/dog_images/valid/062.English_setter
Now iterating through files in /data/dog_images/valid/029.Border_collie
Now iterating through files in /data/dog_images/valid/045.Cardigan_welsh_corgi
Now iterating through files in /data/dog_images/valid/105.Neapolitan_mastiff
Now iterating through files in /data/dog_images/valid/076.Golden_retriever
Now iterating through files in /data/dog_images/valid/063.English_springer_spaniel
```

```
Now iterating through files in /data/dog_images/valid/078.Great_dane
Now iterating through files in /data/dog_images/valid/084.Icelandic_sheepdog
Now iterating through files in /data/dog_images/valid/023.Bernese_mountain_dog
Now iterating through files in /data/dog_images/valid/091.Japanese_chin
Now iterating through files in /data/dog_images/valid/011.Australian_cattle_dog
Now iterating through files in /data/dog_images/valid/021.Belgian_sheepdog
Now iterating through files in /data/dog_images/valid/041.Bullmastiff
Now iterating through files in /data/dog_images/valid/098.Leonberger
Now iterating through files in /data/dog_images/valid/018.Beauceron
Now iterating through files in /data/dog_images/valid/020.Belgian_malinois
Now iterating through files in /data/dog_images/valid/016.Beagle
Now iterating through files in /data/dog_images/valid/039.Bull_terrier
Now iterating through files in /data/dog_images/valid/087.Irish_terrier
Now iterating through files in /data/dog_images/valid/064.English_toy_spaniel
Now iterating through files in /data/dog_images/valid/123.Pomeranian
Now iterating through files in /data/dog_images/valid/097.Lakeland_terrier
Now iterating through files in /data/dog_images/valid/127.Silky_terrier
Now iterating through files in /data/dog_images/valid/120.Pharaoh_hound
Now iterating through files in /data/dog_images/valid/006.American_eskimo_dog
Now iterating through files in /data/dog_images/valid/012.Australian_shepherd
Now iterating through files in /data/dog_images/valid/070.German_pinscher
Now iterating through files in /data/dog_images/valid/095.Kuvasz
Now iterating through files in /data/dog_images/valid/131.Wirehaired_pointing_griffon
Now iterating through files in /data/dog_images/valid/125.Portuguese_water_dog
Now iterating through files in /data/dog_images/valid/071.German_shepherd_dog
Now iterating through files in /data/dog_images/valid/003.Airedale_terrier
Now iterating through files in /data/dog_images/valid/043.Canaan_dog
Now iterating through files in /data/dog_images/valid/118.Pembroke_welsh_corgi
Now iterating through files in /data/dog_images/valid/010.Anatolian_shepherd_dog
Now iterating through files in /data/dog_images/valid/033.Bouvier_des_flandres
Now iterating through files in /data/dog_images/valid/106.Newfoundland
Now iterating through files in /data/dog_images/valid/047.Chesapeake_bay_retriever
Now iterating through files in /data/dog_images/valid/009.American_water_spaniel
Now iterating through files in /data/dog_images/valid/065.Entlebucher_mountain_dog
Now iterating through files in /data/dog_images/valid/002.Afghan_hound
Now iterating through files in /data/dog_images/valid/054.Collie
Now iterating through files in /data/dog_images/valid/093.Kerry_blue_terrier
Now iterating through files in /data/dog_images/valid/061.English_cocker_spaniel
Now iterating through files in /data/dog_images/valid/082.Havanese
Now iterating through files in /data/dog_images/valid/044.Cane_corso
Now iterating through files in /data/dog_images/valid/056.Dachshund
Now iterating through files in /data/dog_images/valid/026.Black_russian_terrier
Now iterating through files in /data/dog_images/valid/132.Xoloitzcuintli
Now iterating through files in /data/dog_images/valid/094.Komondor
Now iterating through files in /data/dog_images/valid/022.Belgian_tervuren
Now iterating through files in /data/dog_images/valid/114.Otterhound
Now iterating through files in /data/dog_images/valid/036.Briard
Now iterating through files in /data/dog_images/valid/074.Giant_schnauzer
```

```
Now iterating through files in /data/dog_images/valid/017.Bearded_collie
Now iterating through files in /data/dog_images/valid/110.Norwegian_lundehund
Now iterating through files in /data/dog_images/valid/085.Irish_red_and_white_setter
Now iterating through files in /data/dog_images/valid/069.French_bulldog
Now iterating through files in /data/dog_images/valid/075.Glen_of_imaal_terrier
Now iterating through files in /data/dog_images/valid/042.Cairn_terrier
Now iterating through files in /data/dog_images/valid/004.Akita
Now iterating through files in /data/dog_images/valid/060.Dogue_de_bordeaux
Now iterating through files in /data/dog_images/valid/128.Smooth_fox_terrier
Now iterating through files in /data/dog_images/valid/083.Ibizan_hound
Now iterating through files in /data/dog_images/valid/117.Pekingese
Now iterating through files in /data/dog_images/valid/081.Greyhound
Now iterating through files in /data/dog_images/valid/051.Chow_chow
Now iterating through files in /data/dog_images/valid/040.Bulldog
Now iterating through files in /data/dog_images/valid/008.American_staffordshire_terrier
Now iterating through files in /data/dog_images/valid/046.Cavalier_king_charles_spaniel
Now iterating through files in /data/dog_images/valid/099.Lhasa_apso
Now iterating through files in /data/dog_images/valid/090.Italian_greyhound
Now iterating through files in /data/dog_images/valid/050.Chinese_shar-pei
Now iterating through files in /data/dog_images/valid/086.Irish_setter
Now iterating through files in /data/dog_images/valid/037.Brittany
Now iterating through files in /data/dog_images/valid/121.Plott
Now iterating through files in /data/dog_images/valid/014.Basenji
Now iterating through files in /data/dog_images/valid/030.Border_terrier
Now iterating through files in /data/dog_images/valid/079.Great_pyrenees
Now iterating through files in /data/dog_images/valid/096.Labrador_retriever
Now iterating through files in /data/dog_images/valid/027.Bloodhound
Now iterating through files in /data/dog_images/valid/048.Chihuahua
Now iterating through files in /data/dog_images/valid/119.Petit_basset_griffon_vendeen
Now iterating through files in /data/dog_images/valid/124.Poodle
Now iterating through files in /data/dog_images/valid/058.Dandie_dinmont_terrier
Now iterating through files in /data/dog_images/valid/052.Clumber_spaniel
Now iterating through files in /data/dog_images/valid/038.Brussels_griffon
Now iterating through files in /data/dog_images/valid/113.Old_english_sheepdog
Now iterating through files in /data/dog_images/valid/057.Dalmatian
Now iterating through files in /data/dog_images/valid/053.Cocker_spaniel
Now iterating through files in /data/dog_images/valid/122.Pointer
Now iterating through files in /data/dog_images/valid/077.Gordon_setter
Now iterating through files in /data/dog_images/valid/073.German_wirehaired_pointer
Now iterating through files in /data/dog_images/valid/088.Irish_water_spaniel
Now iterating through files in /data/dog_images/valid/111.Norwich_terrier
Now iterating through files in /data/dog_images/valid/109.Norwegian_elkhound
Now iterating through files in /data/dog_images/valid/001.Affenpinscher
Now iterating through files in /data/dog_images/valid/015.Basset_hound
Now iterating through files in /data/dog_images/valid/101.Maltese
Now iterating through files in /data/dog_images/valid/092.Keeshond
Now iterating through files in /data/dog_images/valid/100.Lowchen
```

```
In [11]: print('Smallest image dimension is', image_size_dict[min(image_size_dict, key=image_siz
```

```
Smallest image dimension is 105 pixels
```

```
In [12]: #Gather classes and store in dictionary
         classes_dict = {}

         breed_list = os.listdir('/data/dog_images/train')
         breed_list_len = len(breed_list)

         for i in range(breed_list_len):
             classes_dict[i] = breed_list[i]

         print(classes_dict)
```

```
{0: '103.Mastiff', 1: '059.Doberman_pinscher', 2: '055.Curly-coated_retriever', 3: '031.Borzoi',
```

```
In [13]: print(f'There are {len(classes_dict.keys())} dog breeds in the data set.')
```

```
There are 133 dog breeds in the data set.
```

```
In [ ]:
```

```
In [ ]:
```

```
In [14]: import os
         from torchvision import datasets
         import torchvision.transforms as transforms
         import torchvision.datasets as datasets
         from torch.utils.data.sampler import SubsetRandomSampler

         ### TODO: Write data loaders for training, validation, and test sets
         ## Specify appropriate transforms, and batch_sizes

         ### CUDA check
         train_on_gpu = torch.cuda.is_available()

         if not train_on_gpu:
             print('CUDA is not available. Training on CPU...')
         else:
             print('CUDA is available. Training on GPU...')

         ### Data Loading
         # number of subprocesses to use for data loading
         num_workers = 0
         # how many samples per batch to load
```

```
batch_size = 64

train_transform = transforms.Compose([
    transforms.Resize(size=256),
    transforms.CenterCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
])

valid_test_transform = transforms.Compose([
    transforms.Resize(size=256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
])


train_dataset = datasets.ImageFolder("/data/dog_images/train", transform=train_transfor
valid_dataset = datasets.ImageFolder("/data/dog_images/valid", transform=valid_test_tra
test_dataset = datasets.ImageFolder("/data/dog_images/test", transform=valid_test_trans

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
    num_workers=num_workers, shuffle=True)
valid_loader = torch.utils.data.DataLoader(valid_dataset, batch_size=batch_size,
    num_workers=num_workers, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size,
    num_workers=num_workers)
```

```
CUDA is available. Training on GPU...
```

Question 3: Describe your chosen procedure for preprocessing the data. - How does your code resize the images (by cropping, stretching, etc)? What size did you pick for the input tensor, and why? - Did you decide to augment the dataset? If so, how (through translations, flips, rotations, etc)? If not, why not?

Answer: The code performs a RandomResizedCrop. The smallest image dimension encountered in the data set is 105. I didn't want to crop all images below this value since key image info may be lost. So I cropped to 224 x 224 (same as VGG). I also did a random flip and random rotation in the datasets.

1.1.8 (IMPLEMENTATION) Model Architecture

Create a CNN to classify dog breed. Use the template in the code cell below.

```
In [15]: # Assess dimensions of each potential layer, in order to determine input dim in final n
         def conv_out_size(in_vol_size, in_depth, num_filters, filter_size, stride, padding, poo
             '''MY FUNCTION!
```

```
        pool_params = [pool_filter, pool_stride] | for MAX POOL
        in_vol_size = dim of one side of input (need symmetric input)
        filter_size = size of filter
        num_filters = output depth
        pytorch conv2d ref = nn.Conv2d(in_channels, out_channels, kernel_size, stride=1,
    '''

    out_size = (in_vol_size-filter_size+2*padding)/stride+1
    out_dims = (out_size, out_size, num_filters)
    out_params = (filter_size * filter_size * num_filters * in_depth) + num_filters

    if pool == True:
        out_size = ((out_size - pool_filter) / pool_stride) + 1
        out_dims = (out_size, out_size, num_filters)
        return out_dims, out_params, f'Output Size: {out_dims}, Output Parameters: {out

    else:
        return out_dims, out_params, f'Output Size: {out_dims}, Output Parameters: {out

# OPTION 1
# Input impage size = 224,224,3
# self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
# self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
# self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
# self.conv4 = nn.Conv2d(128, 256, 3, padding=1)
# self.conv5 = nn.Conv2d(256, 512, 3, padding=1)
# nn.MaxPool2d(2, 2)

# OPTION 2
# Input impage size = 100,100,3
# self.conv1 = nn.Conv2d(3, 16, 3, padding=1)
# self.conv2 = nn.Conv2d(16, 32, 3, padding=1)
# self.conv3 = nn.Conv2d(32, 64, 3, padding=1)
# self.conv4 = nn.Conv2d(64, 128, 3, padding=1)
# nn.MaxPool2d(2, 2)

dim1, params1, _ = conv_out_size(224, 3, 32, 3, 1, 1, pool=True, pool_filter=2, pool_st
dim2, params2, _ = conv_out_size(dim1[0], 32, 64, 3, 1, 1, pool=True, pool_filter=2, po
dim3, params3, _ = conv_out_size(dim2[0], 64, 128, 3, 1, 1, pool=True, pool_filter=2, p
dim4, params4, _ = conv_out_size(dim3[0], 128, 256, 3, 1, 1, pool=True, pool_filter=2,
# dim5, params5, _ = conv_out_size(dim4[0], 256, 512, 3, 1, 1, pool=True, pool_filter=2
# params = params1 + params2
print(dim1)
print(params1)
print(dim2)
print(params2)
print(dim3)
print(params3)
```

```
        print(dim4)
        print(params4)
        # print(dim5)
        # print(params5)
        # print('Total params',params)
```

```
(112.0, 112.0, 32)
896
(56.0, 56.0, 64)
18496
(28.0, 28.0, 128)
73856
(14.0, 14.0, 256)
295168
```

```
In [16]: import torch.nn as nn
         import torch.nn.functional as F

         # define the CNN architecture
         class Net(nn.Module):
             ### TODO: choose an architecture, and complete the class
             def __init__(self):
                 super(Net, self).__init__()
                 ## Define layers of a CNN
                 self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
                 self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
                 self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
                 self.conv4 = nn.Conv2d(128, 256, 3, padding=1)
         #         self.conv4 = nn.Conv2d(128, 256, 3, padding=1)
         #         self.conv5 = nn.Conv2d(256, 512, 3, padding=1)
                 self.bn1 = nn.BatchNorm2d(32)
                 self.bn2 = nn.BatchNorm2d(64)
                 self.bn3 = nn.BatchNorm2d(128)
                 self.bn4 = nn.BatchNorm2d(256)
         #         self.bn5 = nn.BatchNorm2d(512)
                 self.pool = nn.MaxPool2d(2,2)
                 self.dropout = nn.Dropout(0.2)
                 self.fc1 = nn.Linear(14*14*256, 1024)
         #         self.fc2 = nn.Linear(2048, 1024)
                 self.fc2 = nn.Linear(1024, 512)
                 self.fc3 = nn.Linear(512, 133)

             def forward(self, x):
                 ## Define forward behavior
                 x = self.pool(F.relu(self.bn1(self.conv1(x))))
                 x = self.dropout(x)
                 x = self.pool(F.relu(self.bn2(self.conv2(x))))
```

```
            x = self.dropout(x)
            x = self.pool(F.relu(self.bn3(self.conv3(x))))
            x = self.dropout(x)
            x = self.pool(F.relu(self.bn4(self.conv4(x))))
            x = self.dropout(x)
#             x = self.bn1(self.pool(F.relu(self.conv1(x))))
#             x = self.dropout(x)
#             x = self.bn2(self.pool(F.relu(self.conv2(x))))
#             x = self.dropout(x)
#             x = self.bn3(self.pool(F.relu(self.conv3(x))))
#             x = self.dropout(x)
#             x = self.bn4(self.pool(F.relu(self.conv4(x))))
#             x = self.dropout(x)
#             x = self.bn5(self.pool(F.relu(self.conv5(x))))
#             x = self.dropout(x)
#             print(x.shape)
            x = x.view(-1, 14*14*256)
            x = self.dropout(x)
            x = F.relu(self.fc1(x))
            x = self.dropout(x)
            x = F.relu(self.fc2(x))
            x = F.relu(self.fc3(x))
#             x = self.dropout(x)
#             x = F.relu(self.fc3(x))
            return x

#-#-# You so NOT have to modify the code below this line. #-#-#

# instantiate the CNN
model_scratch = Net()

# move tensors to GPU if CUDA is available
if use_cuda:
    model_scratch.cuda()
```

Question 4: Outline the steps you took to get to your final CNN architecture and your reasoning at each step.

Answer: I based the architecture off of VGG16... decreasing height and width at each layer, but increasing depth by ~ a factor of 2. I determined the dimensions of each layer using a custom function. The last layer is a fully connected layer with 133 classes.

1.1.9 (IMPLEMENTATION) Specify Loss Function and Optimizer

Use the next code cell to specify a loss function and optimizer. Save the chosen loss function as `criterion_scratch`, and the optimizer as `optimizer_scratch` below.

```
In [17]: import torch.optim as optim
```

```
### TODO: select loss function
criterion_scratch = nn.CrossEntropyLoss()

### TODO: select optimizer
optimizer_scratch = optim.SGD(model_scratch.parameters(), lr=0.05)
```

In [18]:
```
#set loaders

loaders_scratch = {
    'train': train_loader,
    'valid': valid_loader,
    'test': test_loader
}
loaders_scratch
```

Out[18]:
```
{'train': <torch.utils.data.dataloader.DataLoader at 0x7f242807d358>,
 'valid': <torch.utils.data.dataloader.DataLoader at 0x7f242807d278>,
 'test': <torch.utils.data.dataloader.DataLoader at 0x7f242807d2b0>}
```

1.1.10 (IMPLEMENTATION) Train and Validate the Model

Train and validate your model in the code cell below. Save the final model parameters at filepath `'model_scratch.pt'`.

In [19]:
```
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

def train(n_epochs, loaders, model, optimizer, criterion, use_cuda, save_path):
    """returns trained model"""
    # initialize tracker for minimum validation loss
    valid_loss_min = np.Inf

    for epoch in range(1, n_epochs+1):
        print('Now in epoch ', epoch, '...')
        # initialize variables to monitor training and validation loss
        train_loss = 0.0
        valid_loss = 0.0

        ###################
        # train the model #
        ###################
        model.train()
        for batch_idx, (data, target) in enumerate(loaders['train']):
            # move to GPU
            if use_cuda:
                data, target = data.cuda(), target.cuda()
            ## find the loss and update the model parameters accordingly
            ## record the average training loss, using something like
            ## train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_lo
```

```
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, target)
            loss.backward()
            optimizer.step()
#             train_loss += loss.item()*data.size(0)
            train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_loss)

        ######################
        # validate the model #
        ######################
        model.eval()
        for batch_idx, (data, target) in enumerate(loaders['valid']):
            # move to GPU
            if use_cuda:
                data, target = data.cuda(), target.cuda()
            ## update the average validation loss



            optimizer.zero_grad()



            with torch.no_grad():
                output = model(data)
            loss = criterion(output, target)
#             valid_loss += loss.item()*data.size(0)
            valid_loss = valid_loss + ((1 / (batch_idx + 1)) * (loss.data - valid_loss)

#         train_loss = train_loss/len(loaders['train'])
#         valid_loss = valid_loss/len(loaders['valid'])

        # print training/validation statistics
        print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.format(
            epoch,
            train_loss,
            valid_loss
            ))

        ## TODO: save the model if validation loss has decreased
        if valid_loss <= valid_loss_min:
            print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.fo
            valid_loss_min,
            valid_loss))
            torch.save(model.state_dict(), 'model_scratch.pt')
            valid_loss_min = valid_loss
```

```
            # return trained model
            return model


        # train the model
        model_scratch = train(25, loaders_scratch, model_scratch, optimizer_scratch,
                              criterion_scratch, use_cuda, 'model_scratch.pt')

        # load the model that got the best validation accuracy
        model_scratch.load_state_dict(torch.load('model_scratch.pt'))
```

```
Now in epoch  1 ...
Epoch: 1         Training Loss: 4.881628         Validation Loss: 4.867357
Validation loss decreased (inf --> 4.867357).  Saving model ...
Now in epoch  2 ...
Epoch: 2         Training Loss: 4.773715         Validation Loss: 4.819668
Validation loss decreased (4.867357 --> 4.819668).  Saving model ...
Now in epoch  3 ...
Epoch: 3         Training Loss: 4.597115         Validation Loss: 4.656600
Validation loss decreased (4.819668 --> 4.656600).  Saving model ...
Now in epoch  4 ...
Epoch: 4         Training Loss: 4.467999         Validation Loss: 4.491435
Validation loss decreased (4.656600 --> 4.491435).  Saving model ...
Now in epoch  5 ...
Epoch: 5         Training Loss: 4.350633         Validation Loss: 4.424088
Validation loss decreased (4.491435 --> 4.424088).  Saving model ...
Now in epoch  6 ...
Epoch: 6         Training Loss: 4.235861         Validation Loss: 4.534289
Now in epoch  7 ...
Epoch: 7         Training Loss: 4.129731         Validation Loss: 4.500051
Now in epoch  8 ...
Epoch: 8         Training Loss: 4.028242         Validation Loss: 4.729065
Now in epoch  9 ...
Epoch: 9         Training Loss: 3.932277         Validation Loss: 4.401161
Validation loss decreased (4.424088 --> 4.401161).  Saving model ...
Now in epoch  10 ...
Epoch: 10         Training Loss: 3.833505         Validation Loss: 4.287504
Validation loss decreased (4.401161 --> 4.287504).  Saving model ...
Now in epoch  11 ...
Epoch: 11         Training Loss: 3.744488         Validation Loss: 4.231480
Validation loss decreased (4.287504 --> 4.231480).  Saving model ...
Now in epoch  12 ...
Epoch: 12         Training Loss: 3.646891         Validation Loss: 4.154488
Validation loss decreased (4.231480 --> 4.154488).  Saving model ...
Now in epoch  13 ...
Epoch: 13         Training Loss: 3.542438         Validation Loss: 4.226670
Now in epoch  14 ...
Epoch: 14         Training Loss: 3.450309         Validation Loss: 4.046222
```

```
Validation loss decreased (4.154488 --> 4.046222).  Saving model ...
Now in epoch  15 ...
Epoch: 15         Training Loss: 3.346071         Validation Loss: 3.970067
Validation loss decreased (4.046222 --> 3.970067).  Saving model ...
Now in epoch  16 ...
Epoch: 16         Training Loss: 3.227172         Validation Loss: 4.034398
Now in epoch  17 ...
Epoch: 17         Training Loss: 3.119082         Validation Loss: 5.180377
Now in epoch  18 ...
Epoch: 18         Training Loss: 3.000857         Validation Loss: 3.959187
Validation loss decreased (3.970067 --> 3.959187).  Saving model ...
Now in epoch  19 ...
Epoch: 19         Training Loss: 2.880043         Validation Loss: 4.264797
Now in epoch  20 ...
Epoch: 20         Training Loss: 2.778096         Validation Loss: 4.121378
Now in epoch  21 ...
Epoch: 21         Training Loss: 2.674355         Validation Loss: 3.861272
Validation loss decreased (3.959187 --> 3.861272).  Saving model ...
Now in epoch  22 ...
Epoch: 22         Training Loss: 2.540817         Validation Loss: 3.970282
Now in epoch  23 ...
Epoch: 23         Training Loss: 2.440238         Validation Loss: 4.353932
Now in epoch  24 ...
Epoch: 24         Training Loss: 2.295078         Validation Loss: 5.102384
Now in epoch  25 ...
Epoch: 25         Training Loss: 2.173931         Validation Loss: 4.013855
```

In [ ]:

1.1.11 (IMPLEMENTATION) Test the Model

Try out your model on the test dataset of dog images. Use the code cell below to calculate and print the test loss and accuracy. Ensure that your test accuracy is greater than 10%.

In [20]:
```
def test(loaders, model, criterion, use_cuda):

    # monitor test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    model.eval()
    for batch_idx, (data, target) in enumerate(loaders['test']):
        # move to GPU
        if use_cuda:
            data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the model
```

```
            output = model(data)
            # calculate the loss
            loss = criterion(output, target)
            # update average test loss
            test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
            # convert output probabilities to predicted class
            pred = output.data.max(1, keepdim=True)[1]
            # compare predictions to true label
            correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
            total += data.size(0)

        print('Test Loss: {:.6f}\n'.format(test_loss))

        print('\nTest Accuracy: %2d%% (%2d/%2d)' % (
            100. * correct / total, correct, total))

    # call test function
    test(loaders_scratch, model_scratch, criterion_scratch, use_cuda)
```

```
Test Loss: 3.913070


Test Accuracy: 14% (119/836)
```

Step 4: Create a CNN to Classify Dog Breeds (using Transfer Learning)

You will now use transfer learning to create a CNN that can identify dog breed from images. Your CNN must attain at least 60% accuracy on the test set.

1.1.12 (IMPLEMENTATION) Specify Data Loaders for the Dog Dataset

Use the code cell below to write three separate data loaders for the training, validation, and test datasets of dog images (located at `dogImages/train`, `dogImages/valid`, and `dogImages/test`, respectively).

If you like, **you are welcome to use the same data loaders from the previous step**, when you created a CNN from scratch.

```
In [ ]:
```

```
In [122]: ## TODO: Specify data loaders

    ## Specify appropriate transforms, and batch_sizes

    ### CUDA check
    train_on_gpu = torch.cuda.is_available()

    if not train_on_gpu:
```

```
        print('CUDA is not available. Training on CPU...')
    else:
        print('CUDA is available. Training on GPU...')

    ### Data Loading
    # number of subprocesses to use for data loading
    num_workers = 0
    # how many samples per batch to load
    batch_size = 64

    train_transform = transforms.Compose([
        transforms.Resize(size=320),
        transforms.CenterCrop(299),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(20),
        transforms.ToTensor(),
        transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
    ])

    valid_test_transform = transforms.Compose([
        transforms.Resize(size=320),
        transforms.CenterCrop(299),
        transforms.ToTensor(),
        transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
    ])


    train_dataset = datasets.ImageFolder("/data/dog_images/train", transform=train_transfo
    valid_dataset = datasets.ImageFolder("/data/dog_images/valid", transform=valid_test_tr
    test_dataset = datasets.ImageFolder("/data/dog_images/test", transform=valid_test_tran

    train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size,
        num_workers=num_workers, shuffle=True)
    valid_loader = torch.utils.data.DataLoader(valid_dataset, batch_size=batch_size,
        num_workers=num_workers, shuffle=True)
    test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size,
        num_workers=num_workers)
```

```
CUDA is available. Training on GPU...
```

1.1.13 (IMPLEMENTATION) Model Architecture

Use transfer learning to create a CNN to classify dog breed. Use the code cell below, and save your initialized model as the variable `model_transfer`.

```
In [123]: import torchvision.models as models
          import torch.nn as nn
```

```
        ## TODO: Specify model architecture
        # Load the pretrained model from pytorch
        # model_transfer = models.vgg16(pretrained=True)
        model_transfer = models.inception_v3(pretrained=True)

        # print out the model structure
        print(model_transfer)

        # print(model_transfer.classifier[6].out_features)
```

```
Inception3(
  (Conv2d_1a_3x3): BasicConv2d(
    (conv): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_2a_3x3): BasicConv2d(
    (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_2b_3x3): BasicConv2d(
    (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_3b_1x1): BasicConv2d(
    (conv): Conv2d(64, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(80, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Conv2d_4a_3x3): BasicConv2d(
    (conv): Conv2d(80, 192, kernel_size=(3, 3), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (Mixed_5b): InceptionA(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch5x5_1): BasicConv2d(
      (conv): Conv2d(192, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(48, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch5x5_2): BasicConv2d(
      (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
```

```
  )
  (branch3x3dbl_2): BasicConv2d(
    (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_3): BasicConv2d(
    (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch_pool): BasicConv2d(
    (conv): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(Mixed_5c): InceptionA(
  (branch1x1): BasicConv2d(
    (conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch5x5_1): BasicConv2d(
    (conv): Conv2d(256, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(48, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch5x5_2): BasicConv2d(
    (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_1): BasicConv2d(
    (conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_2): BasicConv2d(
    (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_3): BasicConv2d(
    (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch_pool): BasicConv2d(
    (conv): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(Mixed_5d): InceptionA(
  (branch1x1): BasicConv2d(
    (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
```

```
    )
    (branch5x5_1): BasicConv2d(
      (conv): Conv2d(288, 48, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(48, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch5x5_2): BasicConv2d(
      (conv): Conv2d(48, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_3): BasicConv2d(
      (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (Mixed_6a): InceptionB(
    (branch3x3): BasicConv2d(
      (conv): Conv2d(288, 384, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(288, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_3): BasicConv2d(
      (conv): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (Mixed_6b): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
```

```
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(128, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(128, 128, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(128, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (Mixed_6c): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_2): BasicConv2d(
```

```
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (Mixed_6d): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
```

```
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_2): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(160, 160, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(160, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (Mixed_6e): InceptionC(
    (branch1x1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_2): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7_3): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_2): BasicConv2d(
```

```
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_3): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_4): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7dbl_5): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (AuxLogits): InceptionAux(
    (conv0): BasicConv2d(
      (conv): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (conv1): BasicConv2d(
      (conv): Conv2d(128, 768, kernel_size=(5, 5), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(768, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (fc): Linear(in_features=768, out_features=1000, bias=True)
  )
  (Mixed_7a): InceptionD(
    (branch3x3_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3_2): BasicConv2d(
      (conv): Conv2d(192, 320, kernel_size=(3, 3), stride=(2, 2), bias=False)
      (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7x3_1): BasicConv2d(
      (conv): Conv2d(768, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch7x7x3_2): BasicConv2d(
      (conv): Conv2d(192, 192, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
```

```
  (branch7x7x3_3): BasicConv2d(
    (conv): Conv2d(192, 192, kernel_size=(7, 1), stride=(1, 1), padding=(3, 0), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch7x7x3_4): BasicConv2d(
    (conv): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(Mixed_7b): InceptionE(
  (branch1x1): BasicConv2d(
    (conv): Conv2d(1280, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3_1): BasicConv2d(
    (conv): Conv2d(1280, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3_2a): BasicConv2d(
    (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=False)
    (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3_2b): BasicConv2d(
    (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=False)
    (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_1): BasicConv2d(
    (conv): Conv2d(1280, 448, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(448, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_2): BasicConv2d(
    (conv): Conv2d(448, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_3a): BasicConv2d(
    (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=False)
    (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch3x3dbl_3b): BasicConv2d(
    (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=False)
    (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (branch_pool): BasicConv2d(
    (conv): Conv2d(1280, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(Mixed_7c): InceptionE(
```

```
    (branch1x1): BasicConv2d(
      (conv): Conv2d(2048, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(320, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3_1): BasicConv2d(
      (conv): Conv2d(2048, 384, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3_2a): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3_2b): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_1): BasicConv2d(
      (conv): Conv2d(2048, 448, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(448, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_2): BasicConv2d(
      (conv): Conv2d(448, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_3a): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch3x3dbl_3b): BasicConv2d(
      (conv): Conv2d(384, 384, kernel_size=(3, 1), stride=(1, 1), padding=(1, 0), bias=False)
      (bn): BatchNorm2d(384, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch_pool): BasicConv2d(
      (conv): Conv2d(2048, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (fc): Linear(in_features=2048, out_features=1000, bias=True)
)
```

```
In [98]: print(model_transfer.fc.in_features)
         print(model_transfer.fc.out_features)
```

```
2048
1000
```

```
In [42]: help(model_transfer)
```

```
Help on Inception3 in module torchvision.models.inception object:

class Inception3(torch.nn.modules.module.Module)
 |  Base class for all neural network modules.
 |
 |  Your models should also subclass this class.
 |
 |  Modules can also contain other Modules, allowing to nest them in
 |  a tree structure. You can assign the submodules as regular attributes::
 |
 |      import torch.nn as nn
 |      import torch.nn.functional as F
 |
 |      class Model(nn.Module):
 |          def __init__(self):
 |              super(Model, self).__init__()
 |              self.conv1 = nn.Conv2d(1, 20, 5)
 |              self.conv2 = nn.Conv2d(20, 20, 5)
 |
 |          def forward(self, x):
 |             x = F.relu(self.conv1(x))
 |             return F.relu(self.conv2(x))
 |
 |  Submodules assigned in this way will be registered, and will have their
 |  parameters converted too when you call `.cuda()`, etc.
 |
 |  Method resolution order:
 |      Inception3
 |      torch.nn.modules.module.Module
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, num_classes=1000, aux_logits=True, transform_input=False)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  forward(self, x)
 |      Defines the computation performed at every call.
 |
 |      Should be overridden by all subclasses.
 |
 |      .. note::
 |          Although the recipe for forward pass needs to be defined within
 |          this function, one should call the :class:`Module` instance afterwards
 |          instead of this since the former takes care of running the
 |          registered hooks while the latter silently ignores them.
 |
 |  ----------------------------------------------------------------------
```

```
|  Methods inherited from torch.nn.modules.module.Module:
|
|  __call__(self, *input, **kwargs)
|      Call self as a function.
|
|  __delattr__(self, name)
|      Implement delattr(self, name).
|
|  __dir__(self)
|      __dir__() -> list
|      default dir() implementation
|
|  __getattr__(self, name)
|
|  __repr__(self)
|      Return repr(self).
|
|  __setattr__(self, name, value)
|      Implement setattr(self, name, value).
|
|  __setstate__(self, state)
|
|  add_module(self, name, module)
|      Adds a child module to the current module.
|
|      The module can be accessed as an attribute using the given name.
|
|      Args:
|          name (string): name of the child module. The child module can be
|              accessed from this module using the given name
|          parameter (Module): child module to be added to the module.
|
|  apply(self, fn)
|      Applies ``fn`` recursively to every submodule (as returned by ``.children()``)
|      as well as self. Typical use includes initializing the parameters of a model
|      (see also :ref:`torch-nn-init`).
|
|      Args:
|          fn (:class:`Module` -> None): function to be applied to each submodule
|
|      Returns:
|          Module: self
|
|      Example::
|
|          >>> def init_weights(m):
|                  print(m)
|                  if type(m) == nn.Linear:
```

```
|                     m.weight.data.fill_(1.0)
|                     print(m.weight)
|
|         >>> net = nn.Sequential(nn.Linear(2, 2), nn.Linear(2, 2))
|         >>> net.apply(init_weights)
|         Linear(in_features=2, out_features=2, bias=True)
|         Parameter containing:
|         tensor([[ 1.,  1.],
|                 [ 1.,  1.]])
|         Linear(in_features=2, out_features=2, bias=True)
|         Parameter containing:
|         tensor([[ 1.,  1.],
|                 [ 1.,  1.]])
|         Sequential(
|           (0): Linear(in_features=2, out_features=2, bias=True)
|           (1): Linear(in_features=2, out_features=2, bias=True)
|         )
|         Sequential(
|           (0): Linear(in_features=2, out_features=2, bias=True)
|           (1): Linear(in_features=2, out_features=2, bias=True)
|         )
|
|  children(self)
|      Returns an iterator over immediate children modules.
|
|      Yields:
|          Module: a child module
|
|  cpu(self)
|      Moves all model parameters and buffers to the CPU.
|
|      Returns:
|          Module: self
|
|  cuda(self, device=None)
|      Moves all model parameters and buffers to the GPU.
|
|      This also makes associated parameters and buffers different objects. So
|      it should be called before constructing optimizer if the module will
|      live on GPU while being optimized.
|
|      Arguments:
|          device (int, optional): if specified, all parameters will be
|              copied to that device
|
|      Returns:
|          Module: self
|
```

```
|  double(self)
|      Casts all floating point parameters and buffers to ``double`` datatype.
|
|      Returns:
|          Module: self
|
|  eval(self)
|      Sets the module in evaluation mode.
|
|      This has any effect only on certain modules. See documentations of
|      particular modules for details of their behaviors in training/evaluation
|      mode, if they are affected, e.g. :class:`Dropout`, :class:`BatchNorm`,
|      etc.
|
|  extra_repr(self)
|      Set the extra representation of the module
|
|      To print customized extra information, you should reimplement
|      this method in your own modules. Both single-line and multi-line
|      strings are acceptable.
|
|  float(self)
|      Casts all floating point parameters and buffers to float datatype.
|
|      Returns:
|          Module: self
|
|  half(self)
|      Casts all floating point parameters and buffers to ``half`` datatype.
|
|      Returns:
|          Module: self
|
|  load_state_dict(self, state_dict, strict=True)
|      Copies parameters and buffers from :attr:`state_dict` into
|      this module and its descendants. If :attr:`strict` is ``True``, then
|      the keys of :attr:`state_dict` must exactly match the keys returned
|      by this module's :meth:`~torch.nn.Module.state_dict` function.
|
|      Arguments:
|          state_dict (dict): a dict containing parameters and
|              persistent buffers.
|          strict (bool, optional): whether to strictly enforce that the keys
|              in :attr:`state_dict` match the keys returned by this module's
|              :meth:`~torch.nn.Module.state_dict` function. Default: ``True``
|
|  modules(self)
|      Returns an iterator over all modules in the network.
```

```
|
|      Yields:
|          Module: a module in the network
|
|      Note:
|          Duplicate modules are returned only once. In the following
|          example, ``l`` will be returned only once.
|
|      Example::
|
|          >>> l = nn.Linear(2, 2)
|          >>> net = nn.Sequential(l, l)
|          >>> for idx, m in enumerate(net.modules()):
|                  print(idx, '->', m)
|
|          0 -> Sequential (
|            (0): Linear (2 -> 2)
|            (1): Linear (2 -> 2)
|          )
|          1 -> Linear (2 -> 2)
|
|  named_children(self)
|      Returns an iterator over immediate children modules, yielding both
|      the name of the module as well as the module itself.
|
|      Yields:
|          (string, Module): Tuple containing a name and child module
|
|      Example::
|
|          >>> for name, module in model.named_children():
|          >>>     if name in ['conv4', 'conv5']:
|          >>>         print(module)
|
|  named_modules(self, memo=None, prefix='')
|      Returns an iterator over all modules in the network, yielding
|      both the name of the module as well as the module itself.
|
|      Yields:
|          (string, Module): Tuple of name and module
|
|      Note:
|          Duplicate modules are returned only once. In the following
|          example, ``l`` will be returned only once.
|
|      Example::
|
|          >>> l = nn.Linear(2, 2)
```

```
|          >>> net = nn.Sequential(l, l)
|          >>> for idx, m in enumerate(net.named_modules()):
|                  print(idx, '->', m)
|
|          0 -> ('', Sequential (
|            (0): Linear (2 -> 2)
|            (1): Linear (2 -> 2)
|          ))
|          1 -> ('0', Linear (2 -> 2))
|
|  named_parameters(self, memo=None, prefix='')
|      Returns an iterator over module parameters, yielding both the
|      name of the parameter as well as the parameter itself
|
|      Yields:
|          (string, Parameter): Tuple containing the name and parameter
|
|      Example::
|
|          >>> for name, param in self.named_parameters():
|          >>>    if name in ['bias']:
|          >>>        print(param.size())
|
|  parameters(self)
|      Returns an iterator over module parameters.
|
|      This is typically passed to an optimizer.
|
|      Yields:
|          Parameter: module parameter
|
|      Example::
|
|          >>> for param in model.parameters():
|          >>>     print(type(param.data), param.size())
|          <class 'torch.FloatTensor'> (20L,)
|          <class 'torch.FloatTensor'> (20L, 1L, 5L, 5L)
|
|  register_backward_hook(self, hook)
|      Registers a backward hook on the module.
|
|      The hook will be called every time the gradients with respect to module
|      inputs are computed. The hook should have the following signature::
|
|          hook(module, grad_input, grad_output) -> Tensor or None
|
|      The :attr:`grad_input` and :attr:`grad_output` may be tuples if the
|      module has multiple inputs or outputs. The hook should not modify its
```

```
|      arguments, but it can optionally return a new gradient with respect to
|      input that will be used in place of :attr:`grad_input` in subsequent
|      computations.
|
|      Returns:
|          :class:`torch.utils.hooks.RemovableHandle`:
|              a handle that can be used to remove the added hook by calling
|              ``handle.remove()``
|
|  register_buffer(self, name, tensor)
|      Adds a persistent buffer to the module.
|
|      This is typically used to register a buffer that should not to be
|      considered a model parameter. For example, BatchNorm's ``running_mean``
|      is not a parameter, but is part of the persistent state.
|
|      Buffers can be accessed as attributes using given names.
|
|      Args:
|          name (string): name of the buffer. The buffer can be accessed
|              from this module using the given name
|          tensor (Tensor): buffer to be registered.
|
|      Example::
|
|          >>> self.register_buffer('running_mean', torch.zeros(num_features))
|
|  register_forward_hook(self, hook)
|      Registers a forward hook on the module.
|
|      The hook will be called every time after :func:`forward` has computed an output.
|      It should have the following signature::
|
|          hook(module, input, output) -> None
|
|      The hook should not modify the input or output.
|
|      Returns:
|          :class:`torch.utils.hooks.RemovableHandle`:
|              a handle that can be used to remove the added hook by calling
|              ``handle.remove()``
|
|  register_forward_pre_hook(self, hook)
|      Registers a forward pre-hook on the module.
|
|      The hook will be called every time before :func:`forward` is invoked.
|      It should have the following signature::
|
```

```
|          hook(module, input) -> None
|
|      The hook should not modify the input.
|
|      Returns:
|          :class:`torch.utils.hooks.RemovableHandle`:
|              a handle that can be used to remove the added hook by calling
|              ``handle.remove()``
|
|  register_parameter(self, name, param)
|      Adds a parameter to the module.
|
|      The parameter can be accessed as an attribute using given name.
|
|      Args:
|          name (string): name of the parameter. The parameter can be accessed
|              from this module using the given name
|          parameter (Parameter): parameter to be added to the module.
|
|  share_memory(self)
|
|  state_dict(self, destination=None, prefix='', keep_vars=False)
|      Returns a dictionary containing a whole state of the module.
|
|      Both parameters and persistent buffers (e.g. running averages) are
|      included. Keys are corresponding parameter and buffer names.
|
|      Returns:
|          dict:
|              a dictionary containing a whole state of the module
|
|      Example::
|
|          >>> module.state_dict().keys()
|          ['bias', 'weight']
|
|  to(self, *args, **kwargs)
|      Moves and/or casts the parameters and buffers.
|
|      This can be called as
|
|      .. function:: to(device)
|
|      .. function:: to(dtype)
|
|      .. function:: to(device, dtype)
|
|      It has similar signature as :meth:`torch.Tensor.to`, but does not take
```

```
|      a Tensor and only takes in floating point :attr:`dtype` s. In
|      particular, this method will only cast the floating point parameters and
|      buffers to :attr:`dtype`. It will still move the integral parameters and
|      buffers to :attr:`device`, if that is given. See below for examples.
|
|      .. note::
|          This method modifies the module in-place.
|
|      Args:
|          device (:class:`torch.device`): the desired device of the parameters
|              and buffers in this module
|          dtype (:class:`torch.dtype`): the desired floating point type of
|              the floating point parameters and buffers in this module
|
|      Returns:
|          Module: self
|
|      Example::
|
|          >>> linear = nn.Linear(2, 2)
|          >>> linear.weight
|          Parameter containing:
|          tensor([[ 0.1913, -0.3420],
|                  [-0.5113, -0.2325]])
|          >>> linear.to(torch.double)
|          Linear(in_features=2, out_features=2, bias=True)
|          >>> linear.weight
|          Parameter containing:
|          tensor([[ 0.1913, -0.3420],
|                  [-0.5113, -0.2325]], dtype=torch.float64)
|          >>> gpu1 = torch.device("cuda:1")
|          >>> linear.to(gpu1, dtype=torch.half)
|          Linear(in_features=2, out_features=2, bias=True)
|          >>> linear.weight
|          Parameter containing:
|          tensor([[ 0.1914, -0.3420],
|                  [-0.5112, -0.2324]], dtype=torch.float16, device='cuda:1')
|          >>> cpu = torch.device("cpu")
|          >>> linear.to(cpu)
|          Linear(in_features=2, out_features=2, bias=True)
|          >>> linear.weight
|          Parameter containing:
|          tensor([[ 0.1914, -0.3420],
|                  [-0.5112, -0.2324]], dtype=torch.float16)
|
|  train(self, mode=True)
|      Sets the module in training mode.
|
```

```
|      This has any effect only on certain modules. See documentations of
|      particular modules for details of their behaviors in training/evaluation
|      mode, if they are affected, e.g. :class:`Dropout`, :class:`BatchNorm`,
|      etc.
|
|      Returns:
|          Module: self
|
|  type(self, dst_type)
|      Casts all parameters and buffers to :attr:`dst_type`.
|
|      Arguments:
|          dst_type (type or string): the desired type
|
|      Returns:
|          Module: self
|
|  zero_grad(self)
|      Sets gradients of all model parameters to zero.
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from torch.nn.modules.module.Module:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  ----------------------------------------------------------------------
|  Data and other attributes inherited from torch.nn.modules.module.Module:
|
|  dump_patches = False
```

In [125]:
```
# Freeze training for all layers except AuxLogits and final fc

params_to_update = []

for name, module in model_transfer.named_children():
    if name == 'AuxLogits':
#         for param in module.conv0.parameters():
#             param.requires_grad = False
#         for param in module.conv1.parameters():
#             param.requires_grad = False
        for param in module.parameters():
            params_to_update.append(param)
```

```
            elif name == 'fc':
                for param in module.parameters():
                    params_to_update.append(param)
            elif name == 'Mixed_7a':
                for param in module.parameters():
                    params_to_update.append(param)
            elif name == 'Mixed_7b':
                for param in module.parameters():
                    params_to_update.append(param)
            elif name == 'Mixed_7c':
                for param in module.parameters():
                    params_to_update.append(param)
            else:
                for param in module.parameters():
                    param.requires_grad = False

        params_to_update
```

```
Out[125]: [Parameter containing:
           tensor([[[[-6.7129e-02]],

                    [[-2.4037e-03]],

                    [[ 1.6222e-02]],

                    ...,

                    [[-1.2233e-02]],

                    [[-2.3126e-02]],

                    [[ 2.4344e-02]]],


                   [[[ 3.0315e-02]],

                    [[ 2.1907e-02]],

                    [[-8.2309e-03]],

                    ...,

                    [[-7.3981e-03]],

                    [[ 6.5920e-02]],

                    [[ 1.0731e-02]]],
```

```
[[[-2.3710e-02]],

 [[ 1.7698e-02]],

 [[ 1.2942e-02]],

 ...,

 [[ 2.2704e-02]],

 [[ 3.6493e-02]],

 [[-1.8254e-02]]],


...,


[[[-1.5283e-01]],

 [[-4.8641e-02]],

 [[ 1.5826e-02]],

 ...,

 [[-1.3240e-03]],

 [[ 3.4759e-02]],

 [[ 5.1086e-03]]],


[[[ 1.3726e-01]],

 [[-3.5739e-02]],

 [[ 4.4856e-02]],

 ...,

 [[-4.4176e-02]],

 [[ 4.2760e-02]],

 [[-1.6315e-02]]],
```

```
        [[[ 6.8182e-02]],

         [[-6.4740e-02]],

         [[-1.4527e-02]],

         ...,

         [[ 2.7920e-02]],

         [[ 3.2788e-02]],

         [[-9.6598e-04]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter containing:
tensor([-0.1107,  0.0855, -0.1361, -0.0703,  0.3883, -0.1979, -0.0128,
         0.0941, -0.0155, -0.2168, -0.0935, -0.0593, -0.0301,  0.0318,
        -0.1886,  0.0057,  0.0635, -0.0528, -0.2067, -0.1267, -0.1022,
        -0.2055, -0.0309,  0.1435, -0.0729, -0.2450,  0.0587,  0.0357,
        -0.2015, -0.0947, -0.1211, -0.0077, -0.1852, -0.0324,  0.0719,
        -0.0172, -0.1033,  0.0096, -0.1557, -0.1020,  0.0123, -0.2573,
         0.0215, -0.1316, -0.0852,  0.0431, -0.2738,  0.0185, -0.0606,
        -0.0392,  0.0222, -0.1599, -0.0118,  0.0428,  0.0159, -0.0387,
        -0.1918,  0.0695,  0.2174, -0.2463, -0.1661,  0.0127, -0.2169,
        -0.2147, -0.1457,  0.3959, -0.1039,  0.0611, -0.1968, -0.0291,
        -0.1346,  0.1610, -0.0514, -0.1518, -0.0603, -0.1266,  0.1024,
         0.0778, -0.0558, -0.0044, -0.1336,  0.0629,  0.0849, -0.0523,
         0.1466, -0.0044, -0.0773,  0.0008, -0.0654, -0.2641, -0.1143,
        -0.0799, -0.0867,  0.0290,  0.0676, -0.0762, -0.0288, -0.0137,
         0.0388, -0.0504,  0.0809,  0.2140, -0.0861, -0.1324,  0.1397,
        -0.0257, -0.0596,  0.0178, -0.2088, -0.1349,  0.2165, -0.2169,
        -0.1142, -0.1348, -0.0333,  0.0679, -0.1651, -0.0547, -0.0179,
        -0.0845, -0.0348, -0.0354, -0.0975, -0.0276, -0.0534, -0.1675,
        -0.1076,  0.0098]), Parameter containing:
tensor([[[[-4.0899e-03, -9.6348e-03, -5.1115e-03, -1.3125e-02, -2.6694e-03],
          [-9.1506e-03, -7.9589e-03, -1.2011e-02, -3.1004e-03, -9.4923e-03],
          [-6.8799e-04, -1.2124e-02,  7.0282e-04, -1.4533e-02,  2.2083e-03],
          [-1.0537e-02, -8.8404e-03, -1.4569e-02, -8.1636e-03, -8.6491e-03],
```

```
 [ 1.9365e-03, -1.5060e-02, -5.1909e-03, -8.9324e-03,  1.2099e-03]],

[[ 4.3038e-03,  9.1932e-03,  3.5646e-03,  9.1807e-03,  5.8193e-04],
 [ 1.5862e-03,  2.9794e-03,  3.9198e-03,  4.1428e-03, -4.4741e-03],
 [ 3.3470e-03,  8.1074e-03,  6.9080e-03,  8.2265e-03,  1.0504e-05],
 [ 6.0341e-03,  3.3410e-04, -6.2147e-04,  1.0141e-03,  3.2103e-03],
 [-5.0926e-03,  6.6761e-03, -1.8965e-03,  8.3066e-03, -3.3586e-03]],

[[-1.6259e-02, -2.2856e-02, -1.6355e-02, -2.1717e-02, -1.6642e-02],
 [-2.3507e-02, -1.6459e-02, -1.7094e-02, -2.0293e-02, -2.4471e-02],
 [-1.6462e-02, -1.9682e-02, -1.0496e-02, -1.9244e-02, -1.7208e-02],
 [-1.3155e-02, -9.4907e-03, -3.3585e-03, -6.8402e-03, -1.3523e-02],
 [-1.3573e-02, -1.4313e-02, -1.4990e-02, -1.3405e-02, -1.0868e-02]],

...,

[[ 2.4888e-03,  1.5246e-03,  1.8705e-03,  2.7621e-03, -4.9932e-05],
 [-9.3702e-04, -8.1421e-03, -1.0705e-02, -7.9176e-03,  9.1505e-04],
 [-6.6616e-03, -1.1851e-02,  6.5251e-04, -1.3824e-02, -5.5987e-03],
 [ 3.6585e-03, -3.5477e-03, -1.4917e-03, -6.3925e-03,  3.1039e-03],
 [ 4.1160e-03, -1.3925e-02,  1.0049e-02, -1.3038e-02,  3.7793e-03]],

[[-1.1482e-02, -1.1297e-02, -1.2036e-02, -1.1412e-02, -1.2845e-02],
 [-8.9055e-03, -9.2590e-03, -8.8119e-03, -9.5793e-03, -1.1672e-02],
 [-1.7315e-02, -4.3146e-03, -6.9820e-03, -5.8363e-03, -1.9358e-02],
 [-1.1745e-02, -3.5208e-03, -1.1782e-02, -6.4159e-04, -1.2590e-02],
 [-1.1946e-02, -7.9794e-03, -1.0555e-02, -8.6018e-03, -1.2095e-02]],

[[-1.1501e-02, -1.3275e-02, -1.2631e-03, -1.6102e-02, -9.3621e-03],
 [ 5.1445e-04, -3.7454e-03, -1.2626e-02, -8.1104e-03,  6.1461e-04],
 [-9.3287e-03, -1.1775e-02, -2.4268e-03, -1.3559e-02, -7.1378e-03],
 [-1.2558e-03, -4.6898e-03, -1.2420e-02, -2.7410e-03, -2.9713e-03],
 [-1.8123e-02, -4.7162e-03, -1.2581e-02, -6.0962e-03, -1.8730e-02]]],


[[[-1.9765e-02, -6.7605e-03, -1.9380e-02, -4.1497e-03, -2.0146e-02],
  [-5.4351e-03, -7.2871e-03, -6.2648e-03, -3.5609e-03, -9.6049e-03],
  [-1.3879e-02, -1.1651e-02, -2.5458e-03, -1.2534e-02, -1.3013e-02],
  [-1.2228e-02, -1.4868e-02, -7.6464e-03, -1.6478e-02, -1.4054e-02],
  [-1.8376e-02, -1.0449e-02, -4.3409e-03, -1.6425e-02, -1.8717e-02]],

 [[-8.8323e-03, -3.1743e-02, -1.2347e-02, -3.3410e-02, -8.1586e-03],
  [-2.1473e-02, -2.8194e-02, -7.4031e-03, -2.3152e-02, -2.2683e-02],
  [-1.3678e-02, -2.5739e-02, -7.6874e-03, -2.4545e-02, -1.4646e-02],
  [-2.4023e-02, -3.0989e-02, -2.1821e-02, -2.6312e-02, -1.8329e-02],
  [-8.6538e-03, -2.1586e-02, -9.2583e-03, -2.1711e-02, -9.4008e-03]],

 [[ 2.4685e-02,  2.2505e-02,  1.2346e-02,  2.8896e-02,  1.8783e-02],
```

```
  [ 2.5872e-02,  1.7294e-02,  2.8672e-02,  2.3359e-02,  2.5978e-02],
  [ 2.6423e-02,  1.2338e-02,  1.2922e-02,  1.3593e-02,  2.6629e-02],
  [ 2.0208e-02,  1.7391e-02,  1.5323e-02,  1.3641e-02,  2.0909e-02],
  [ 3.4529e-02,  2.6841e-02,  2.3518e-02,  2.3101e-02,  3.7567e-02]],

 ...,

 [[-5.0417e-03, -6.4492e-03, -1.4046e-02, -6.1876e-03, -5.3768e-03],
  [-4.3159e-03, -4.4667e-03, -1.5684e-02, -6.2827e-03, -4.7708e-03],
  [ 1.5258e-03, -2.8118e-03, -4.8613e-03, -3.7201e-03,  2.0841e-03],
  [-1.1967e-02, -6.8793e-03, -8.0783e-03, -1.0490e-02, -1.2634e-02],
  [-1.3237e-03,  1.8551e-03, -3.1417e-03, -2.7008e-03,  1.1669e-03]],

 [[-1.1553e-02, -4.9029e-03, -1.7237e-02, -3.0092e-03, -1.3813e-02],
  [-1.2239e-02, -1.0538e-02, -1.2776e-02, -1.1418e-02, -8.1064e-03],
  [-8.4521e-03, -9.3751e-03, -9.8256e-03, -1.0067e-02, -1.5488e-02],
  [-5.2270e-03, -8.9389e-03, -1.8835e-02, -6.2812e-03, -8.9322e-04],
  [-1.4160e-02, -4.4896e-03, -1.5792e-02, -4.0210e-03, -1.1344e-02]],

 [[ 1.8821e-03, -2.5131e-02, -3.5786e-03, -2.7395e-02,  2.7381e-03],
  [-2.5287e-02, -1.1960e-02, -2.0776e-02, -1.4396e-02, -2.4298e-02],
  [-3.1454e-03, -1.8009e-02, -7.5227e-03, -2.1014e-02, -5.4085e-03],
  [-3.3000e-02, -1.8246e-02, -2.7932e-02, -1.3904e-02, -3.3486e-02],
  [-2.3372e-03, -3.7558e-02, -2.2768e-03, -3.6606e-02, -3.5756e-03]]],


[[[-1.0484e-02, -4.3172e-03, -1.7313e-02, -2.2516e-03, -1.2725e-02],
  [-8.5686e-03, -1.1917e-02, -2.1199e-02, -1.0021e-02, -5.6388e-03],
  [-9.0937e-03, -1.2977e-02, -9.2685e-03, -1.0819e-02, -5.8871e-03],
  [-9.1478e-03, -1.7320e-02, -1.8400e-02, -2.1488e-02, -1.0724e-02],
  [-3.7251e-03, -6.7648e-03, -3.4696e-03, -1.5397e-03, -1.1495e-03]],

 [[-1.5206e-03, -4.1147e-03, -2.0259e-03, -1.1901e-02, -5.0771e-04],
  [-1.6183e-02, -7.0341e-03, -3.9271e-03, -1.1268e-02, -1.1599e-02],
  [-7.9196e-03, -1.3578e-02, -1.0194e-02, -1.3883e-02, -5.1034e-03],
  [-1.9448e-02, -1.7392e-02, -1.0252e-02, -2.1167e-02, -2.2192e-02],
  [-1.3470e-02, -2.7408e-02, -1.8425e-02, -2.6825e-02, -1.4654e-02]],

 [[ 1.6533e-02,  2.3485e-02, -3.5527e-04,  2.4625e-02,  1.3386e-02],
  [ 1.3850e-02,  2.0916e-02,  2.0927e-02,  1.8121e-02,  1.7906e-02],
  [ 1.6911e-02,  1.3977e-02, -3.4755e-03,  1.4850e-02,  1.3414e-02],
  [ 1.9422e-02,  2.3008e-02,  2.6143e-02,  2.2824e-02,  1.4770e-02],
  [ 3.5868e-03,  2.1366e-02,  2.2439e-03,  2.4100e-02,  8.9616e-03]],

 ...,

 [[-2.8887e-03,  1.3388e-04, -5.8360e-03, -2.5588e-03, -2.3279e-03],
  [ 4.2728e-04, -2.8621e-03, -1.0375e-02, -3.2417e-03, -4.5940e-03],
```

```
  [-7.6533e-03, -3.6522e-04, -8.8232e-03,  5.3446e-03, -7.7505e-03],
  [ 2.3607e-04,  1.9092e-03, -6.0200e-03, -5.0867e-04,  8.4104e-04],
  [-1.6057e-02,  6.2463e-03, -2.0997e-02,  4.0406e-03, -9.2421e-03]],

 [[ 9.5598e-03,  9.2004e-03,  1.0287e-02,  8.4579e-03,  1.2546e-02],
  [ 1.1298e-02,  5.3800e-03,  6.7986e-03,  8.2806e-03,  1.1492e-02],
  [ 1.0222e-03,  9.6307e-03, -8.8387e-03,  7.4089e-03, -1.3865e-03],
  [ 1.9366e-03,  4.7782e-03,  2.4259e-03,  7.1280e-03,  6.3821e-03],
  [-1.4116e-02, -6.3340e-04, -1.8532e-02,  3.6766e-03, -1.5668e-02]],

 [[ 3.7934e-03,  1.9320e-02,  1.1320e-02,  1.7419e-02,  6.5428e-03],
  [ 8.7477e-03,  1.1100e-02,  3.5607e-03,  7.5184e-03,  9.6984e-03],
  [ 7.4885e-03,  1.7394e-02,  7.8541e-03,  1.6224e-02,  1.0463e-02],
  [ 1.4003e-02,  2.4002e-02,  1.2822e-02,  2.3284e-02,  1.3558e-02],
  [ 1.4490e-02,  4.9994e-03,  2.3222e-02,  1.1079e-02,  1.5846e-02]]],


...,


[[[ 5.0872e-03,  1.9602e-02,  5.5497e-04,  1.4492e-02,  5.1974e-03],
  [ 1.1539e-02,  1.9621e-02,  4.2702e-03,  1.7949e-02,  1.2976e-02],
  [ 7.9324e-03,  2.6060e-02,  5.7422e-03,  2.3820e-02,  5.8109e-03],
  [ 1.0835e-02,  8.3813e-03,  4.3046e-03,  1.1583e-02,  5.0393e-03],
  [ 6.5890e-03,  1.8746e-02,  8.5582e-03,  1.6180e-02,  6.2839e-03]],

 [[-7.3818e-03, -1.7096e-02, -1.4434e-02, -1.6531e-02, -4.6492e-03],
  [-1.6245e-02,  3.1738e-04, -6.9815e-03, -3.7002e-03, -2.1124e-02],
  [-1.5761e-02,  6.7128e-04, -1.7966e-02, -3.7071e-03, -1.3236e-02],
  [-1.6864e-02, -5.5611e-03, -1.7112e-02, -1.1937e-03, -1.5379e-02],
  [-6.7422e-03, -5.5689e-04, -1.2198e-02,  4.0840e-03, -5.9723e-03]],

 [[-5.7372e-03, -2.6903e-03, -4.7120e-03, -3.0087e-03, -5.4249e-03],
  [-8.7110e-03, -6.7475e-03, -4.4400e-03, -1.0247e-02, -5.5699e-03],
  [ 4.4089e-04, -2.6288e-03,  9.8739e-03, -3.7513e-03,  2.3216e-03],
  [-9.2190e-03, -1.1140e-02, -1.3590e-02, -1.4901e-02, -1.1982e-02],
  [-4.4144e-03, -1.1038e-02,  8.4192e-04, -1.7413e-02,  1.1263e-03]],

 ...,

 [[-3.0806e-03,  1.0851e-03,  2.6743e-03, -3.2819e-03, -2.6215e-03],
  [-1.0536e-02, -7.0619e-03,  4.5230e-03, -1.1949e-02, -9.1472e-03],
  [-2.9074e-03, -3.3247e-03,  1.0855e-02, -9.3096e-04,  7.4027e-04],
  [-9.7735e-03, -1.4189e-02, -1.8764e-03, -1.2105e-02, -1.0486e-02],
  [-2.4670e-03, -8.3887e-03,  2.0914e-03, -1.1000e-02, -1.2345e-04]],

 [[-8.4832e-03, -1.0126e-02,  1.4646e-03, -1.0036e-02, -9.6062e-03],
  [-5.9945e-03, -7.9984e-03, -6.8694e-03, -9.7095e-03, -4.5183e-03],
```

```
  [-2.3943e-02, -7.3646e-03, -1.4788e-02, -8.3108e-03, -2.1740e-02],
  [-1.1094e-02, -1.3324e-02, -1.1502e-02, -1.3741e-02, -1.2467e-02],
  [-9.2244e-03, -2.4845e-03, -3.2011e-03,  2.2649e-03, -1.3652e-02]],

 [[-1.3731e-02, -3.1077e-03, -1.2924e-02, -1.5492e-03, -1.6184e-02],
  [ 4.0982e-03,  3.4489e-03,  5.3293e-03,  1.7734e-04,  6.8027e-03],
  [-1.1417e-02,  4.6236e-03, -2.5973e-03,  4.8352e-03, -1.5621e-02],
  [-6.3029e-03,  3.1075e-03, -1.0455e-02,  6.0927e-03, -8.0933e-03],
  [-2.1992e-03,  5.3346e-03, -1.1623e-03,  2.7736e-03, -1.7967e-03]]],


[[[-1.0246e-02, -5.8809e-03, -8.8186e-03, -4.5373e-04, -9.6841e-03],
  [-1.2672e-02,  4.2868e-03, -9.4204e-03,  1.8563e-03, -1.1968e-02],
  [-1.7599e-02, -4.7283e-03, -1.2212e-02, -7.6953e-03, -1.6631e-02],
  [-1.8858e-02, -1.2859e-03, -1.7285e-02, -7.8725e-04, -1.1141e-02],
  [-2.8452e-02, -1.7595e-02, -2.5917e-02, -2.0331e-02, -2.5770e-02]],

 [[-1.0713e-03,  6.7592e-04, -3.8764e-03,  2.1364e-03, -1.1427e-03],
  [ 1.2375e-03, -5.2596e-03,  3.9413e-04, -3.7363e-03, -1.8441e-03],
  [-8.4770e-03,  2.7544e-03, -1.0045e-02,  2.0919e-03, -1.0276e-02],
  [-2.3904e-03, -7.9396e-03, -8.4324e-03, -2.6042e-03, -8.8285e-03],
  [-8.4872e-03,  7.9367e-03, -1.2950e-02,  7.5372e-03, -7.3492e-03]],

 [[-3.8687e-03, -1.2309e-02, -3.9258e-03, -1.3881e-02, -6.4859e-03],
  [-8.5751e-03, -4.6068e-03, -5.6217e-03, -3.4819e-03, -7.7570e-03],
  [-1.0429e-03, -1.2067e-03,  1.1150e-03, -2.6274e-03, -1.9346e-03],
  [-5.8101e-03, -2.5173e-03, -8.9137e-03, -5.9754e-03, -9.8980e-03],
  [-3.1412e-03,  1.5693e-03, -2.7324e-03, -1.6073e-03, -1.7899e-03]],

 ...,

 [[ 7.5235e-03, -1.0311e-03,  6.0158e-04, -3.3934e-03,  8.8661e-03],
  [-8.9815e-03, -7.8624e-03, -8.1288e-03, -5.9800e-03, -5.4549e-03],
  [ 1.7473e-02, -5.7112e-03, -1.6558e-04, -8.5299e-03,  1.7686e-02],
  [-1.7698e-03, -1.4161e-02, -6.0580e-03, -1.5465e-02, -4.7328e-03],
  [ 1.2377e-02, -2.2191e-03,  7.6589e-03, -2.7988e-03,  1.1173e-02]],

 [[-1.1605e-02,  7.5188e-03, -1.8189e-02,  5.0476e-03, -1.0560e-02],
  [ 3.6877e-03,  1.0044e-02,  3.8767e-03,  1.3383e-02,  4.6042e-03],
  [-8.6393e-03,  1.4677e-02,  1.8076e-03,  8.7530e-03, -7.6612e-03],
  [ 4.3214e-03,  4.3568e-03, -2.4991e-03,  5.6059e-03,  6.9399e-03],
  [-6.6037e-04,  8.5427e-03, -4.2493e-03,  6.2292e-03, -3.6677e-03]],

 [[-7.4883e-03, -7.8557e-03, -1.5490e-02, -1.0125e-02, -9.2785e-03],
  [-1.3815e-02, -4.6393e-03, -1.9608e-02, -7.4823e-03, -9.7328e-03],
  [-4.6483e-03, -6.5813e-03,  3.8897e-05, -3.3026e-03, -7.6728e-03],
  [-9.0873e-03, -3.3578e-03,  4.0460e-04, -1.5618e-03, -6.2361e-03],
  [-4.9915e-03, -3.9452e-03,  1.0025e-04, -8.6527e-03, -3.4209e-03]]],
```

```
          [[[ 1.0451e-02,  1.8375e-02,  4.3429e-03,  1.6902e-02,  9.0019e-03],
            [ 1.5745e-02,  1.7718e-02,  1.0721e-02,  1.9263e-02,  1.3374e-02],
            [-2.2830e-03,  2.5371e-02,  3.1787e-03,  2.2740e-02, -4.6061e-03],
            [ 1.5178e-02,  1.2290e-02,  1.4630e-02,  9.2210e-03,  1.3511e-02],
            [ 7.9822e-03,  1.8085e-02,  1.1520e-02,  1.5225e-02,  8.8921e-03]],

           [[ 1.4534e-03,  4.7720e-03,  5.7566e-03,  6.6404e-03,  2.3766e-03],
            [ 7.7752e-03,  3.8758e-03,  8.9002e-03, -1.0767e-03,  1.0382e-02],
            [-5.4682e-03, -4.5699e-03, -2.2202e-03, -4.8174e-03, -6.9412e-03],
            [ 3.5919e-03,  3.5737e-03,  9.8773e-03,  6.7722e-03,  7.5087e-03],
            [-8.8517e-03,  2.5335e-05,  1.8203e-03, -3.5800e-03, -9.0906e-03]],

           [[-3.1397e-03,  1.3150e-04, -1.5436e-02,  1.5334e-03, -3.3164e-03],
            [-6.5944e-03, -2.2768e-03, -4.8568e-03, -1.9894e-04, -2.2111e-03],
            [-9.3686e-03, -5.2340e-03, -1.8876e-05, -6.7396e-03, -6.1452e-03],
            [-1.0106e-03, -6.0533e-03, -8.6192e-03, -3.8687e-03, -1.2327e-03],
            [-3.7749e-03, -1.5871e-02, -5.4538e-03, -9.5409e-03, -7.0508e-03]],

           ...,

           [[ 1.1331e-02,  9.9896e-03,  6.6815e-03,  1.2849e-02,  1.0873e-02],
            [ 1.0630e-02,  2.1208e-02,  1.0569e-02,  1.9118e-02,  1.1341e-02],
            [ 1.0088e-02,  1.5647e-02, -1.3570e-03,  1.3931e-02,  4.5452e-03],
            [ 1.0107e-02,  1.1653e-02,  7.3275e-03,  1.1401e-02,  8.2761e-03],
            [ 1.2061e-02,  1.1961e-02,  1.7961e-02,  1.3319e-02,  1.1705e-02]],

           [[ 2.3347e-02,  2.0542e-02,  9.2498e-03,  2.0085e-02,  1.6157e-02],
            [ 2.9560e-02,  1.7861e-02,  2.2560e-02,  1.7875e-02,  2.4944e-02],
            [ 1.4421e-02,  1.6264e-02,  8.2590e-03,  1.2945e-02,  1.5167e-02],
            [ 1.7066e-02,  9.6061e-03,  1.5573e-02,  9.2698e-03,  1.9580e-02],
            [ 1.3584e-02,  1.4611e-02,  1.6434e-02,  1.1685e-02,  1.7538e-02]],

           [[ 6.9368e-03, -1.1859e-03,  9.4230e-03,  5.3163e-03,  7.8285e-03],
            [ 6.3102e-03, -4.8882e-03,  6.6265e-03, -2.6475e-03,  1.5213e-03],
            [ 1.0583e-02, -3.5425e-03,  2.4499e-02, -6.0200e-04,  1.1829e-02],
            [ 2.9365e-03, -4.9014e-03,  1.6748e-02, -3.7059e-03,  3.8688e-05],
            [ 9.4121e-03,  1.3983e-02,  1.4536e-02,  1.0499e-02,  9.1768e-03]]]]), Para
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
```

```
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
      tensor([ 0.1126,  0.4006,  0.3551,  0.3814,  0.2760,  0.3865,  0.2420,
               0.1764,  0.2312,  0.3308,  0.4672,  0.2291,  0.3555,  0.4063,
               0.2778,  0.2775,  0.1772,  0.2339,  0.3711,  0.5950,  0.2838,
               0.3539,  0.5350,  0.1207,  0.4288,  0.3089,  0.2970,  0.3174,
               0.2854,  0.3146,  0.1542,  0.2883,  0.3578,  0.4054,  0.3500,
               0.1568, -0.1251,  0.3301,  0.3997,  0.2891,  0.2818,  0.3258,
               0.3331,  0.3357,  0.3815,  0.2650,  0.1975,  0.2817,  0.2700,
               0.3161,  0.4397,  0.2592,  0.3863,  0.2711,  0.3771,  0.4450,
               0.2575,  0.2923,  0.3225,  0.2766,  0.2464,  0.2917,  0.4393,
               0.3908,  0.2175,  0.5008,  0.4587,  0.3918,  0.4694,  0.3608,
               0.4128,  0.2752,  0.1869,  0.3507,  0.2318,  0.3723,  0.4038,
               0.3487,  0.2665,  0.3253,  0.3325,  0.4761,  0.3399,  0.2793,
               0.3712,  0.3339,  0.3305,  0.2170,  0.4057,  0.2140,  0.2573,
               0.3821,  0.2299,  0.2217,  0.3374,  0.2727,  0.0628,  0.1477,
               0.4362,  0.2312,  0.1208,  0.3004,  0.1538,  0.3912,  0.3088,
               0.3754,  0.2314,  0.1392,  0.4072,  0.3139,  0.2544,  0.3627,
               0.1647,  0.3859,  0.3385,  0.3509,  0.3464,  0.3628,  0.3214,
               0.3066,  0.3225,  0.2662,  0.4581,  0.2499,  0.2814,  0.3489,
               0.2711,  0.2620,  0.4392,  0.3124,  0.3535,  0.4278,  0.3223,
               0.0727,  0.4141,  0.3501,  0.2430,  0.4453,  0.3101,  0.4578,
               0.2240,  0.0838,  0.3688,  0.4065,  0.3313,  0.5167,  0.3223,
               0.3185,  0.3787,  0.3289,  0.2803,  0.1382,  0.4139,  0.3773,
               0.4025,  0.3142,  0.2417,  0.2538,  0.3878,  0.1724,  0.3472,
               0.3699,  0.3833,  0.3180,  0.2718,  0.3013,  0.1375,  0.3385,
               0.3117,  0.5739,  0.3251,  0.3378,  0.2077,  0.4848,  0.4129,
               0.3750,  0.4355,  0.3761,  0.2541,  0.3703,  0.5282,  0.3025,
               0.3682,  0.3035,  0.5199,  0.3508,  0.4352,  0.5958,  0.2814,
               0.3188,  0.2641,  0.2791,  0.2319,  0.3697,  0.5515,  0.2938,
               0.6307,  0.2923,  0.2921,  0.3855,  0.4937,  0.4423,  0.1631,
               0.5933,  0.3294,  0.4094,  0.4582,  0.4631,  0.3586,  0.1430,
               0.3581,  0.3626,  0.3091,  0.4424,  0.4045,  0.3413,  0.3605,
               0.3369,  0.1601,  0.3716,  0.2401,  0.1760,  0.2182,  0.4961,
               0.3561,  0.1422,  0.3140,  0.1636,  0.3136,  0.4209,  0.3551,
               0.2614,  0.2666,  0.2675,  0.2886,  0.2200,  0.4534,  0.2796,
               0.3512,  0.3250,  0.1600,  0.3014,  0.3609,  0.4405,  0.3052,
               0.3509,  0.1576,  0.3842,  0.4150,  0.2516,  0.4673,  0.3025,
               0.2605,  0.4929,  0.2090,  0.4520,  0.2197,  0.2153,  0.5191,
               0.1798,  0.2212,  0.3440,  0.2862,  0.3856,  0.2873,  0.4542,
               0.2731,  0.4582,  0.2988,  0.2810,  0.1765,  0.2708,  0.3329,
               0.4392,  0.3500,  0.3216,  0.3604,  0.1993,  0.3796,  0.3151,
               0.2521,  0.3239,  0.1002,  0.1032,  0.4001,  0.3452,  0.2973,
```

```
0.3132,  0.2037,  0.2134,  0.6312,  0.1516,  0.2206,  0.4650,
0.4786,  0.3960,  0.3278,  0.4325,  0.2526,  0.2447,  0.3830,
0.3257,  0.3696,  0.4136,  0.3703,  0.3757,  0.2362,  0.1826,
0.2813,  0.3785,  0.3366,  0.1305,  0.1439,  0.2090,  0.1905,
0.1260,  0.1368,  0.2286,  0.1378,  0.2468,  0.5766,  0.2696,
0.3351,  0.3473,  0.1634,  0.4069,  0.4098,  0.1665,  0.3005,
0.3604,  0.3113,  0.4358,  0.3240,  0.3746,  0.4205,  0.1969,
0.2701,  0.4110,  0.4166,  0.2622,  0.4250,  0.3526,  0.5435,
0.3783,  0.1745,  0.4198, -0.0053,  0.3149,  0.3457,  0.3156,
0.1904,  0.5783,  0.2589,  0.4016,  0.1837,  0.1463,  0.2876,
0.4249,  0.3471,  0.2322,  0.3104,  0.2233,  0.3950,  0.2845,
0.3175,  0.4313,  0.2012,  0.2879,  0.2164,  0.0588,  0.4123,
0.3426,  0.4796,  0.2550,  0.3910,  0.5252,  0.1837,  0.1448,
0.4006,  0.1609,  0.1856,  0.4649,  0.2112,  0.3872,  0.5582,
0.3560,  0.4968,  0.2021,  0.2461,  0.2362,  0.1972,  0.1478,
0.3217,  0.3659,  0.1238,  0.2831,  0.2721,  0.5009,  0.2300,
0.3220,  0.3288,  0.2346,  0.2590,  0.3700,  0.2638,  0.2411,
0.3066,  0.3683,  0.2777,  0.2293,  0.1924,  0.2747,  0.3065,
0.4017,  0.4705,  0.4489,  0.4688,  0.4535,  0.2892,  0.3838,
0.2786,  0.2918,  0.2632,  0.2592,  0.1546,  0.3044,  0.3060,
0.2605,  0.4978,  0.3460,  0.4584,  0.2758,  0.2949,  0.4924,
0.3580,  0.2800,  0.3531,  0.2311,  0.1809,  0.3079,  0.2360,
0.3455,  0.2688,  0.2328,  0.3402,  0.2054,  0.4699,  0.4745,
0.2783,  0.2663,  0.2938,  0.2659,  0.3734,  0.2985,  0.3250,
0.2862,  0.3641,  0.2709,  0.3920,  0.3271,  0.2687,  0.4737,
0.2951,  0.3236,  0.1491,  0.3427,  0.2811, -0.0874,  0.4806,
0.2015,  0.3232,  0.1793,  0.5024,  0.0351,  0.2311,  0.2688,
0.2533,  0.3820,  0.2768,  0.1706,  0.2198,  0.3450,  0.1863,
0.2730,  0.0911,  0.3382,  0.4634,  0.2721,  0.4082,  0.3060,
0.2171,  0.2248,  0.2105,  0.4160,  0.4404,  0.2843,  0.3179,
0.3515,  0.3750,  0.2362,  0.3080, -0.2725,  0.3666,  0.3073,
0.2457,  0.3897,  0.4407,  0.2968,  0.4236,  0.4505,  0.4580,
0.4716,  0.2574,  0.2654,  0.2107, -0.0966,  0.3095,  0.2453,
0.5228,  0.5758,  0.3571,  0.2695,  0.2936,  0.2375,  0.1383,
0.1121,  0.2205,  0.3348,  0.2581,  0.3527,  0.1486,  0.3267,
0.3808,  0.4698,  0.2036,  0.5017,  0.5029,  0.3033,  0.3639,
0.3991,  0.2346,  0.1714,  0.1338,  0.2435,  0.3629,  0.1208,
0.2619,  0.3441,  0.1786,  0.1648,  0.0970,  0.3101,  0.1316,
0.2320,  0.1839,  0.2638,  0.1862,  0.1104,  0.3829,  0.4447,
0.2585,  0.0847,  0.3361,  0.1803,  0.1819,  0.4397,  0.2915,
0.4021,  0.3160,  0.1154,  0.3062,  0.3359,  0.2785,  0.5691,
0.3668,  0.1674,  0.2930,  0.0722,  0.2780,  0.1896,  0.3090,
0.2393,  0.2025,  0.3363,  0.3528,  0.4975,  0.2878,  0.3373,
0.5605,  0.4572,  0.3149,  0.2846,  0.2708,  0.4804,  0.5328,
0.2815,  0.1580,  0.2557,  0.3418,  0.5219,  0.4572,  0.4265,
0.2609,  0.3639,  0.1044,  0.1916,  0.3280,  0.3768,  0.3233,
0.2357,  0.1641,  0.2380,  0.4406,  0.3965,  0.4368,  0.3590,
0.4421,  0.3093,  0.3487,  0.3244,  0.0607,  0.2422,  0.2448,
```

```
         0.2521,  0.1055,  0.2827,  0.3675,  0.3509,  0.5220,  0.6554,
         0.2800,  0.2684,  0.1735,  0.3112,  0.2680,  0.4508,  0.3783,
         0.2547,  0.4431,  0.3425,  0.3761,  0.1439,  0.4273,  0.3640,
         0.2475,  0.3105,  0.2884,  0.4479,  0.2038,  0.3311,  0.2602,
         0.2902,  0.2638,  0.3096,  0.2540,  0.2532,  0.4772,  0.2757,
         0.2985,  0.4311,  0.4830,  0.5296,  0.3307,  0.3184,  0.2361,
         0.3557,  0.3144,  0.3233,  0.5078,  0.3940,  0.2873,  0.2205,
         0.1934,  0.1710, -0.1204,  0.3228,  0.3114,  0.3922,  0.1957,
         0.3765,  0.2126,  0.3690,  0.3869,  0.3773,  0.2714,  0.4253,
         0.3829,  0.3177,  0.4892,  0.3122,  0.1602,  0.4019,  0.3165,
         0.2003,  0.3641,  0.3321,  0.2347,  0.2522,  0.4141,  0.3220,
         0.3306,  0.4657,  0.3171,  0.3957, -0.0046,  0.3079,  0.2622,
         0.2806,  0.4159,  0.3497,  0.3390,  0.2230,  0.1496,  0.4678,
         0.2150,  0.3944,  0.3558,  0.4656,  0.1761,  0.4565,  0.2771,
         0.3127,  0.2505,  0.2480,  0.4794,  0.2760,  0.3126,  0.4921,
         0.2260,  0.1782,  0.1773,  0.2318,  0.2365,  0.2921,  0.3042,
         0.3581,  0.3846,  0.4133,  0.2834,  0.3858,  0.1570,  0.2911,
         0.1675,  0.2208,  0.5062,  0.4290,  0.2761,  0.3315,  0.3567,
         0.3111,  0.3440,  0.2777,  0.4681,  0.3012,  0.4033,  0.2658,
         0.0023,  0.3871,  0.1182,  0.1567,  0.5212,  0.3158,  0.2466,
         0.3516,  0.1310,  0.2610,  0.3569,  0.1495]), Parameter containing:
tensor([[ 1.1876e-02,  1.3307e-01, -2.1387e-02,  ...,  3.2194e-02,
         -3.2013e-02, -2.9397e-02],
        [ 1.0155e-01, -1.7175e-02, -1.5966e-02,  ..., -1.1804e-02,
          4.1009e-02, -3.5827e-02],
        [-1.9860e-02,  9.6962e-03, -2.4665e-02,  ..., -3.1029e-02,
         -1.9245e-02,  1.5183e-01],
        ...,
        [ 9.5499e-02, -2.3604e-02, -4.3647e-02,  ..., -5.2226e-02,
          4.2011e-02, -1.1601e-02],
        [ 5.5776e-02, -3.7618e-02,  3.4657e-02,  ..., -3.6613e-02,
          4.8366e-02,  1.3109e-01],
        [ 1.8254e-02, -3.2060e-02,  1.0467e-01,  ...,  1.4193e-02,
          1.4389e-02,  3.0633e-02]]), Parameter containing:
tensor([ 0.5108, -0.3865,  0.0163, -0.0020, -0.4012, -0.3396, -0.0999,
        -0.1865, -0.3615, -0.3017, -0.2470,  0.1988, -0.1376,  0.0791,
        -0.1894, -0.1326,  0.2047,  0.2242,  0.0798, -0.1221, -0.3449,
        -0.1478,  0.2324,  0.0660,  0.0240, -0.3388, -0.1662, -0.1204,
         0.1144,  0.4214,  0.1029, -0.4254, -0.2035, -0.0506, -0.3660,
        -0.1540, -0.2571,  0.0896, -0.0170,  0.3548, -0.1548,  0.3902,
        -0.3133, -0.4288,  0.7371,  0.3374,  0.0843,  0.2321, -0.3938,
        -0.1726, -0.3001,  0.2837, -0.2758, -0.1048, -0.4594, -0.0645,
        -0.1319,  0.2885,  0.1736, -0.2749, -0.0611, -0.1904, -0.0233,
         0.0169, -0.1016, -0.2854, -0.1999,  0.1644,  0.0081,  0.3310,
        -0.1620,  0.3584,  0.6731, -0.7200,  0.0858,  0.2380, -0.3007,
        -0.1074, -0.3038,  0.3087, -0.3315, -0.8555, -0.3100, -0.2504,
        -0.2728,  0.0903,  0.0659,  0.5948, -0.1444, -0.2421, -0.2133,
         0.0654, -0.4771,  0.2038, -0.1229, -0.2439,  0.0334,  0.0125,
```

```
-0.2615, -0.3513, -0.2009,  0.3074,  0.4317, -0.4169,  0.0509,
 0.2537, -0.0408, -0.9108, -0.1690, -0.4615, -1.1210, -0.9911,
-0.3655, -0.1771,  0.0153, -0.8706, -0.5690,  0.2132,  0.1044,
-0.4850, -0.1026,  0.3132,  0.6107,  0.3677,  0.6741,  0.1007,
-0.4592, -0.1519, -0.1299,  0.0035, -0.4973,  0.0626,  0.0420,
-0.2928, -0.0911, -0.3088, -0.1545, -0.4224, -0.1004, -0.1431,
-0.1858, -0.2474, -0.4309, -0.4042, -0.3490, -0.4931, -0.2365,
-0.2277,  0.3578, -0.0532, -0.1299,  0.0329, -0.4963,  0.3057,
 0.0816,  0.3288,  0.1911, -0.0194, -0.4819,  0.4738,  0.4750,
 0.6383,  0.3852, -0.1249,  0.2730, -0.5936,  0.0192, -0.4554,
 0.2442,  0.1605,  0.4104,  0.1658, -0.2783, -0.0944,  0.6320,
-0.0334,  0.2466,  0.3084, -0.0246, -0.0678,  0.4149,  0.1961,
 0.6023, -0.1386,  0.3661,  0.4241,  0.4280,  0.2635, -0.0129,
 0.1588, -0.0364,  0.1700,  0.2458,  0.4225,  0.5092,  0.3624,
 0.7659,  0.2785,  0.5989,  0.1380,  0.2289,  0.3747,  0.5608,
 0.3190,  0.3360,  0.2813, -0.0582,  0.5379, -0.3295,  0.3565,
-0.2159,  0.0854,  0.2727,  0.0460,  0.2211,  0.5676,  0.1459,
 0.3110,  0.3123,  0.1460, -0.0189, -0.3507,  0.4031,  0.2370,
 0.1184,  0.1180,  0.3531,  0.2314,  0.0923,  0.7258,  0.4369,
 0.3132,  0.5094,  0.2382,  0.1471,  0.4353,  0.1922, -0.1763,
 0.6865,  0.6471,  0.5015, -0.1037,  0.4340,  0.1476,  0.4652,
-0.1622, -0.1756,  0.7710,  0.4199,  0.3803,  0.3798, -0.0110,
-0.0685,  0.2635, -0.1624,  0.5286,  0.4487,  0.5736,  0.1655,
 0.1102, -0.0667,  0.3185, -0.0879,  0.2192,  0.2587,  0.1893,
 0.4078,  0.3303, -0.0004,  0.0396, -0.1475,  0.1508, -0.0866,
-0.1876, -0.2673, -0.2093,  0.0983, -0.0856, -0.1974,  0.0272,
 0.1116,  0.4741,  0.2752,  0.0338,  0.2943,  0.2535, -0.2224,
 0.1522,  0.1474, -0.0481,  0.1161,  0.3334,  0.3721,  0.0302,
-0.2778, -0.3227, -0.6023, -0.0484, -0.5397, -0.3533, -0.2546,
-0.5565, -0.3059,  0.0679, -0.1612, -0.9009,  0.0309, -0.1162,
 0.1329, -0.1944, -0.4510,  0.0028, -0.0904,  0.0005, -0.2853,
-0.1729,  0.0750, -0.2869,  0.2121,  0.1695,  0.0430,  0.4880,
-0.0604, -0.0280,  0.1973,  0.1715, -0.1379, -0.3605, -0.3701,
-0.3953, -0.5628, -0.6710,  0.4193,  0.2535,  0.3411,  0.1024,
-0.5118, -0.5348, -0.4393,  0.8382,  0.2029, -0.6437, -0.2805,
-0.1239, -0.1373,  0.0302,  0.2648,  0.0491,  0.0302,  0.0387,
-0.0112, -0.0328, -0.3715, -0.3104, -0.1429,  0.1332, -0.2840,
-0.5684,  0.0254,  0.4162, -0.5387, -0.0302,  0.3617,  0.5639,
-0.1326, -0.1632, -0.0948, -0.3163, -0.5596, -0.5244,  0.4447,
 0.0110, -0.0462,  0.2655, -0.2521, -0.0304, -0.2989,  0.1561,
 0.3840, -0.0709,  0.2250, -0.3143,  0.2007, -0.0145, -0.2062,
-0.1765,  0.1555,  0.2884,  0.2693,  0.4911, -0.3989,  0.4977,
-0.0097, -0.4009,  0.6183,  0.6603,  0.2414,  0.3168, -0.0082,
-0.3681, -0.3404,  0.3545,  0.3035, -0.3044, -0.3380,  0.1235,
-0.2771,  0.1332,  0.1705,  0.2890, -0.2337, -0.0426,  0.2836,
 0.2668, -0.1777, -0.4710, -0.4828, -0.0406, -0.5943, -0.3930,
 1.0471, -0.6956, -0.0376, -0.2444, -0.0292, -0.3378,  0.2673,
-0.2375,  0.0988, -0.2716, -0.3621, -0.1160,  0.5188,  0.1644,
```

```
-0.8402,  0.0401,  0.4532, -0.1329,  0.4384, -0.4634,  0.0760,
 0.4197,  0.0976, -0.5252,  0.1050,  0.2946, -0.4961,  0.2410,
 0.3758,  0.2874,  0.0794, -0.5263,  0.5788,  0.2931, -0.0142,
-0.4156,  0.1192, -0.2910, -0.3525, -0.2064, -0.0733, -0.1123,
-0.1821,  0.1936, -0.0195, -0.2667,  0.5487,  0.0209,  0.2263,
 0.5489, -0.4581,  0.6714,  0.0169,  0.3065,  0.8133,  0.5070,
-0.2240,  0.9336, -0.1776,  0.0961,  0.4941, -0.3669,  0.3152,
-0.5044, -0.0387,  0.2980,  0.3584,  0.4945, -1.0411,  0.1381,
-0.0553, -0.1312, -0.0212, -0.0829, -0.3768, -0.0905,  0.2526,
-0.1588,  0.3274,  0.1474, -0.3024, -0.0685, -0.1846,  0.0626,
 0.7352,  0.5901, -1.6986,  0.0314, -0.2439, -0.2703,  0.2498,
 0.2412,  0.1627,  0.2781,  0.1480,  0.2184, -0.5690,  0.1595,
 0.0208, -0.5535, -0.1654,  0.6109,  0.0915,  0.3064,  0.0917,
-0.1220,  0.5173,  0.4705, -0.0124, -0.6750,  0.1697,  0.2587,
-0.5245, -0.2362,  0.1941, -0.5654, -0.0203, -0.1573, -0.3957,
-0.2861, -0.1579, -0.0796, -0.2240, -0.5414,  0.6667, -0.2668,
 0.3685,  0.1977,  0.7381, -0.6340,  0.5038,  0.1397,  0.0596,
-0.2050,  0.4935,  0.5111, -0.5796, -0.0388, -0.2000, -0.8129,
 0.2754,  0.0151, -0.3693,  0.2818,  0.0123, -0.1403,  0.3549,
 0.0946, -0.0548,  0.1991,  0.8913,  0.5109, -0.1137, -0.3852,
 0.1294,  0.2843, -0.1766,  0.6299,  0.3927,  0.0636, -0.9107,
 0.0259, -0.4305, -0.0543, -1.4942,  0.0430,  0.1808,  0.3015,
 0.2545,  0.6404,  0.6915, -0.8776,  0.0282, -0.1335,  0.6546,
-0.1655, -0.1801,  0.1429,  0.5633, -0.3653, -0.1585,  0.0188,
-0.4295,  0.7029,  0.3224,  0.5813,  0.0693, -0.3125, -0.0340,
 0.4128,  0.0815, -0.3957,  0.6107,  0.3844, -0.1797, -0.9421,
-1.2215,  0.1225,  0.0793, -0.5921,  0.5648,  0.1066, -0.0237,
-0.2393,  0.0120,  0.0972, -0.1017,  0.5823,  0.2016, -0.0157,
 0.0664, -0.0820, -0.0999,  0.7067, -0.2827,  0.3284, -0.0108,
 0.1109,  0.4290,  0.4852, -0.5337,  0.3435, -0.1848, -0.3020,
-0.8555, -0.4547, -0.7028,  0.4849, -0.1467, -0.6815,  0.1456,
 0.1769,  0.2805,  0.1419,  0.2026,  0.6903,  0.4027,  0.4769,
-0.0148, -0.1562, -0.2746,  0.3121,  0.2074, -0.4041,  0.5634,
 0.2773, -0.1137,  0.1342, -0.2653,  0.0667,  0.0355,  0.1531,
 0.1952,  0.2125,  0.1330,  0.2787, -0.0528, -0.9228, -0.1407,
-0.7323, -0.3576,  0.6137, -0.0206, -0.3453, -0.4524,  0.4642,
 0.1772, -0.2051,  0.3700, -0.2072,  0.1525,  0.4275,  0.0537,
-0.0689, -0.0826, -0.0827,  0.4024,  0.1294, -0.4674,  0.0279,
-0.2896, -0.1730, -0.5675, -0.5047,  0.7033,  0.3903, -0.0416,
 0.4949, -0.2584,  0.1928,  0.2163, -0.2194, -0.3440, -0.0788,
-0.7215,  0.1043, -0.4634,  0.1817,  0.3095, -0.0612, -0.0111,
-0.7355, -0.0032, -1.2708,  0.1086,  0.1528, -0.2123, -0.0958,
-0.1452, -0.4023, -0.3840,  0.4502,  0.2178, -0.3248,  0.5384,
 0.1135, -0.1793, -0.0012, -0.0761,  0.6211,  0.1259, -0.5820,
 0.2049, -0.2530,  0.5688,  0.4290, -0.3705,  0.4403,  0.0426,
 0.1553, -0.3624,  0.0479,  0.6793, -0.4909,  0.4226,  0.2044,
 0.3340,  0.6591,  0.5858,  0.1992, -0.3201,  0.0932, -0.0711,
 0.6906,  0.1229, -0.4653, -0.0783, -0.5363, -0.4314, -0.2357,
```

```
          0.0571, -0.3251, -0.3496,  0.0718, -0.5930,  0.2227,  0.5221,
         -0.2763,  0.4311,  0.1451,  0.4368, -0.1686,  0.5836, -0.8861,
          0.1104,  0.1155,  0.2574, -0.0800, -0.2600,  0.0452, -0.1908,
         -0.0155,  0.1718, -0.1479, -0.4375, -0.6162,  0.3424, -0.4211,
         -0.2880,  0.1798,  0.3103,  0.3272, -0.4244,  0.1820, -0.2466,
          0.0490, -0.0667, -0.1120,  0.4814,  0.6735,  0.0210,  0.3051,
         -0.1197,  0.4576, -0.1596, -0.2313,  0.1265, -0.0025, -0.5430,
          0.4424, -0.0506,  0.0087,  0.7611,  0.4124, -0.3180, -0.8870,
          0.0303,  0.1233,  0.3894, -0.0657, -0.6592, -0.2459,  0.0501,
          0.2531,  0.4296,  0.1954,  0.1138,  0.0187, -0.0604,  0.2097,
         -0.0039, -0.2205,  0.1841,  0.0272, -0.4460,  0.0727,  0.0173,
          0.0164,  0.6954, -0.1261, -0.4820,  0.7617,  0.0619, -0.3389,
         -0.0833,  0.1180, -0.0866, -0.3981, -1.1552,  0.2819, -0.2026,
          0.2612,  0.0320, -0.4613,  0.4293, -0.2500,  0.0336,  0.4155,
         -0.3490, -0.0778,  0.3419, -0.2882, -0.3467, -0.1443,  0.0714,
          0.5191,  0.1284, -0.3226, -0.1129, -0.2462,  0.8807, -0.2293,
          0.9080, -0.3914, -0.7117, -0.9196,  0.2674, -0.1359,  0.3978,
          0.1783, -0.1393,  0.2493, -0.2602, -0.0158, -0.0578, -0.2345,
         -0.1096,  0.4580,  0.1088,  0.2517,  0.1113,  0.2575, -0.2660,
         -0.0544, -0.1656, -0.1616, -0.5771, -0.0992, -0.5918,  0.5339,
          0.5556, -1.0867, -0.5307, -0.0200, -0.2505,  0.2330, -0.0214,
         -0.2999,  0.4987, -0.0848, -0.1613,  0.0049, -0.0145,  0.7184,
          0.6865,  0.2516,  0.6475, -0.0461,  0.5273,  0.5403, -0.3897,
          0.1155, -0.1748,  0.7277,  0.7136,  0.9828, -0.5179, -0.3025,
          0.1896,  0.1327,  0.0597,  0.1141,  0.1041,  0.1609, -0.5633,
         -0.2694, -0.2967, -0.6448,  0.2855,  0.0271, -0.2142, -0.1315,
         -0.1236, -0.2056, -0.1973,  0.1928, -0.0393, -0.4025, -0.5439,
          0.3569,  0.3930, -0.2932,  0.3049,  0.1190,  0.2337,  0.7675,
         -0.0723,  0.3622, -0.1088,  0.3370, -0.4724,  0.0665, -0.4085,
         -0.6112, -0.3083, -0.0142, -0.2328, -0.1839, -0.4621, -0.5118,
         -0.2060,  0.0197,  0.9506,  0.1890, -0.3932, -0.8124, -0.3948,
         -0.0645,  0.0282,  0.0078, -0.4813, -0.6527, -0.3910, -0.3695,
         -0.6337, -0.4388, -1.0729, -0.1245, -0.3699, -0.1060]), Parameter containing:
tensor([[[[ 1.7690e-02]],

         [[ 1.0148e-02]],

         [[ 2.4345e-03]],

         ...,

         [[ 1.3340e-02]],

         [[ 3.8968e-03]],

         [[-8.1534e-03]]],
```

```
[[[-1.3515e-02]],

 [[ 4.8309e-03]],

 [[-1.0365e-02]],

 ...,

 [[ 7.0203e-02]],

 [[-1.3361e-02]],

 [[-2.4033e-02]]],


[[[ 2.1483e-02]],

 [[-1.9623e-02]],

 [[ 4.3003e-03]],

 ...,

 [[-4.0974e-03]],

 [[ 2.4187e-02]],

 [[-6.9473e-03]]],


...,


[[[ 1.3122e-02]],

 [[-1.3035e-02]],

 [[-4.1658e-02]],

 ...,

 [[ 2.4511e-02]],

 [[-1.2560e-02]],

 [[-8.9364e-03]]],
```

```
         [[[ 2.8043e-02]],

          [[ 1.8019e-02]],

          [[ 2.6315e-02]],

          ...,

          [[-2.1704e-02]],

          [[-1.1877e-03]],

          [[-2.9643e-03]]],


         [[[ 7.3524e-03]],

          [[-4.3344e-03]],

          [[ 5.1834e-03]],

          ...,

          [[ 2.4892e-02]],

          [[ 5.4057e-02]],

          [[-9.1578e-03]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.8600, -0.7804, -0.6507, -0.8885, -0.8728, -0.9873, -0.5299,
        -0.9593, -0.4747, -0.6612, -0.1577, -1.0089, -0.7887, -1.0917,
        -0.5629, -0.8837, -0.6882, -0.7316, -0.8187, -0.6349, -0.9513,
        -0.7788, -0.9502, -0.8220, -0.6691, -0.6324, -0.8597, -0.9439,
```

```
        -0.8029, -0.6876, -1.0921, -0.6953, -0.8858, -0.9901, -0.7493,
        -1.0997, -0.6617, -0.7448, -0.7608, -0.6965, -0.7912, -0.6047,
        -0.7222, -1.2632, -0.6567, -1.2474, -0.8114, -0.6083, -0.6805,
        -0.5091, -0.8287, -0.6135, -0.8155, -0.5974, -0.8996, -1.3089,
        -0.8425, -0.8074, -0.8539, -0.7984, -0.8433, -0.4926, -1.0137,
        -1.0281, -0.9347, -0.9733, -1.1689, -0.7007, -0.7904, -0.8961,
        -0.7664, -0.8919, -0.7873, -0.6953, -0.7049, -0.6739, -0.6132,
        -0.5834, -0.8008, -0.8477, -0.8877, -0.6165, -0.8450, -1.3024,
        -0.5894, -0.6669, -0.8119, -0.8142, -1.0303, -0.5437, -0.9215,
        -1.0344, -0.6817, -0.8146, -0.6440, -0.9229, -0.8224, -0.8150,
        -0.8394, -0.6580, -0.8474, -0.6552, -0.7033, -0.6779, -0.9955,
        -0.7971, -0.6232, -0.5877, -0.8959, -0.7998, -0.7467, -0.6090,
        -0.9066, -0.7397, -0.8536, -0.7736, -1.2146, -0.9253, -1.1422,
        -0.6971, -0.7406, -0.6729, -1.2443, -0.8391, -0.7871, -0.9472,
        -0.7689, -0.7230, -1.0291, -0.6426, -0.6452, -0.5890, -0.7809,
        -0.5483, -0.7361, -1.2930, -1.1781, -1.0189, -0.8563, -0.6921,
        -0.8153, -0.8113, -0.5934, -0.5339, -0.7571, -0.8355, -0.8373,
        -1.6074, -0.8920, -0.6809, -0.6114, -1.0282, -0.5907, -1.1598,
        -0.7170, -1.3246, -0.9752, -0.6366, -0.4820, -0.6467, -0.8413,
        -0.8512, -0.5617, -0.8397, -0.7904, -0.7747, -0.8058, -0.5588,
        -0.7409, -0.8150, -0.7230, -0.6320, -1.0495, -0.5927, -0.5663,
        -0.7573, -0.8776, -0.8381, -0.9123, -0.8890, -0.6368, -0.9564,
        -0.5594, -0.8141, -0.6710, -0.5500, -0.8412, -0.7220, -0.8481,
        -0.9731, -0.8471, -1.2994]), Parameter containing:
tensor([[[[-1.0272e-02, -5.7760e-04, -6.2740e-03],
          [-5.0777e-03,  3.5425e-03, -4.3724e-03],
          [-1.0584e-02, -2.2178e-03, -1.2314e-02]],

         [[-8.0882e-03,  1.1496e-02, -9.0464e-03],
          [-4.2311e-04,  1.3943e-02, -4.9200e-03],
          [-7.1756e-03, -5.0424e-03, -1.0369e-02]],

         [[-4.0401e-03,  5.6225e-03, -1.6367e-02],
          [ 3.6364e-03,  2.1750e-03, -6.8060e-03],
          [ 9.9808e-03,  1.3031e-02,  3.1039e-03]],

         ...,

         [[-4.6517e-03, -1.4739e-02, -8.9217e-03],
          [-1.0467e-02, -1.3091e-02, -1.1866e-02],
          [ 3.0766e-04, -1.6581e-02, -8.3980e-03]],

         [[ 2.7970e-03, -8.8811e-03, -6.3076e-05],
          [-1.8955e-02, -1.9048e-02, -8.6783e-03],
          [-4.1534e-03, -1.4525e-02,  5.7280e-03]],

         [[ 9.1410e-03, -4.6228e-03,  3.1426e-03],
          [ 5.9413e-05, -7.0278e-03,  8.1348e-03],
```

```
  [ 4.4525e-03, -2.0782e-02,  1.1096e-02]]],


[[[-1.5678e-03, -1.1308e-02, -7.6600e-04],
  [-3.8677e-03, -2.4549e-02, -5.6674e-03],
  [-3.6618e-04, -2.5656e-03,  8.4074e-03]],

 [[ 1.3804e-02,  2.0594e-02,  2.0751e-02],
  [ 4.0933e-03,  1.6194e-02,  2.8815e-03],
  [ 2.6193e-03, -9.3285e-03, -3.3706e-03]],

 [[ 2.0442e-04, -1.2306e-02,  1.8389e-03],
  [-3.4500e-04,  2.9693e-03, -3.6434e-04],
  [ 1.3615e-02,  3.4774e-04,  1.5842e-02]],

 ...,

 [[ 1.3366e-02, -4.9991e-03,  4.2651e-03],
  [-6.7472e-03, -6.4603e-04, -9.2810e-04],
  [ 8.9586e-03,  2.1602e-03,  2.4181e-04]],

 [[-2.1511e-02, -1.1063e-02, -1.4425e-02],
  [-3.5235e-03, -4.4109e-03, -5.2861e-03],
  [-7.1103e-03, -1.3883e-02, -5.4027e-03]],

 [[ 2.2556e-03, -5.3224e-03,  5.8783e-03],
  [-6.1551e-03, -5.0543e-03,  2.7860e-03],
  [ 7.4522e-03, -8.6505e-03,  5.7815e-03]]],


[[[-2.2360e-03,  1.6161e-02, -1.7905e-03],
  [ 1.7461e-02, -1.7376e-04,  8.5467e-03],
  [-1.3107e-02,  1.5809e-02, -1.6064e-02]],

 [[-1.0146e-02, -4.4910e-03,  5.4883e-05],
  [-8.6125e-03, -2.5347e-03, -1.0164e-02],
  [ 1.6859e-03, -1.7260e-03, -6.7321e-05]],

 [[ 2.5767e-02, -5.4976e-03,  3.5366e-03],
  [ 1.3339e-02,  6.6421e-04,  5.1082e-03],
  [ 6.7899e-03,  2.8492e-03,  1.2752e-02]],

 ...,

 [[ 1.0700e-02, -6.7702e-03,  5.4549e-03],
  [-5.2611e-04, -7.4196e-03,  5.9797e-03],
  [ 1.3553e-02,  1.3454e-02,  1.8942e-02]],
```

```
 [[-2.0155e-03, -3.7937e-03, -6.8100e-03],
  [-1.2017e-02, -9.2876e-03, -8.4256e-03],
  [-8.7667e-03, -6.4274e-03, -1.5496e-02]],

 [[-5.0550e-03, -1.0371e-02, -7.3004e-03],
  [ 5.7045e-03, -1.7271e-03, -2.8601e-03],
  [-6.4744e-03, -1.2735e-02, -7.4642e-03]]],


...,


[[[-5.9662e-03, -1.1592e-02, -8.7799e-03],
  [-8.1021e-03, -2.0030e-02, -5.0764e-03],
  [-1.1386e-02, -4.6418e-03, -9.1803e-03]],

 [[-2.4986e-03, -2.5674e-03, -1.1134e-02],
  [-7.1066e-04,  7.9680e-03, -1.9123e-03],
  [ 4.5977e-03,  1.7361e-02, -1.0242e-03]],

 [[ 2.2169e-02,  2.1747e-02,  1.9215e-02],
  [ 2.1171e-03,  1.4827e-02,  1.7349e-02],
  [ 1.0882e-03,  2.8435e-05,  4.2188e-03]],

 ...,

 [[-2.5091e-02, -1.8832e-02, -1.8769e-02],
  [-1.4184e-02, -1.4106e-02, -1.2448e-02],
  [ 2.4191e-03, -5.5313e-03, -5.4926e-04]],

 [[ 3.6643e-02,  4.8616e-02,  2.4657e-02],
  [ 3.9747e-02,  1.0527e-02,  4.8810e-02],
  [ 1.6325e-02,  3.8729e-02,  1.0851e-02]],

 [[-1.0723e-02, -3.7010e-03, -1.6241e-02],
  [-4.7677e-03,  1.8100e-02, -6.6376e-03],
  [-1.2719e-02,  3.9595e-03, -1.9648e-02]]],


[[[-8.7284e-03, -6.7079e-03,  2.4111e-03],
  [ 2.2158e-03,  6.1247e-03,  2.3058e-03],
  [ 1.2268e-03, -5.3443e-03,  9.6119e-03]],

 [[ 2.1916e-02,  1.4835e-02,  1.8226e-02],
  [ 2.0795e-02,  2.2208e-02,  1.8249e-02],
  [ 1.5050e-02,  5.1883e-03,  2.4659e-02]],

 [[-1.5178e-02, -1.7377e-02, -1.5529e-02],
```

```
          [ 4.3945e-03,  1.0767e-02,  4.9382e-03],
          [-5.0887e-03,  8.4190e-03, -1.2929e-02]],

         ...,

         [[-2.2107e-03, -6.6739e-03,  7.5740e-03],
          [ 1.4470e-03, -2.9908e-03,  8.9971e-03],
          [-9.0209e-04, -5.9621e-03, -1.1560e-03]],

         [[ 1.1790e-02,  4.0304e-03, -2.9665e-04],
          [-4.0167e-03,  2.0621e-04, -6.9045e-03],
          [-7.6521e-04, -7.4008e-03,  1.8508e-04]],

         [[ 2.0279e-02, -2.3091e-03,  1.4891e-03],
          [ 8.8079e-03, -1.3081e-02, -1.5257e-03],
          [ 2.2656e-02, -3.2424e-03,  1.1810e-02]]],


        [[[-4.1700e-03,  1.9708e-02,  8.9472e-03],
          [ 4.4084e-03,  1.4306e-02,  2.0793e-03],
          [-1.1032e-02,  1.0885e-02, -7.4103e-04]],

         [[ 2.2010e-02,  3.4318e-02,  2.0147e-02],
          [ 2.3278e-02,  1.8009e-02,  2.6982e-02],
          [ 1.5635e-02,  3.4007e-02,  2.6012e-02]],

         [[-1.5053e-03, -1.9816e-02,  2.3180e-03],
          [ 2.7858e-03, -4.0348e-04, -2.3466e-03],
          [-4.2915e-03, -1.4269e-02,  5.2550e-03]],

         ...,

         [[-8.9005e-03, -1.0963e-02, -1.0136e-02],
          [-5.2578e-03, -6.6147e-03, -2.1870e-05],
          [-1.2817e-03,  2.0722e-03,  6.1423e-03]],

         [[-3.6050e-03, -3.3484e-03, -1.1169e-03],
          [ 3.3823e-03,  1.9707e-02,  6.2673e-03],
          [-2.6656e-03,  1.5057e-02, -3.6735e-03]],

         [[-1.2118e-03, -1.9363e-03, -5.7474e-03],
          [ 2.6437e-03, -1.1395e-02,  1.4434e-03],
          [ 2.7725e-03,  8.3711e-04, -6.6763e-03]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter containing:
tensor([-0.7173, -0.6833, -0.8410, -0.9232, -0.9244, -1.1371, -0.9175,
       -0.7375,  0.8215, -0.8585, -0.7648, -0.9725, -0.7991, -1.3199,
       -0.7595, -0.6794, -0.8570, -0.9996, -0.8401, -0.7856, -0.7836,
       -0.8468, -0.9818, -0.8397, -0.8407, -0.8204, -0.8218, -0.9487,
       -0.8518, -0.6648, -0.9082, -0.9785, -0.9763, -1.2374, -0.9400,
       -0.9141, -0.8638, -1.0113, -0.7533, -1.1275, -0.8064, -0.9554,
       -0.9903, -0.6855, -0.7729, -0.8934, -0.6663, -1.2057, -1.0908,
       -0.9254, -0.9166, -0.9401, -0.7975, -0.8744, -1.0414, -0.8733,
       -0.8640, -0.9612, -0.9214, -1.0265, -1.3565, -1.2081, -0.9073,
       -0.7657, -1.3240, -0.9559, -0.8168, -0.7602, -0.8222, -0.7167,
       -0.6939, -0.8243, -1.0238, -1.0182, -0.8464, -1.1919, -0.8013,
       -0.7758, -0.9432, -1.0624, -1.0318, -0.7691, -0.8086, -0.8287,
       -1.0766, -0.7280, -1.0131, -0.7760, -0.8370, -0.7981, -1.1592,
       -0.8052, -0.9542, -0.6853, -0.6875, -0.8457, -0.7554, -0.8653,
       -1.0051, -0.8084, -0.8869, -0.6939, -1.0978, -0.9505, -1.2145,
       -0.8719, -0.8600, -0.9738, -0.7804, -1.1317, -0.8533, -0.8695,
       -0.7219, -0.6688, -0.7955, -0.6388, -0.9285, -0.8791, -1.0511,
       -1.0382, -0.9367, -0.9145, -0.7895, -1.1633, -0.8163, -1.1694,
       -0.9459, -1.0588, -0.8179, -0.9645, -0.6045, -0.9987, -0.7559,
       -0.5697, -0.8445, -1.1828, -0.9029, -0.6842, -0.8298, -0.7203,
       -1.0884, -0.7895, -0.9147, -0.6881, -0.9500, -1.0016, -0.8522,
       -0.8403, -0.8468, -1.3001, -0.8057, -1.0280, -0.8910, -0.5699,
       -0.9796, -1.3729, -0.9571, -0.8930, -0.6767, -0.7417, -0.8491,
       -0.9927, -0.8225, -0.7646, -0.8888, -0.7607, -1.1130, -0.7681,
       -0.3453, -0.7087, -0.7164, -0.7456, -0.9260, -0.7352, -0.7911,
       -0.8391, -0.8921, -1.0673, -0.6724, -0.8614, -0.9294, -0.7415,
```

```
        -1.0803, -0.7066, -0.8145, -1.0758, -0.9964, -0.7260, -1.2035,
        -0.0888, -0.9800, -0.8885, -1.0499, -0.8425, -0.5935, -0.9832,
        -0.8503, -0.9106, -0.7620, -1.0147, -0.7910, -0.8907, -1.0835,
        -1.0067, -0.7692, -0.9473, -0.8562, -1.0275, -0.9976, -0.6838,
        -0.8051, -0.7559, -0.7067, -1.0788, -0.8815, -0.8450, -0.9059,
        -0.7949, -1.1075, -0.7093, -0.5922, -1.3274, -0.8824, -0.7649,
        -1.0121, -0.4561, -0.7814, -0.7312, -1.2899, -1.3374, -0.7852,
        -0.7017, -0.8782, -1.0839, -0.8371, -0.7031, -1.2628, -0.9338,
        -1.2233, -0.8745, -1.0994, -0.8801, -0.7905, -1.0637, -1.1780,
        -0.6834, -1.2066, -0.8776, -0.6871, -0.9627, -1.0099, -0.4328,
        -0.7806, -0.6560, -0.9176, -0.7326, -0.9086, -0.6296, -0.9683,
        -0.9029, -0.7292, -0.7196, -0.8011, -1.0679, -0.5269, -0.9818,
        -0.9052, -0.8448, -0.8608, -1.0178, -0.9506, -0.9213, -0.8197,
        -0.8876, -0.7644, -1.1235, -0.8605, -1.2250, -1.0405, -0.7099,
        -0.8648, -0.8459, -0.8395, -0.7971, -0.9857, -0.7420, -0.8060,
        -0.9676, -1.0635, -0.7646, -0.9085, -0.6918, -0.9937, -0.9760,
        -0.8920, -1.1072, -1.5347, -0.9411, -0.9165, -0.9565, -0.9228,
        -0.6472, -0.6482, -0.7874, -1.0990, -1.0827, -0.8503, -0.7554,
        -0.7187, -1.0106, -0.9283, -0.8364, -0.7541, -0.8881, -0.7817,
        -0.6678, -0.8696, -0.9532, -0.9382, -0.8953]), Parameter containing:
tensor([[[[ 1.3765e-02]],

         [[ 2.7339e-02]],

         [[-9.0542e-03]],

         ...,

         [[ 4.3343e-03]],

         [[ 8.3372e-03]],

         [[-4.6769e-03]]],


        [[[ 8.1026e-03]],

         [[-1.5880e-02]],

         [[-6.0513e-03]],

         ...,

         [[-3.7032e-03]],

         [[-1.4191e-02]],

         [[-3.5323e-02]]],
```

```
[[[-1.5870e-02]],

 [[ 2.2802e-02]],

 [[ 2.7343e-02]],

 ...,

 [[-1.8475e-02]],

 [[ 1.3847e-01]],

 [[ 4.7423e-02]]],


...,


[[[ 3.6135e-02]],

 [[-4.3132e-02]],

 [[ 8.1385e-03]],

 ...,

 [[ 1.5515e-02]],

 [[-1.6361e-02]],

 [[-2.2165e-02]]],


[[[-1.7188e-02]],

 [[ 2.3621e-04]],

 [[ 1.8670e-02]],

 ...,

 [[-8.3817e-03]],

 [[-2.3623e-02]],

 [[-4.2265e-03]]],
```

```
        [[[-4.3977e-03]],

         [[ 1.4633e-02]],

         [[-2.9727e-02]],

         ...,

         [[-3.8511e-02]],

         [[ 6.2658e-03]],

         [[-2.5219e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.3500, -0.6394, -0.3086, -0.6920, -0.1650, -0.5446, -0.3971,
        -0.7730, -0.6139, -0.7671, -0.6939,  0.0460, -0.3220, -0.6414,
        -0.2968, -0.4538, -0.8278, -0.6999, -0.6120, -0.1941, -0.0558,
        -0.6004, -0.6951, -0.7007, -0.5886, -0.3919, -0.4412, -0.5924,
        -0.0347, -0.4825, -0.2937, -0.4751, -0.5463, -0.5533, -0.5930,
        -0.4017, -0.6942, -0.7689, -0.4976,  0.0376, -0.4841, -0.3045,
        -0.7016,  0.0699, -0.5096, -0.7757, -0.7521, -0.5266, -0.3431,
        -0.4250, -0.7847, -0.7599, -0.7801, -0.2261, -0.0086, -0.4574,
        -0.6791, -0.3337, -0.1695, -0.4322, -0.6859, -0.2576, -0.9748,
        -0.3335, -0.1972, -0.6727, -0.3846, -0.1934, -0.4359, -0.5661,
        -0.9285, -0.4635, -0.7990, -0.7473, -0.6730, -0.3047, -0.5829,
        -0.1606, -0.5368, -0.3919, -0.2660, -0.4788, -0.8849, -0.5512,
        -0.3908, -0.8187, -0.2686, -0.4080, -0.2559, -0.7866, -0.6685,
        -0.7847, -0.6675, -0.3914, -0.4417, -0.5344, -0.7081, -0.3640,
        -0.4724, -0.6561, -0.5676, -0.7151, -0.5995, -0.8323, -0.7128,
        -0.4609, -0.6058, -0.5457, -0.8629, -0.7651, -0.4404, -0.9782,
        -0.3824, -0.4590, -0.7897, -0.6305, -0.5274, -0.7808, -0.5975,
```

```
        -0.1970, -0.7361, -0.4914, -0.6942, -0.7178, -0.2608, -0.5757,
        -0.5579,  2.2388, -0.5814, -0.3332, -0.5092, -0.9450, -0.6513,
        -0.5909, -0.3612, -0.7222, -0.5546, -0.9001, -0.2604, -0.5148,
        -0.5492, -0.6324, -0.6142, -0.2544, -0.6576, -0.4559, -0.1833,
        -0.1799, -0.6440, -0.5662, -0.7824, -0.5850, -0.2958, -0.1124,
        -0.8356, -0.6495, -0.3766, -0.4021, -0.5674, -0.7034,  0.2597,
        -0.5868, -0.7150,  0.2589, -0.5031, -0.8458, -0.7146, -0.6801,
        -0.2872,  0.2986, -0.4034, -0.7879, -0.5912, -0.8815, -0.6752,
        -0.8782, -0.5006, -0.5938, -0.6740, -0.2722, -0.6541, -0.4424,
        -0.5103, -0.5604, -0.6815, -0.6063, -0.1544, -0.2421, -0.1239,
        -0.5393, -0.5172, -0.4288]), Parameter containing:
tensor([[[[ 1.4244e-02, -1.4156e-02,  1.0407e-02,  ...,  6.9201e-03,
            -9.2419e-03, -1.5335e-02]],

          [[-1.2224e-02,  7.6798e-03, -1.1806e-02,  ...,  1.6422e-02,
            -1.9567e-04, -2.8754e-03]],

          [[ 2.4118e-02, -7.7642e-03, -1.9454e-02,  ..., -2.2359e-02,
            -6.1468e-03, -2.5123e-02]],

          ...,

          [[-2.7565e-02, -1.4537e-03, -4.8675e-03,  ..., -6.2123e-03,
             2.1180e-02, -1.0045e-03]],

          [[ 8.2559e-03, -1.9151e-02, -7.5451e-03,  ..., -2.1018e-03,
            -4.3196e-03,  4.1095e-03]],

          [[-2.4168e-02, -1.9277e-02, -4.8974e-03,  ...,  1.3209e-02,
             2.9086e-03, -4.4271e-03]]],


        [[[ 4.4997e-03,  6.5568e-03, -2.5365e-03,  ...,  1.6851e-03,
             5.9518e-03, -1.8812e-03]],

          [[ 6.3039e-03,  1.5583e-02,  1.0216e-02,  ..., -4.5017e-03,
            -2.2517e-03, -1.9762e-02]],

          [[-3.0884e-02, -3.0191e-03, -5.4126e-03,  ...,  2.2393e-02,
             1.1723e-02, -6.8726e-03]],

          ...,

          [[ 6.2937e-04,  4.0137e-03,  1.0474e-02,  ..., -6.8448e-03,
            -2.3120e-02, -9.5895e-03]],

          [[-2.1539e-02, -1.4233e-02, -3.1452e-03,  ..., -1.6118e-03,
            -8.7306e-03, -2.7747e-03]],
```

```
 [[-5.3355e-03, -6.9912e-03,  8.9551e-03,  ...,  1.4549e-02,
   -1.8328e-02,  2.2275e-03]]],


[[[-1.4776e-02, -1.7883e-02,  3.0986e-03,  ...,  1.0586e-02,
   -1.5992e-02, -1.0848e-02]],

 [[-5.6990e-03, -6.3538e-03,  1.0430e-03,  ..., -4.0644e-03,
   -2.5940e-02, -2.1052e-02]],

 [[-2.1026e-02,  3.4191e-03,  7.3892e-04,  ...,  7.5873e-07,
   -5.7693e-03, -1.0729e-03]],

 ...,

 [[ 1.3131e-02,  9.0111e-03,  5.6428e-03,  ..., -4.6825e-03,
    6.6085e-03,  9.2072e-03]],

 [[ 1.9690e-02,  3.6770e-02,  1.2073e-02,  ..., -1.7371e-02,
   -3.2416e-02, -1.2901e-02]],

 [[-8.6864e-03, -1.5728e-02,  1.1563e-03,  ..., -4.0228e-03,
   -2.6407e-03, -1.6715e-02]]],


...,


[[[-7.2038e-03, -2.5744e-03, -4.1641e-03,  ..., -1.6287e-02,
   -9.2820e-03,  1.3410e-02]],

 [[ 1.9542e-02,  6.6543e-03,  1.1137e-03,  ...,  1.3721e-02,
    1.9557e-03, -7.6444e-03]],

 [[ 3.1231e-02, -8.5336e-04, -1.2493e-02,  ...,  4.5295e-02,
   -1.4205e-02, -2.6474e-02]],

 ...,

 [[ 1.3169e-03, -1.2426e-02, -2.0922e-02,  ..., -6.5131e-03,
    5.1828e-03, -1.2204e-02]],

 [[-2.2323e-03, -1.3590e-02,  5.3296e-03,  ...,  5.8091e-03,
   -2.7815e-02, -9.7807e-03]],

 [[-1.1610e-02, -1.2467e-02, -5.7109e-03,  ...,  1.9648e-03,
    5.8474e-03, -5.1804e-04]]],
```

```
          [[[-7.4225e-04, -6.9810e-03, -6.3260e-04,  ..., -9.3526e-03,
              2.4422e-03,  4.4159e-03]],

           [[ 1.6766e-02,  4.3398e-03, -1.3771e-03,  ..., -1.3426e-02,
             -8.8115e-03,  2.7639e-03]],

           [[-1.4112e-02,  1.2351e-03, -2.0481e-02,  ...,  7.2324e-03,
              1.3749e-04,  2.1629e-02]],

           ...,

           [[ 8.7515e-03,  6.9722e-04,  2.4991e-03,  ..., -1.2108e-02,
             -1.1051e-02, -1.1086e-02]],

           [[ 1.1058e-02,  1.5347e-02,  1.0257e-02,  ..., -1.5002e-02,
             -1.2225e-02,  1.5187e-03]],

           [[-5.0092e-03,  2.1232e-03, -6.4821e-03,  ...,  3.5289e-04,
             -1.7624e-03, -1.1622e-02]]],


          [[[-7.1552e-03, -3.3109e-03, -2.9029e-03,  ..., -4.0862e-03,
              5.5568e-04, -1.0050e-02]],

           [[ 1.7402e-03, -8.8494e-03, -9.5364e-03,  ...,  1.2726e-02,
              3.2156e-02,  2.2174e-02]],

           [[-1.0155e-02, -1.0392e-02, -1.9035e-03,  ...,  5.5782e-03,
             -1.1623e-03,  1.3661e-02]],

           ...,

           [[ 1.7994e-02,  9.5025e-03,  1.5442e-02,  ..., -6.0285e-03,
             -1.5218e-02, -4.0074e-03]],

           [[-5.6418e-03, -1.5489e-02, -6.1853e-03,  ..., -1.1749e-03,
              1.4371e-02,  1.4315e-02]],

           [[ 2.1682e-02,  4.3815e-04,  9.0688e-04,  ..., -2.1019e-02,
             -2.4367e-02, -1.4892e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.5636, -0.6795, -0.7026, -0.3163, -0.6502, -0.2749, -1.0867,
        -0.6643, -0.5954, -0.0961, -0.0856, -0.6898, -0.3800, -0.3043,
        -0.9198, -0.5257, -0.5976, -0.5585, -0.3781, -0.7138,  0.1709,
        -0.4400, -0.4726, -0.3809, -0.5189, -0.5835, -0.5251, -0.5197,
        -0.6668, -0.5016, -0.7054, -0.4268,  0.1335, -0.3934, -0.3720,
        -0.5632, -0.6111, -0.5925, -0.2171, -0.4070, -0.4274, -0.2594,
        -0.3325, -0.7387, -0.6160,  0.0108, -0.1066, -0.5293, -0.6192,
        -0.7379, -0.3946,  0.0812, -0.5998, -0.5956, -0.2013, -0.3445,
        -0.5416, -0.0741, -0.1245, -0.2820, -0.5346, -0.2717,  0.0029,
        -0.4061, -0.3109, -0.3501, -0.0679, -0.4740, -0.0457, -0.3359,
        -0.4749, -0.5589, -0.1809, -0.0499, -0.1688, -0.9717, -0.5080,
        -0.1708, -0.3726, -0.3255, -0.4779, -0.7093, -0.5739, -0.1111,
        -0.0958,  0.0669, -0.2270, -0.0911, -0.5809,  0.1537, -0.4083,
        -0.2079,  0.0404, -0.2697, -0.2655, -0.7483, -0.3466, -0.6541,
        -0.7021,  0.1052, -0.4745, -0.4228, -0.4666, -0.4131, -0.3432,
        -0.2891, -0.4834, -0.3745, -0.5258, -0.5559, -0.3149, -0.3423,
        -0.6617, -0.7201, -0.3053, -0.3396,  0.0324, -0.2318, -0.5760,
        -0.2833, -0.5874, -0.1441, -0.8888,  0.0503, -0.3192, -0.3666,
        -0.6189, -0.5403, -0.5988, -0.6020, -0.2516, -0.2955, -0.4668,
         0.0749, -0.4506, -0.5039, -0.0107, -0.5512, -0.2952, -0.5259,
        -0.5774,  0.1047, -0.4061, -0.7994, -0.7243, -0.6055,  0.0847,
        -0.6883, -0.3725,  0.0138, -0.3902,  0.1753,  2.9035, -0.6473,
         0.3682, -0.2896, -0.5412, -0.3709,  0.2055, -0.3830, -0.1310,
        -0.4632, -0.2194,  0.1356, -0.0935, -0.2053, -0.1073, -0.5284,
        -0.5448, -0.5609, -0.3751, -0.4907, -0.4382,  0.6983, -0.4870,
        -0.6055, -0.2413, -0.3872, -0.0581, -0.2952, -0.2612, -0.1806,
        -0.9401, -0.5440, -0.4325, -0.6423, -0.2455, -0.2103, -0.0104,
        -0.5565, -0.5225, -0.7442]), Parameter containing:
tensor([[[[-1.7625e-03],
          [-2.4767e-02],
          [-5.6810e-04],
          ...,
          [ 2.1643e-03],
          [-1.0925e-02],
          [-2.3736e-02]],

         [[-1.9332e-02],
          [-3.9892e-04],
```

```
  [-8.1151e-03],
  ...,
  [-4.3738e-03],
  [ 2.1863e-03],
  [ 5.7849e-03]],

 [[ 1.4833e-02],
  [-1.4461e-03],
  [ 7.9768e-03],
  ...,
  [ 1.1194e-03],
  [-7.2513e-03],
  [-2.0861e-02]],

 ...,

 [[ 1.0406e-02],
  [ 6.2353e-03],
  [ 1.3261e-02],
  ...,
  [-3.0067e-02],
  [-1.1480e-03],
  [-2.0758e-02]],

 [[-1.1532e-02],
  [-1.6888e-02],
  [ 2.5861e-02],
  ...,
  [ 1.7977e-02],
  [ 7.7242e-03],
  [ 3.1774e-03]],

 [[ 1.3447e-02],
  [ 3.5315e-03],
  [ 1.7023e-02],
  ...,
  [ 4.0129e-03],
  [-5.2706e-03],
  [-1.0618e-03]]],


[[[ 3.5386e-02],
  [ 1.9516e-02],
  [-1.3367e-02],
  ...,
  [-2.0524e-02],
  [-2.8993e-02],
  [-4.9880e-02]],
```

```
 [[-3.1023e-02],
  [-3.4372e-02],
  [-2.2236e-02],
  ...,
  [-1.0404e-02],
  [-1.3425e-02],
  [-8.1599e-04]],

 [[-1.3997e-02],
  [-1.5118e-02],
  [-2.9903e-03],
  ...,
  [ 1.7701e-02],
  [ 2.2547e-02],
  [ 1.2769e-02]],

 ...,

 [[-3.0442e-02],
  [-1.2400e-02],
  [-2.4470e-03],
  ...,
  [ 3.7672e-03],
  [ 1.3629e-02],
  [ 4.7144e-02]],

 [[-2.5742e-02],
  [-2.9419e-02],
  [-1.9106e-02],
  ...,
  [-9.6355e-03],
  [ 6.2202e-03],
  [ 1.4646e-02]],

 [[-2.6130e-02],
  [-2.9623e-02],
  [-1.6056e-02],
  ...,
  [-6.1769e-03],
  [-8.7950e-03],
  [-8.2645e-03]]],


[[[-7.0926e-03],
  [ 5.5194e-03],
  [ 1.3119e-02],
  ...,
```

```
  [ 2.5774e-02],
  [ 1.4798e-03],
  [ 1.7213e-02]],

 [[-1.6528e-03],
  [ 1.9573e-03],
  [-1.1550e-02],
  ...,
  [-6.3418e-03],
  [ 2.1293e-02],
  [ 1.7616e-02]],

 [[ 2.7036e-03],
  [ 1.8237e-02],
  [ 9.5015e-03],
  ...,
  [ 3.2174e-03],
  [-5.5689e-03],
  [-2.8143e-03]],

 ...,

 [[ 1.1173e-02],
  [-2.1773e-03],
  [ 2.4896e-03],
  ...,
  [ 2.0255e-02],
  [ 2.1412e-03],
  [ 1.7782e-02]],

 [[-5.0312e-03],
  [ 1.1441e-02],
  [ 3.0361e-03],
  ...,
  [ 2.3431e-02],
  [ 1.3617e-03],
  [ 2.6758e-02]],

 [[-4.3113e-04],
  [-1.0820e-02],
  [-1.6875e-02],
  ...,
  [-9.1984e-03],
  [-7.3377e-03],
  [ 3.7579e-03]]],


...,
```

```
[[[-1.6421e-02],
  [-9.7255e-03],
  [-1.0995e-02],
  ...,
  [ 1.4241e-02],
  [ 1.6818e-02],
  [ 9.1426e-04]],

 [[-1.4056e-03],
  [-3.3638e-03],
  [-1.8081e-03],
  ...,
  [-1.5816e-02],
  [-1.6758e-03],
  [ 3.1753e-03]],

 [[-1.1913e-02],
  [-5.4323e-03],
  [ 5.8603e-04],
  ...,
  [-1.9885e-04],
  [ 3.9423e-03],
  [ 1.1119e-03]],

 ...,

 [[-1.2320e-02],
  [-1.8302e-03],
  [-2.3203e-02],
  ...,
  [-1.6173e-02],
  [-8.4697e-03],
  [-1.0620e-04]],

 [[-4.0026e-02],
  [-2.6332e-03],
  [-2.0736e-02],
  ...,
  [ 3.8277e-03],
  [ 2.6053e-02],
  [ 1.3714e-02]],

 [[ 9.5612e-03],
  [-1.3451e-03],
  [ 4.1184e-03],
  ...,
```

```
  [-4.4932e-03],
  [-5.0491e-03],
  [ 7.4332e-03]]],


[[[-6.6721e-03],
  [ 3.4015e-03],
  [ 2.1524e-04],
  ...,
  [-1.1209e-02],
  [ 6.4111e-03],
  [-7.9409e-03]],

 [[ 3.1014e-03],
  [ 5.4328e-03],
  [ 1.3520e-03],
  ...,
  [ 4.6038e-03],
  [ 9.6838e-03],
  [ 1.0440e-02]],

 [[-1.3565e-02],
  [-2.9225e-03],
  [-5.8561e-03],
  ...,
  [-2.0841e-03],
  [ 1.0376e-02],
  [ 2.5686e-03]],

 ...,

 [[ 5.1230e-03],
  [ 3.1403e-02],
  [ 1.1400e-02],
  ...,
  [ 2.4920e-03],
  [-1.6560e-02],
  [ 2.4333e-03]],

 [[-2.8736e-02],
  [ 3.2861e-03],
  [-1.1119e-02],
  ...,
  [-1.2356e-02],
  [-4.4503e-04],
  [-2.4446e-03]],

 [[ 7.1914e-03],
```

```
   [-2.2904e-03],
   [ 6.5274e-03],
   ...,
   [ 1.2654e-02],
   [-5.9705e-03],
   [ 1.2548e-03]]],


 [[[ 8.2126e-03],
   [ 3.3815e-02],
   [ 2.4790e-02],
   ...,
   [ 1.3716e-02],
   [ 1.3390e-02],
   [ 1.9485e-02]],

  [[-1.6670e-02],
   [-1.2077e-02],
   [-1.2152e-02],
   ...,
   [-1.8286e-02],
   [ 3.2871e-04],
   [-2.9855e-03]],

  [[ 3.9815e-03],
   [-8.9900e-03],
   [-8.3359e-03],
   ...,
   [-9.4663e-03],
   [-2.1679e-03],
   [ 5.7843e-03]],

  ...,

  [[ 4.9722e-03],
   [ 8.9304e-03],
   [ 1.0093e-02],
   ...,
   [ 1.7318e-02],
   [ 1.2331e-02],
   [ 1.7109e-03]],

  [[-1.4604e-02],
   [-1.6390e-02],
   [-2.3536e-02],
   ...,
   [-2.9342e-02],
   [-1.9120e-02],
```

```
          [-2.1611e-02]],

         [[ 1.2191e-02],
          [-4.6415e-03],
          [-1.5522e-02],
          ...,
          [-1.8580e-02],
          [-7.9420e-04],
          [ 1.7106e-02]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.0452, -0.8475, -0.6076, -0.9300, -1.1250, -0.9119, -0.8210,
        -1.0400, -0.3856, -0.7857, -0.7342, -0.6489, -0.9634, -0.7317,
        -0.8858, -0.8579, -0.8076, -1.0026, -0.4301, -0.4039, -0.6962,
        -0.6166, -0.9448, -0.7590, -0.8693, -0.4040, -1.0149, -0.4301,
        -0.9092, -0.9303, -0.4567, -1.0374, -0.9574, -1.0926, -0.6812,
        -0.4274, -0.6468, -0.8050, -0.5414, -0.6481, -0.9922, -0.9876,
        -1.0401, -0.7515, -0.4613, -1.1418, -0.4993, -0.9158, -0.2965,
        -1.1650, -0.8028, -0.8200, -0.8847, -0.8719, -0.7132, -0.9969,
        -0.7032, -0.6842, -0.7657, -0.6914, -0.8429, -0.4669, -0.8792,
        -0.5396, -0.9743, -0.3926, -0.6805, -0.8683,  0.5108, -0.7770,
        -0.5097, -0.7612, -0.7115, -0.9590, -0.5691, -0.7546, -0.1983,
        -0.8183, -0.8280, -1.2421, -1.1092, -0.8012, -1.0312, -1.3719,
        -0.1631, -0.4689, -0.8569, -0.8305, -0.3929, -0.5944, -0.8221,
        -0.5514, -0.8883, -0.6057, -0.5263, -0.5913, -0.7038, -0.3132,
        -0.4518, -0.4349, -0.8149, -0.7264, -0.8201,  0.1084, -0.5098,
        -0.6942, -0.4400, -0.7908, -1.0509, -0.6609, -0.7734, -0.6735,
        -0.8714, -0.7376, -0.4786, -1.1080, -0.6465, -0.5048, -0.1084,
         0.1357, -1.2416, -0.6619, -0.8312, -0.6906, -0.3466, -0.2480,
        -1.4411, -0.8723, -0.9869, -0.8195, -0.2366, -1.3545, -0.8748,
        -0.3228, -0.9118, -0.5203, -0.9445, -0.8458, -0.7752, -0.4045,
        -0.8264, -0.3925, -0.8147, -1.0056, -0.7548, -0.7124,  0.0904,
        -0.8535, -0.9752, -0.6571, -0.5656, -0.6653, -0.6212, -0.9342,
        -0.7874, -0.9319, -1.0165, -0.2029, -0.8281, -0.6226, -0.5445,
```

```
        -0.9910, -0.3401, -0.7127, -0.9640, -0.8288, -0.9310, -0.4067,
        -0.8015, -0.6314, -0.6246, -0.6692, -0.5425, -1.0032, -0.9364,
        -0.6851, -1.1567, -0.7927, -0.6969, -0.9907, -1.0090, -0.9554,
        -0.0884, -0.6105, -0.6239, -0.7175, -0.5303, -0.9070, -0.2473,
        -0.3413, -0.6947, -0.5523]), Parameter containing:
tensor([[[[-2.0379e-02, -8.0766e-04, -2.1584e-02],
          [ 1.3432e-03, -7.1682e-03,  3.3130e-04],
          [-7.2126e-03, -4.2986e-03, -6.5026e-03]],

         [[ 1.1585e-03,  5.5601e-03, -8.8894e-03],
          [-4.4478e-03, -3.6045e-03, -1.0306e-02],
          [-7.5619e-03, -1.0448e-03, -1.4009e-02]],

         [[-1.8490e-03,  2.7419e-03, -2.7080e-03],
          [-2.0340e-03,  1.2709e-02,  9.8442e-04],
          [ 2.8450e-03,  3.1539e-03, -5.4972e-03]],

         ...,

         [[ 2.7168e-02, -1.0482e-04,  2.0142e-02],
          [ 5.8911e-03, -5.0331e-03,  4.0982e-03],
          [-3.2208e-03, -2.2260e-02, -2.9772e-03]],

         [[-1.7610e-02, -7.3466e-03, -1.8863e-02],
          [-1.0019e-03, -3.3721e-04, -1.5867e-03],
          [-1.6342e-04, -1.1605e-03, -1.9824e-04]],

         [[ 3.9637e-03, -8.1479e-03, -7.1755e-03],
          [ 1.1972e-02, -9.9920e-03,  5.5092e-03],
          [ 5.6587e-03, -4.3939e-03, -1.1865e-02]]],


        [[[ 2.6501e-03,  1.7280e-02,  1.2748e-02],
          [ 1.9837e-02,  1.7578e-02, -4.3250e-03],
          [-5.8657e-02, -2.1422e-03,  1.2073e-02]],

         [[ 3.7970e-02, -9.6427e-03, -4.5524e-02],
          [ 8.9902e-03, -1.7384e-03, -2.3406e-03],
          [ 6.8792e-02, -7.5519e-03,  2.0390e-02]],

         [[ 3.7566e-03,  1.1200e-02, -6.8245e-03],
          [ 2.1289e-02,  2.0693e-02, -4.9259e-03],
          [-1.4719e-02, -6.3852e-04, -2.8308e-02]],

         ...,

         [[ 5.3071e-04,  2.3524e-02,  3.7600e-02],
          [-4.6924e-03,  1.3652e-02,  1.0998e-02],
```

```
  [-2.6676e-02,  2.1480e-02,  4.1711e-03]],

 [[ 1.7448e-02,  6.0911e-04, -8.9029e-03],
  [-1.3206e-03, -7.2660e-03, -4.9112e-03],
  [-6.4777e-03, -8.3121e-03, -4.3508e-03]],

 [[-1.4438e-02, -1.5479e-03,  1.0017e-02],
  [-2.8438e-02, -2.0222e-03,  1.0289e-02],
  [-3.4703e-02, -8.0015e-04,  9.9540e-03]]],


[[[-7.7720e-02, -1.5787e-02, -7.2921e-02],
  [ 3.7975e-03, -7.7163e-03,  5.0000e-03],
  [ 2.2945e-02,  6.3138e-03,  3.5790e-02]],

 [[-5.4443e-03,  5.1716e-03, -2.4476e-02],
  [ 5.3686e-03,  5.5299e-03,  2.7932e-03],
  [-9.0733e-03, -8.0307e-03, -2.2830e-02]],

 [[ 1.3201e-02, -2.6312e-02,  1.5086e-02],
  [ 6.6120e-03,  4.2637e-03,  1.5542e-03],
  [-4.0322e-02, -3.8931e-03, -5.7725e-02]],

 ...,

 [[ 4.3600e-02, -7.0873e-02,  3.9902e-02],
  [ 3.9889e-02, -1.8464e-02,  2.6945e-02],
  [ 1.9744e-02, -1.5152e-02,  1.8931e-02]],

 [[ 9.7935e-03, -1.5317e-02,  2.1001e-02],
  [-7.1754e-03, -4.8613e-03, -4.9043e-03],
  [ 2.4968e-02, -1.8389e-02,  2.2371e-02]],

 [[ 1.2053e-02,  1.4011e-02, -1.0862e-02],
  [ 1.7608e-02,  7.2881e-03, -7.4945e-03],
  [ 2.5108e-02,  1.6153e-02, -2.1090e-02]]],


...,


[[[-2.3357e-02,  7.2535e-03, -1.1636e-02],
  [-1.0986e-02,  1.9076e-02, -7.0328e-03],
  [ 2.4274e-03,  1.6119e-02,  4.9327e-03]],

 [[ 9.2819e-03,  2.4439e-03,  2.9468e-03],
  [ 1.5981e-03, -4.6615e-03,  3.6505e-03],
  [-5.3861e-03, -6.6969e-03, -2.5528e-03]],
```

```
 [[ 2.4115e-03, -2.4502e-02, -9.2382e-03],
  [ 1.0000e-02,  3.9817e-03,  7.8147e-03],
  [ 1.4924e-02,  3.4671e-03,  4.4996e-03]],

 ...,

 [[ 5.5203e-03, -1.1111e-02,  4.5308e-03],
  [ 4.6472e-03,  1.4750e-03,  1.3601e-02],
  [-1.0631e-04, -1.4203e-02,  5.5951e-04]],

 [[ 2.2191e-02,  1.3287e-02,  1.6494e-02],
  [-1.1056e-02, -1.1589e-03, -1.1161e-02],
  [-1.9694e-02, -1.8816e-02, -2.3584e-02]],

 [[-3.7834e-03,  1.4195e-03, -6.5487e-03],
  [-1.0746e-02,  1.7323e-03, -5.5545e-03],
  [-3.9878e-03,  3.5860e-03, -3.7412e-03]]],


[[[-2.6780e-03, -2.1746e-02, -1.5384e-03],
  [-1.3774e-02, -1.1490e-02, -1.0378e-02],
  [-2.0252e-02, -2.0851e-02, -2.3272e-02]],

 [[ 8.2400e-03, -9.3566e-03,  1.1122e-02],
  [-8.0541e-04,  8.2704e-03, -2.3379e-03],
  [-1.1274e-02,  6.9066e-03, -3.2262e-03]],

 [[-2.8572e-02,  1.4368e-02, -2.3387e-02],
  [ 4.7084e-03,  3.8863e-03,  1.5712e-03],
  [-1.7339e-02, -7.9694e-04, -1.0427e-02]],

 ...,

 [[-1.4510e-02,  7.7106e-03, -1.7919e-02],
  [-2.2780e-02, -6.2990e-03, -1.7275e-02],
  [-4.0151e-04,  1.3626e-03, -5.2367e-03]],

 [[ 1.1852e-02, -8.2626e-03,  1.6632e-02],
  [-3.4508e-03, -9.7580e-05, -3.2892e-04],
  [ 5.1600e-03, -1.0096e-02,  5.8271e-03]],

 [[-1.6108e-02,  7.3703e-03,  1.4649e-02],
  [-1.7572e-03, -5.9024e-03,  1.4436e-04],
  [-1.5447e-03, -4.0417e-04,  1.0898e-02]]],


[[[ 1.7815e-02,  2.3672e-02,  2.2100e-02],
```

```
          [ 1.1128e-03,  1.4265e-02,  1.8235e-02],
          [ 3.3604e-02,  4.0436e-03, -3.8509e-02]],

         [[ 7.0534e-04, -1.5921e-03,  2.2985e-03],
          [-7.1929e-03, -2.8636e-03,  1.1974e-02],
          [-5.5830e-03, -8.7426e-04, -7.8195e-03]],

         [[ 1.1015e-02,  1.3521e-02,  1.6727e-03],
          [-8.8975e-03,  1.9137e-02,  1.1484e-02],
          [-1.8926e-02,  7.9509e-03,  1.2649e-02]],

         ...,

         [[ 2.7888e-02,  2.2733e-02,  6.9183e-04],
          [ 4.9922e-03,  1.4068e-02, -5.1883e-04],
          [-3.2223e-04,  1.3758e-02, -2.5170e-02]],

         [[-1.9592e-02,  5.4293e-03,  3.1278e-03],
          [ 1.5075e-03, -5.0189e-03,  9.0607e-03],
          [-5.2190e-03, -9.6496e-03, -4.0984e-04]],

         [[-1.0701e-03, -7.3906e-04, -1.4306e-03],
          [-8.6153e-03,  1.7172e-02,  2.0672e-02],
          [ 2.6654e-02,  1.1704e-02,  2.6346e-02]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-1.3936, -0.8007, -0.5469, -0.6668, -0.6238, -0.4083, -0.9899,
        -1.0097, -0.5027, -0.6866, -0.3774, -1.0382, -0.6826, -0.9024,
        -0.6972, -0.3490, -1.1897, -0.4954, -1.2327, -0.8851, -0.5122,
        -0.9197, -0.4887, -1.1085, -0.7469, -0.5629, -0.8240, -0.3428,
        -0.2606, -0.9006, -0.8120,  0.0315, -0.8820, -0.5042, -0.7105,
        -0.6724, -0.5023, -1.3286, -0.1650, -1.2503, -0.6670, -0.9849,
        -0.8416, -0.7507, -0.6860, -0.8257, -1.0352, -0.5138, -0.5594,
        -0.3301,  0.0466, -0.7712, -0.8184, -0.8382, -0.3347, -0.6988,
```

```
        -0.1637, -0.7691, -0.9176,  1.4788, -0.5366, -0.8009, -0.4205,
        -0.5508,  0.4873, -0.8086, -0.7822, -0.9045, -1.0799, -0.7930,
        -0.7086, -0.8132, -0.7265, -0.7026, -0.2196, -1.0198, -1.1677,
        -0.6590, -0.9634, -0.2745, -1.0015, -1.4624, -0.6216, -0.5295,
        -0.4801, -0.7733, -0.8203, -0.7602, -0.7247,  0.0835, -0.9773,
        -0.7414, -0.9402, -0.7492, -0.6950, -1.0195, -0.7883, -0.0036,
        -0.9117, -0.4814, -0.5737, -0.9665, -0.8071,  0.0354, -0.5564,
        -0.0694, -1.4545,  0.1695, -1.0848, -0.4404, -0.8978, -0.9806,
        -0.5927, -0.5815, -1.2582, -0.6967, -0.4129, -0.6884, -0.3036,
        -0.8325, -0.8100, -0.5929, -0.8648, -1.0090, -0.5971, -0.7253,
         0.0896, -0.2279, -0.3885, -0.1903,  0.1675, -0.8336, -0.5447,
        -0.0914, -0.7888, -1.0995, -1.0485, -0.6222, -0.6597, -0.5688,
        -0.7024, -0.3296, -1.1295, -0.6932, -0.1316, -1.2851, -0.3438,
        -0.3139,  0.5483, -0.1996, -0.9184, -0.9037, -0.5802, -1.6198,
        -0.7761, -0.4512, -0.8087, -1.1522,  0.0771, -0.2439, -0.9217,
        -0.8017, -0.6379, -0.7342, -0.9955, -0.8137, -0.2537, -0.9826,
        -0.6882, -1.2424, -1.1518, -0.6993,  0.2098, -0.5517, -1.0921,
         0.8000, -1.1797, -0.8913, -0.6259, -1.0055, -0.2195, -0.5977,
        -1.4496, -1.0078, -0.5331, -0.8068, -0.8911, -0.4889, -0.7529,
        -0.9231, -0.5907, -0.7143]), Parameter containing:
tensor([[[[ 3.8634e-02]],

         [[-1.1381e-02]],

         [[-1.2402e-02]],

         ...,

         [[ 5.2069e-02]],

         [[-2.0312e-03]],

         [[-2.3333e-02]]],


        [[[-2.7409e-02]],

         [[-1.0828e-02]],

         [[-2.5711e-02]],

         ...,

         [[ 4.6801e-02]],

         [[ 5.0182e-04]],

         [[-1.5840e-02]]],
```

```
[[[-5.4778e-03]],
 [[-1.2390e-02]],
 [[-2.7284e-02]],
 ...,
 [[-6.5552e-02]],
 [[ 2.5620e-03]],
 [[-2.0224e-02]]],


...,


[[[-7.4930e-03]],
 [[-8.6029e-03]],
 [[ 1.0688e-03]],
 ...,
 [[ 3.7122e-02]],
 [[ 5.8293e-02]],
 [[-1.3808e-02]]],


[[[ 1.6836e-02]],
 [[-6.9334e-03]],
 [[ 2.7840e-04]],
 ...,
 [[-3.3795e-02]],
 [[ 5.9306e-02]],
 [[ 3.8112e-03]]],
```

```
        [[[-2.0422e-02]],

         [[ 3.5080e-03]],

         [[-7.3579e-03]],

         ...,

         [[-3.0192e-03]],

         [[-3.3634e-02]],

         [[ 2.0009e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter containing:
tensor([-0.9849, -0.7978, -0.5854, -0.8407, -1.0257, -1.2272, -0.9326,
        -0.8524, -0.9831, -0.8193, -0.8861, -1.1031, -1.0033, -0.9222,
        -1.6212, -0.7802, -0.9055, -0.9281, -0.9723, -0.9363, -0.9601,
        -0.9119, -0.9124, -0.9683, -0.9772, -0.8717, -1.1092, -0.9902,
        -1.0682, -0.9137, -0.9677, -0.8923, -1.0801, -0.9818, -0.9580,
        -0.9432, -0.8818, -1.0582, -0.9507, -0.9633, -0.9992, -0.8095,
```

```
          -0.9233, -0.9091,  0.4950, -0.6597, -0.8073, -0.8903, -0.8381,
          -1.1214, -1.0923, -0.9571, -0.8392, -0.7316, -1.1784, -0.9953,
          -0.8695, -0.9181, -0.9572, -0.9941, -1.0019, -0.9835, -0.8199,
          -0.9438, -0.8983, -0.8521, -0.9617, -1.0181, -0.9416, -0.9710,
          -1.0652, -1.1543, -0.8683, -0.9884, -0.8807, -1.1020, -0.9819,
          -0.9402, -0.8677, -0.9730, -0.9362, -0.8721, -1.1243, -1.0389,
          -0.8756, -1.0576, -0.8087, -1.0508, -0.9038, -1.1266, -0.9448,
          -0.7657, -0.9508, -0.9486, -0.7928, -1.0090, -0.7651, -0.8871,
          -0.9389, -0.9387, -1.0283, -0.8731, -0.8673, -0.8046, -0.7888,
          -1.0916, -1.0329, -0.9598, -0.8346, -0.9027, -0.9849, -1.0825,
          -0.7775, -0.9548, -1.1420, -1.0019, -0.9652, -0.7977, -0.9082,
          -0.8539, -1.0134, -0.8694, -0.9080, -0.8741, -1.0520, -0.7715,
          -0.7978, -0.7134, -0.8894, -0.9844, -0.8352, -1.1215, -1.0526,
          -1.0469, -0.9893, -1.0271, -0.9647, -1.0457, -0.8548, -0.7899,
          -0.9411, -1.0380, -0.9904, -0.8452, -0.8898, -0.9613, -0.9753,
          -0.8313, -0.9819, -0.9915, -1.0645, -1.0368, -1.0873, -0.8382,
          -1.0998, -0.9482, -0.8899, -0.7072, -0.8237, -0.8273, -0.9160,
          -0.7867, -0.8128, -0.8624, -0.9671, -1.0217, -1.0173, -0.9231,
          -0.8228, -0.7506, -0.9523, -0.9774, -1.0942, -1.0308, -0.9043,
          -0.8933, -0.7826, -0.8812, -1.0840, -0.5365, -1.0429, -0.9947,
          -1.0425, -0.8634, -0.8361, -0.8858, -0.8218, -0.8634, -1.0223,
          -0.8622, -1.0047, -0.9458, -1.0705, -1.0347, -1.0177, -0.8690,
          -0.8931, -1.0775, -1.0199, -0.9952, -0.8889, -0.9035, -0.8990,
          -0.8810, -0.9082, -0.8343, -0.9112, -0.9914, -0.9382, -0.9812,
          -0.9278, -0.7260, -1.2033, -0.8607, -0.9392, -1.0710, -0.7159,
          -0.8751, -1.0957, -1.1222, -0.9573, -0.9876, -0.7796, -0.9195,
          -1.0582, -1.0656, -0.9064, -0.7744, -0.8818, -0.7661, -1.0683,
          -0.8585, -0.8860, -0.9522, -1.0433, -0.4426, -0.9621, -0.9480,
          -1.0318, -1.1156, -1.0305, -0.9390, -0.9462, -0.9345, -0.8381,
          -0.9708, -1.0802, -0.8802, -0.9835, -0.7942, -0.8925, -1.0104,
          -0.9776, -0.6508, -0.9813, -0.9888, -1.0694, -1.1061, -0.9976,
          -0.9094, -0.8938, -0.9266, -0.8607, -0.8682, -0.9422, -1.1357,
          -0.9208, -0.8051, -1.1299, -0.8130, -0.8536, -0.7081, -0.9731,
          -0.8434, -1.1248, -0.9338, -0.8300, -1.1563, -0.7547, -1.0917,
          -1.1161, -0.9045, -1.1051, -0.8550, -0.7917, -0.9475, -1.2128,
          -1.0315, -0.8407, -0.7820, -0.9783, -0.9239, -0.8910, -1.0145,
          -0.8146, -1.0383, -0.9712, -0.8522,  0.0365, -1.0434, -0.6563,
          -0.9376, -0.8902, -0.9296, -1.1705, -0.9290, -0.9017, -0.8397,
          -1.1598, -0.8172, -1.0694, -1.0088, -0.8910, -0.8957, -0.8908,
          -0.9359, -0.7433, -0.8178, -0.6410, -0.8109]), Parameter containing:
tensor([[[[-1.6036e-02]],

         [[-2.0061e-02]],

         [[-3.0621e-02]],

         ...,
```

```
 [[ 2.7190e-02]],

 [[ 2.9724e-02]],

 [[ 2.2724e-02]]],


[[[-1.1529e-02]],

 [[-2.8641e-03]],

 [[-2.1307e-02]],

 ...,

 [[ 1.6981e-02]],

 [[ 1.8470e-03]],

 [[ 4.6011e-03]]],


[[[-2.4511e-02]],

 [[ 4.5695e-03]],

 [[ 1.1925e-02]],

 ...,

 [[-1.4918e-02]],

 [[-4.4636e-02]],

 [[ 9.5212e-03]]],


...,


[[[ 2.7104e-02]],

 [[-7.3900e-03]],

 [[ 3.1997e-02]],

 ...,
```

```
            [[ 3.1857e-02]],

            [[-6.0368e-03]],

            [[-1.8528e-02]]],


           [[[ 1.4190e-02]],

            [[ 2.8188e-03]],

            [[-1.9413e-03]],

            ...,

            [[-7.5365e-02]],

            [[ 1.2497e-02]],

            [[-7.3860e-03]]],


           [[[-2.2980e-02]],

            [[-2.5208e-02]],

            [[-2.4902e-02]],

            ...,

            [[-3.7351e-03]],

            [[ 7.1736e-03]],

            [[ 1.2715e-02]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
                  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.5828, -0.5656, -0.5616, -0.7573, -0.7003, -0.6991, -0.9016,
        -0.7918, -0.4422, -0.4924, -0.8506, -0.7920, -0.5083, -0.7417,
        -0.8364, -1.0093, -1.2642, -0.5194, -0.5630, -0.7010, -0.6362,
        -0.6029, -0.6873, -0.8836, -0.7709, -0.6807, -0.6900, -1.0221,
        -0.9253, -0.8837, -0.8937, -0.5771, -0.6315, -0.7523, -0.5480,
        -0.6054, -0.6968, -0.5942, -0.6408, -0.6998, -0.6662, -0.8640,
        -0.6721, -0.5566, -0.8402, -0.6373, -0.5626, -0.6603, -1.0112,
        -0.5666, -0.5955, -0.6371, -0.5583, -0.6916, -0.6803, -0.6836,
        -0.5795, -0.5399, -0.6927, -0.5678, -0.6643, -0.5066, -0.6284,
        -0.7071, -0.5446, -0.9476, -0.5485, -0.5558, -0.6788, -0.6351,
        -0.5930, -0.6480, -0.6751, -0.6793, -0.6017, -0.6665, -0.6666,
        -0.6443, -0.9897, -0.7749, -0.5768, -0.5536, -0.8895, -0.6683,
        -0.7133, -0.5541, -0.7114, -0.6156, -0.7238, -0.7575, -0.6667,
        -0.5375, -0.6977, -0.5354, -0.9348, -0.6716, -0.6345, -0.8149,
        -0.6284, -0.5545, -0.5385, -0.8970, -0.8020, -0.6388, -0.4274,
        -0.5568, -1.0039, -0.5933, -0.7582, -0.7744, -0.6142, -0.7164,
        -0.5014, -0.6135, -0.6023, -0.5419, -0.4801, -0.6265, -0.6657,
        -0.7319, -0.7478, -0.5338, -0.8466, -0.7694, -0.6676, -0.6281,
        -0.5346, -0.5729, -0.6959, -0.7665, -0.6929, -0.8174, -0.8876,
        -0.6272, -1.2548, -0.4935, -0.6012, -0.7409, -0.4823, -0.6514,
        -0.6051, -0.6319, -0.7161, -0.6353, -0.4373, -0.5797, -0.5822,
        -0.7760, -0.5318, -0.5692, -0.5225, -0.6078, -0.5006, -0.6300,
        -0.5725, -0.7200, -0.6468, -0.6643, -0.9152, -0.7409, -0.4513,
        -0.9736, -0.9846, -0.8006, -0.5432, -0.9633, -0.7041, -0.5405,
        -0.9902, -0.4219, -0.8206, -0.7490, -0.8599, -0.6233, -1.1333,
        -0.6223, -1.0424, -0.9782, -0.7047, -0.9277, -0.6952, -0.6840,
        -1.0071, -0.7794, -0.6868, -0.6338, -1.0432, -0.6001, -0.6064,
        -0.5790, -0.6170, -0.6236, -1.3029, -0.5707, -0.7298, -0.5606,
        -0.8080, -0.5692, -0.8304, -0.7395, -0.7334, -0.5296, -0.6941,
```

```
        -0.8020, -0.9091, -0.4738, -0.8522, -1.1487, -0.6783, -0.5144,
        -0.9828, -0.5233, -0.7621, -1.0644, -0.4834, -0.7773, -0.7877,
        -0.6496, -0.9636, -0.7210, -0.4600, -0.6475, -0.8903, -1.0398,
        -0.8262, -0.7402, -0.7696, -0.4574, -0.8507, -0.9400, -0.6518,
        -0.6394, -0.9573, -0.6854, -0.5548, -0.6805, -0.5672, -0.7053,
        -0.5777, -0.5910, -0.7210, -0.6422, -0.9453, -0.5347, -0.6393,
        -0.8320, -1.0529, -0.9344, -0.7127, -0.5375, -0.8982, -0.7720,
        -0.7658, -0.5878, -0.5643, -0.5461, -1.4069, -0.6481, -0.6695,
        -0.7327, -0.6444, -1.3557, -0.6915, -0.4735, -1.0232, -1.0522,
        -0.7054, -0.6958, -0.4070, -0.6647, -0.6595, -0.6886, -0.6421,
        -0.6143, -0.7655, -0.5847, -0.8800, -0.6265, -0.5151, -0.6701,
        -0.4577, -0.4939, -0.6441, -0.6821, -0.8877, -0.4976, -0.5940,
        -0.6055, -0.6497,  0.6527, -0.4880, -0.8177, -0.6315, -0.6078,
        -0.6982, -0.8673, -0.6900, -0.6698, -0.7000, -0.5777, -0.6447,
        -0.8120, -0.6477, -0.6541, -0.7452, -0.6086, -0.9626, -0.5949,
        -0.4409, -0.6534, -0.7737, -0.6724, -0.7273, -0.7390, -0.8010,
        -0.6970, -0.7701, -0.7370, -1.1568, -0.5920, -0.6939, -0.5933,
        -0.5696, -0.6276, -0.6549, -0.4932, -0.7340, -0.5202, -0.9235,
        -0.5681, -0.8422, -0.6148, -0.5115, -0.5889, -0.7580, -0.4755,
        -0.9083, -0.6643, -0.5734, -0.6131, -0.7424, -0.5389, -0.7858,
        -0.7770, -0.6096, -0.7501, -0.8193, -0.4421, -0.6426, -0.5602,
        -0.6982, -0.8379, -0.6224, -0.5565, -0.5676, -0.9740, -0.5616,
        -0.5582, -0.4371, -0.5989, -0.9926, -0.4896, -0.7954, -0.6070,
        -0.5942, -0.6491, -0.7122, -0.5969, -0.8245, -0.7576, -0.8136,
        -0.5538, -0.6502, -0.4136, -0.5560, -0.7616, -0.5608, -0.6750,
        -0.5712, -0.7342, -0.6964, -0.6042, -0.7462, -0.8052]), Parameter containing:
tensor([[[[-5.2179e-03, -5.7738e-03, -1.6137e-02]],

         [[ 3.8388e-02,  1.8610e-02,  3.5186e-02]],

         [[-1.3349e-02, -3.1612e-03, -1.2315e-02]],

         ...,

         [[ 1.1375e-02,  6.4053e-03,  1.9232e-02]],

         [[-1.1068e-02,  8.5698e-04, -4.3608e-03]],

         [[ 4.6590e-02,  3.2576e-02,  4.7747e-02]]],


        [[[-4.1878e-05, -5.1661e-03, -5.7354e-03]],

         [[-2.6376e-03, -1.3031e-02, -1.3146e-02]],

         [[-1.1427e-03, -6.8500e-03, -1.9290e-03]],

         ...,
```

```
 [[ 1.1458e-04, -1.0909e-02, -5.0142e-03]],
 [[-1.5092e-02, -1.7321e-02, -4.7377e-03]],
 [[ 1.1411e-02, -4.2050e-03,  1.6704e-03]]],


[[[ 1.3523e-02,  2.2608e-02,  1.0879e-02]],
 [[ 1.4168e-02,  3.9096e-02,  1.4072e-02]],
 [[ 2.4190e-02,  2.6205e-02,  2.1731e-02]],
 ...,
 [[-1.9268e-02, -2.3396e-02, -6.8460e-03]],
 [[ 1.8664e-02,  2.0477e-02,  2.4970e-02]],
 [[ 1.8933e-03,  9.6325e-03,  3.7538e-03]]],


...,

[[[-3.9458e-03, -5.2844e-03, -1.3826e-02]],
 [[-1.2288e-02, -1.2391e-02, -1.1719e-02]],
 [[-1.0548e-02, -3.9799e-03, -1.2136e-02]],
 ...,
 [[ 4.9949e-02,  2.1887e-03,  4.5058e-02]],
 [[ 5.0636e-02,  5.4897e-02,  5.5486e-02]],
 [[-1.2199e-02, -1.5758e-02, -1.6230e-02]]],


[[[-7.9086e-03, -4.8914e-03, -7.6249e-03]],
 [[ 1.6837e-02,  2.3613e-02,  2.1195e-02]],
 [[ 2.8196e-02,  1.9027e-02,  2.4615e-02]],
 ...,
```

```
        [[ 4.1400e-02,  2.1574e-02,  4.6929e-02]],

        [[-6.5806e-03,  3.3283e-03, -1.1295e-02]],

        [[-6.7029e-03,  1.8540e-03, -1.3327e-02]]],


       [[[ 5.6377e-02,  3.7524e-02,  5.3893e-02]],

        [[-7.6469e-03, -3.5088e-03, -6.1789e-03]],

        [[-7.4472e-03, -1.6766e-02, -1.6848e-02]],

        ...,

        [[-6.7332e-03, -1.8335e-02, -1.0342e-02]],

        [[-3.5469e-02, -1.1763e-02, -3.0505e-02]],

        [[ 5.0633e-02,  2.7697e-02,  5.5093e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.9936, -1.3094, -1.3860, -1.1366, -1.1304, -1.2538, -1.6466,
        -1.4643, -1.4228, -1.2673, -1.6878, -1.2245, -1.2954, -1.2756,
        -1.1428, -1.3165, -1.3656, -1.3117, -1.1371, -1.2252, -1.6353,
        -1.1760, -1.3554, -1.1430, -1.4546, -1.5506, -1.3133, -1.4022,
        -1.1758, -1.4843, -1.2601, -1.3526, -1.2211, -1.2924, -1.2042,
        -1.4990, -1.5760, -1.0512, -1.1293, -1.5958, -1.1432, -1.3200,
        -1.2595, -1.1298, -1.1835, -1.2378, -1.1077, -1.2597, -1.0659,
        -1.4029, -1.2666, -1.2728, -1.5010, -1.0964, -1.0685, -1.0970,
        -1.4447, -1.1332, -1.0854, -1.1008, -1.2867, -1.2877, -1.1600,
        -1.1938, -1.0667, -1.3302, -1.2177, -1.2633, -1.1754, -1.0995,
        -1.2396, -1.2459, -1.4106, -1.6538, -1.3131, -1.0782, -1.3135,
        -1.2386, -1.1294, -1.1633, -1.2460, -1.4178, -1.2179, -1.2219,
        -1.2201, -1.3532, -1.2939, -1.4502, -1.1326, -1.2248, -1.2497,
        -1.2280, -1.4371, -1.2448, -1.1154, -1.3929, -1.2275, -1.2974,
        -1.2070, -1.1744, -1.0983, -1.3997, -1.2362, -1.2517, -1.5373,
        -1.2410, -1.2452, -1.1909, -1.1302, -1.1846, -1.2990, -1.0974,
        -1.5869, -1.7985, -1.2427, -1.3828, -1.3890, -1.2929, -1.2267,
        -1.0109, -1.8184, -1.5369, -1.2344, -1.3968, -1.3814, -1.1612,
        -1.0616, -1.2932, -1.2362, -1.2998, -1.4958, -1.7803, -1.1482,
        -1.0017, -1.3254, -1.3418, -2.0488, -1.5005, -1.1787, -1.2372,
        -1.1077, -1.3416, -1.3161, -1.1418, -1.4012, -1.1030, -1.2465,
        -1.2805, -1.2135, -1.1433, -1.4632, -1.1647, -1.3155, -1.2205,
        -1.4372, -1.4147, -1.1407, -1.6158, -1.1457, -1.2117, -1.2870,
        -1.0968, -1.2347, -1.0869, -1.2840, -1.1943, -1.1500, -1.1316,
        -1.3915, -1.1213, -1.1539, -1.2116, -1.0727, -1.2743, -1.2924,
        -1.2403, -1.3216, -1.4295, -1.1301, -1.3817, -1.1632, -1.1819,
        -1.2060, -1.1775, -1.1230, -1.2142, -1.3679, -1.1987, -1.4419,
        -1.2875, -1.1685, -1.7152, -1.2169, -1.4483, -1.3015, -1.2275,
        -1.3341, -1.3660, -1.1837, -1.2514, -1.1954, -1.1430, -1.1001,
        -1.1209, -1.1556, -1.2357, -1.1179, -1.2738, -1.3514, -1.1900,
        -1.3021, -1.2353, -1.4160, -1.2530, -1.1675, -1.3183, -1.2912,
        -1.2469, -1.4184, -1.1775, -1.1980, -1.1442, -1.0803, -1.0813,
        -1.2179, -1.2661, -1.2590, -0.9801, -1.2648, -1.2650, -1.1478,
        -1.1413, -1.1796, -1.1980, -1.4293, -1.3573, -1.3516, -1.1646,
        -1.2000, -1.4065, -1.4052, -1.1376, -1.3810, -1.4701, -1.2236,
        -1.1095, -1.3214, -1.2318, -1.5421, -1.3337, -1.3188, -1.1129,
        -1.2907, -1.1414, -1.2550, -1.1608, -1.4124, -1.0429, -1.2141,
        -1.1727, -1.2308, -1.1247, -1.1739, -1.5573, -1.1622, -1.2624,
        -1.4830, -1.2748, -1.1780, -1.1643, -1.3181, -1.4482, -1.2812,
        -1.1407, -1.4959, -1.1744, -1.1867, -1.1517, -1.6149, -1.5035,
        -1.3728, -1.1398, -1.2091, -1.2805, -1.3056, -1.2433, -1.2480,
        -1.3359, -1.1078, -1.4105, -1.2256, -1.2581, -1.3767, -1.3109,
        -1.2813, -1.4184, -1.1903, -1.0957, -1.4966, -1.7531, -1.4522,
```

```
        -1.2864, -1.2286, -1.4059, -1.3465, -1.3198, -1.1984, -1.2409,
        -1.1036, -1.3237, -1.1012, -1.1469, -1.4485, -1.2674, -1.5249,
        -1.1485, -1.2113, -1.3066, -1.2244, -1.5167, -1.5317, -1.2717,
        -1.2229, -1.1573, -1.2710, -1.8129, -1.2967, -1.1656, -1.0904,
        -1.1374, -1.2183, -1.1376, -1.3936, -1.2415, -1.2365, -1.1767,
        -1.4520, -1.1089, -1.1197, -1.0413, -1.3066, -1.3687, -1.0025,
        -1.2045, -1.1761, -1.3251, -1.2227, -1.2642, -1.3075, -1.1761,
        -1.1747, -1.5555, -1.1771, -1.0965, -1.1690, -1.3820, -1.3180,
        -1.7032, -1.1569, -1.2624, -1.4356, -1.1818, -1.3862, -1.3784,
        -1.2434, -1.4447, -1.1719, -1.2297, -1.2810, -1.2834, -1.2713,
        -1.1435, -1.2401, -1.0605, -1.3113, -1.4770, -1.1926, -1.4907,
        -1.1885, -1.3766, -1.1199, -1.4383, -1.1375, -1.1943]), Parameter containing:
tensor([[[[ 1.4376e-02],
          [-1.0614e-02],
          [-1.9072e-03]],

         [[-2.4573e-02],
          [-2.0700e-02],
          [-5.5081e-03]],

         [[-6.5987e-03],
          [ 1.0924e-02],
          [ 1.3707e-02]],

         ...,

         [[ 2.3688e-02],
          [-1.5025e-02],
          [-2.6018e-02]],

         [[ 6.0650e-03],
          [ 4.3086e-03],
          [ 2.5001e-02]],

         [[-1.5213e-03],
          [-4.7393e-02],
          [-2.3251e-02]]],


        [[[-2.5434e-02],
          [-3.6828e-02],
          [-3.1126e-02]],

         [[-2.0382e-02],
          [-3.7144e-02],
          [-5.4891e-02]],

         [[ 1.3597e-03],
```

```
  [-2.2941e-02],
  [-2.4049e-02]],

 ...,

 [[-2.2082e-02],
  [ 1.9309e-02],
  [ 4.2429e-03]],

 [[ 1.3607e-02],
  [ 2.9334e-02],
  [ 2.8914e-02]],

 [[-1.6159e-02],
  [-4.3632e-03],
  [ 6.0440e-03]]],


[[[-2.4603e-02],
  [-3.0131e-02],
  [-1.3060e-02]],

 [[-2.2176e-02],
  [ 1.9854e-02],
  [ 2.1362e-02]],

 [[-2.2629e-02],
  [-3.0356e-02],
  [-2.4737e-02]],

 ...,

 [[ 2.3811e-02],
  [ 2.5392e-02],
  [ 4.0586e-02]],

 [[ 8.2289e-03],
  [ 1.5833e-02],
  [ 3.6326e-03]],

 [[ 3.1494e-02],
  [ 2.0691e-02],
  [-2.0370e-02]]],


...,
```

```
[[[-2.1171e-02],
  [-2.4304e-03],
  [-6.9178e-03]],

 [[ 1.2782e-02],
  [ 1.2298e-02],
  [ 2.5668e-02]],

 [[-1.8135e-02],
  [-1.6530e-02],
  [-5.7732e-03]],

 ...,

 [[-2.0376e-02],
  [-2.1998e-02],
  [-2.2262e-02]],

 [[-2.5783e-02],
  [-2.2110e-02],
  [-2.0308e-02]],

 [[-8.5227e-03],
  [ 1.4555e-03],
  [-1.1512e-02]]],


[[[-1.5259e-03],
  [-1.3935e-02],
  [-3.8123e-02]],

 [[-2.7583e-02],
  [-5.7139e-03],
  [-4.1347e-02]],

 [[-6.4707e-03],
  [-2.0697e-02],
  [-3.8853e-02]],

 ...,

 [[ 1.7889e-02],
  [ 2.9243e-02],
  [ 4.1177e-02]],

 [[-1.8499e-02],
  [-3.5784e-02],
  [-4.0887e-02]],
```

```
          [[ 8.6533e-03],
           [ 1.6434e-02],
           [ 3.8044e-03]]],


         [[[-7.2773e-03],
           [-3.0879e-02],
           [-2.3858e-02]],

          [[ 2.1131e-02],
           [ 1.7673e-02],
           [ 2.1349e-02]],

          [[-1.7400e-02],
           [-2.5220e-02],
           [-1.3569e-02]],

          ...,

          [[-7.2023e-03],
           [ 1.3082e-02],
           [ 2.4819e-02]],

          [[-1.9132e-02],
           [-3.4453e-02],
           [-1.0189e-02]],

          [[ 2.8986e-02],
           [ 7.6562e-03],
           [ 6.6666e-03]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-1.4484, -1.0636, -1.1961, -1.2263, -1.2618, -1.5399, -1.4714,
        -1.1339, -0.9856, -1.3033, -1.1145, -1.2714, -1.3510, -1.1399,
        -1.0701, -1.3004, -1.2524, -1.1290, -1.0543, -1.0392, -1.0867,
        -1.4679, -1.1044, -1.2308, -1.1676, -1.0948, -1.2797, -1.3598,
        -1.0756, -1.0226, -1.1059, -1.1802, -1.5257, -1.1870, -0.9823,
        -1.1832, -1.1983, -1.3028, -1.3310, -1.4042, -1.0925, -1.5290,
        -1.0380, -1.2911, -1.2697, -1.0959, -1.1928, -1.2164, -1.0253,
        -1.2178, -1.3275, -1.2304, -1.1074, -1.2996, -1.0547, -1.2007,
        -1.5432, -1.1050, -1.2440, -1.0519, -1.2281, -1.1310, -1.1311,
        -1.3883, -0.9816, -1.3832, -1.2022, -1.1313, -1.2029, -1.2192,
        -1.0497, -1.3162, -1.0427, -1.2962, -1.3127, -1.2062, -1.0801,
        -1.0672, -1.2799, -1.1559, -1.2008, -1.0908, -1.1207, -1.5570,
        -1.1865, -1.0338, -1.0013, -1.2063, -1.1385, -1.1955, -1.3527,
        -1.2352, -1.1410, -0.9557, -1.2383, -1.1372, -1.0050, -1.3271,
        -1.3144, -1.1645, -1.3148, -1.4519, -1.1853, -1.1500, -1.2473,
        -1.1821, -1.0893, -1.2116, -1.1706, -1.1373, -1.1152, -1.3219,
        -0.9807, -1.1403, -1.3197, -1.2939, -1.2007, -1.3572, -1.3263,
        -1.0493, -1.2328, -1.1997, -1.2810, -1.2577, -1.1678, -1.2085,
        -1.1080, -1.3854, -1.1224, -1.1999, -1.1577, -1.0640, -1.0543,
        -1.0964, -1.1804, -0.9402, -1.1656, -1.1459, -0.9976, -1.2749,
        -1.1548, -1.2255, -1.6290, -1.2392, -1.1836, -1.1967, -1.1301,
        -1.4413, -1.5087, -0.9897, -1.1132, -1.1365, -1.0844, -1.1006,
        -0.9373, -1.5735, -1.3251, -1.2515, -1.1806, -1.0385, -1.0631,
        -1.3054, -1.2921, -1.1984, -1.1207, -0.9660, -1.1050, -1.1052,
        -1.1372, -1.0903, -1.0261, -1.1981, -1.2406, -0.9732, -1.2314,
        -1.3902, -1.0457, -1.2403, -1.1686, -1.2964, -1.0768, -1.4614,
        -1.1353, -1.3166, -1.2347, -0.9567, -1.3171, -1.2698, -0.9826,
        -1.2443, -1.1984, -1.0749, -1.0650, -1.2325, -1.2770, -1.0728,
        -1.1016, -1.0757, -1.0611, -1.1291, -1.0741, -1.2310, -1.6939,
        -1.0392, -1.1265, -1.5513, -1.2491, -1.5751, -1.0206, -1.0349,
        -1.1655, -1.1457, -1.0736, -0.9571, -1.2006, -1.0597, -1.0760,
        -1.3597, -1.0855, -1.0994, -0.9020, -1.5189, -1.2770, -1.2864,
        -1.1475, -1.1102, -1.3145, -0.9657, -1.0965, -1.1111, -1.2360,
```

```
        -1.2870, -1.1459, -1.2753, -1.1435, -1.0413, -1.1311, -1.2480,
        -1.1971, -1.1641, -1.1384, -1.1292, -1.2425, -1.2103, -1.1128,
        -1.2617, -1.2086, -1.2726, -1.1973, -1.0035, -1.4080, -1.0467,
        -1.1141, -1.1137, -1.2686, -1.0995, -1.2618, -1.2015, -1.2600,
        -0.9958, -1.3644, -1.1290, -1.1186, -1.1110, -0.9335, -1.2778,
        -1.1876, -1.0934, -1.1206, -1.2358, -1.0583, -1.2001, -1.2032,
        -1.0230, -1.2076, -1.1706, -1.3075, -1.1906, -1.0839, -1.3941,
        -1.2180, -1.1051, -1.3742, -1.1479, -1.1519, -1.2468, -1.1189,
        -1.3750, -1.0566, -0.9701, -1.2719, -1.7142, -1.0962, -1.0959,
        -1.0047, -1.4002, -1.1139, -1.1195, -1.3921, -0.9833, -1.1290,
        -1.0445, -1.1139, -1.1782, -1.2310, -1.2165, -1.1197, -1.5590,
        -1.2083, -1.3229, -1.3945, -1.2693, -1.0707, -1.2373, -1.2259,
        -1.0013, -1.4114, -1.3992, -1.0650, -1.1883, -1.1436, -1.0987,
        -1.4171, -1.0817, -1.1945, -1.2181, -1.1411, -1.1815, -1.1870,
        -1.1324, -1.1012, -1.1188, -1.1999, -1.2406, -0.9079, -1.1668,
        -1.2519, -1.0897, -1.2374, -1.3827, -1.1964, -1.3009, -1.3517,
        -1.2163, -1.0873, -1.5446, -1.3482, -1.2237, -1.1278, -1.1847,
        -1.2256, -0.9811, -1.2416, -1.2372, -1.1320, -1.4182, -1.3931,
        -1.2967, -1.0546, -1.1843, -1.1378, -1.1005, -1.1155, -1.3138,
        -1.3939, -0.9740, -1.2205, -1.1895, -1.0821, -0.9890, -1.1151,
        -1.1819, -1.1861, -1.0280, -1.0374, -1.3023, -1.1261, -1.1290,
        -1.2410, -1.1527, -1.3219, -1.0598, -1.0634, -1.2521]), Parameter containing:
tensor([[[[ 3.6281e-02]],

         [[-3.3891e-02]],

         [[-8.6201e-03]],

         ...,

         [[-1.6436e-02]],

         [[ 4.9372e-03]],

         [[ 9.9728e-03]]],


        [[[-1.1750e-02]],

         [[ 7.7887e-03]],

         [[-1.3936e-02]],

         ...,

         [[-3.3720e-02]],

         [[-5.9442e-04]],
```

```
 [[-3.1935e-02]]],


[[[-2.2434e-02]],

 [[ 4.3818e-03]],

 [[-4.0787e-03]],

 ...,

 [[-1.3647e-03]],

 [[-3.4030e-02]],

 [[ 1.9443e-02]]],


...,


[[[ 2.4220e-02]],

 [[-1.8065e-03]],

 [[-3.2675e-02]],

 ...,

 [[ 1.8432e-02]],

 [[ 2.7246e-02]],

 [[ 2.5132e-02]]],


[[[-2.6181e-03]],

 [[ 7.6961e-03]],

 [[ 1.2348e-02]],

 ...,

 [[ 2.0674e-02]],

 [[-2.2537e-02]],
```

```
         [[ 8.7033e-03]]],


        [[[-1.1433e-02]],

         [[-2.8033e-02]],

         [[ 1.4127e-02]],

         ...,

         [[-9.4705e-03]],

         [[ 2.4326e-02]],

         [[-1.5011e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.]), Parameter containing:
tensor([-0.7289, -0.7436, -0.9426, -0.8101, -0.4269, -0.5374, -1.1898,
        -0.7201, -1.1745, -0.8873, -0.6055, -0.6978, -1.0480, -0.6627,
        -0.7412, -0.6021, -0.5621, -0.4759, -0.6358, -0.6964, -0.7440,
        -0.7011, -0.6958, -0.7207, -0.6026, -0.7594, -0.4453, -1.0166,
        -0.6677, -0.5461, -0.7467, -0.6857, -0.7635, -0.7505, -0.6709,
        -0.8238, -0.6819, -0.7789, -0.7299, -0.6777, -0.6246, -0.5772,
        -0.8124, -0.8554, -0.8991, -0.7538, -1.0450, -0.9838, -0.7379,
        -0.7348, -0.7051, -0.5201, -0.5862, -0.8058, -0.6881, -0.8115,
        -0.8198, -0.5384, -0.8141, -0.5593, -0.8500, -0.6763, -0.6430,
        -0.6317, -0.1728, -0.6960, -0.9555, -0.7558, -0.6531, -0.4889,
        -0.7130, -0.1730, -0.8849, -0.6723, -0.6971, -0.8225, -0.6361,
        -1.0507, -0.6552, -0.7228, -0.5015, -0.5830, -0.7675, -0.7343,
        -0.6329, -0.5746, -0.5357, -0.7380, -0.8852, -0.8129, -0.8285,
        -0.7401,  0.3565, -0.8788, -0.8414, -0.8025, -0.8908, -0.8031,
        -0.5126, -0.6607, -0.6155, -0.8715, -0.9338, -0.5178, -0.8067,
        -0.6493, -0.8853, -0.7350, -0.6393, -0.7348, -0.8958, -0.7846,
        -0.6337, -0.6068, -0.3579, -0.6541, -0.8487, -0.7717, -0.7685,
        -0.4675, -0.6497, -0.8651, -0.8613, -0.8506, -0.7706, -0.6533,
        -0.4099, -1.8767, -0.6822, -0.7188, -0.7843, -0.5380, -0.5275,
        -0.5139, -0.7309, -1.2066, -0.9057, -0.4562, -0.9088, -0.9324,
        -0.9579, -0.9734, -0.4982, -0.8429, -0.5826, -0.8546, -0.7768,
        -0.5865, -0.6702, -0.7718, -0.7764, -0.6287, -0.9728, -0.6011,
        -0.7727, -0.5797, -0.7823, -0.7616, -0.5138, -0.6996, -1.0487,
        -0.6180, -0.8454, -0.6915, -0.6908, -0.6905, -0.5916, -0.8037,
        -0.6123, -0.5799, -0.8049, -0.9528, -0.8715, -0.6643, -0.5565,
        -0.7315, -0.7676, -0.9304, -0.8326, -0.7549, -0.5739, -0.7071,
        -0.6153, -0.8597, -0.0059,  0.0160, -0.7793, -0.7477, -0.4762,
        -0.8297, -0.8984, -1.2509, -0.9539, -0.8009, -0.9316, -0.8082,
        -0.4660, -0.8675, -0.6086, -0.6516, -0.7453, -0.6595, -0.6387,
        -0.6703, -0.6719, -0.6191, -0.7822, -0.5038, -0.6451, -0.9205,
        -1.0713, -0.9250, -0.6990, -0.7021, -1.5522, -1.2785, -0.8302,
        -0.8352, -0.9560, -0.8172, -0.6102, -0.6246, -0.9564, -0.6923,
        -0.8237, -0.7566, -0.7149, -0.7992, -0.6430, -0.6223, -0.6518,
        -0.6656, -0.8154, -0.5589, -0.6126, -0.6901, -0.5502, -0.7784,
        -0.5776, -0.8206, -0.6060, -0.8270, -0.8193, -0.8771, -0.5523,
        -0.9080, -1.0264, -0.6787, -0.8960, -0.5955, -0.6725, -0.6500,
        -0.6127, -0.5808, -0.7930, -1.0192, -0.6817, -0.8096, -0.6643,
        -0.6713, -0.6249, -0.6487, -0.7663, -0.6067, -0.7068, -0.6029,
        -0.8756, -0.7120, -0.8302, -1.1951, -0.7795, -0.6916, -0.9656,
        -0.6906, -0.7220, -0.9298, -0.7338, -0.6558, -0.5302, -0.7141,
        -0.8232, -0.5098, -0.8192, -0.6674, -0.8424, -0.6679, -0.7525,
```

```
          -0.8553, -0.8102, -0.7407, -0.5803, -0.6604, -0.9428, -0.8109,
          -0.6864, -0.7998, -0.7929,  0.2188, -0.9057, -0.8498, -1.3306,
          -0.7645, -0.7440, -0.7148, -0.5802, -0.8059, -0.8649, -0.6441,
          -0.6789, -0.6593, -0.7836, -0.5696, -0.8080, -0.5598, -0.6649,
          -0.6319, -0.7612, -0.7230, -0.7894, -0.6958, -0.9518, -0.7581,
          -0.6149, -0.8355, -0.8845, -0.6041, -0.7597, -0.6458, -0.5954,
          -0.5784, -0.7635, -0.6804, -0.7458, -0.7007, -0.5646, -0.7291,
          -0.7713, -0.8633, -0.7641, -0.9028, -0.8385, -0.6974, -0.8385,
          -0.2984, -0.7047, -0.6182, -0.7389, -0.8093, -0.7979, -0.7068,
          -0.7221, -0.8929, -0.5594, -0.8580, -0.5352, -0.5919, -0.6519,
          -0.6837, -0.7801, -0.7070, -0.7831, -0.9489, -0.9635, -0.6814,
          -0.7325, -0.7595, -0.6518,  0.1028, -0.6963, -0.7021, -0.6179,
          -0.6159, -0.7307, -1.1423, -0.9086, -0.7574, -0.7327, -0.7871,
          -0.5712, -0.5925, -0.8959, -0.6247, -0.5692, -0.8453, -0.7220,
          -0.6089, -0.7616, -0.7412, -0.6405, -0.8493, -0.7746, -0.6417,
          -0.8882, -0.5360, -0.5703, -0.6849, -0.9963, -0.8968, -0.8409,
          -0.7815, -0.7785, -0.7785, -0.8126, -0.8787, -0.8709, -0.7546,
          -0.5703, -0.7252, -0.5290, -0.5977, -0.9642, -0.6351, -0.9099,
          -0.9640, -0.7297, -0.4508, -0.7584, -0.8359, -0.5925, -0.8823,
          -0.5532, -0.8817, -0.6657, -0.5263, -0.6407, -0.8079, -0.8519,
          -0.8153, -0.9700, -0.6481, -0.5591, -0.4592, -0.7414, -0.8494,
          -0.5364, -0.7449, -0.6741, -0.7207, -0.4803, -0.9307, -0.5926,
          -0.6203, -0.5252, -0.8064, -1.0972, -0.6752, -0.9462, -0.8671]), Parameter cc
tensor([[[[-1.6842e-02, -2.0598e-02, -1.4276e-02],
          [-1.0271e-02, -4.2601e-03,  1.1210e-05],
          [-2.3601e-03, -1.6672e-02,  2.3177e-03]],

         [[-1.3276e-02, -1.7367e-02, -1.7671e-02],
          [-1.8644e-02, -1.5122e-02, -1.9907e-02],
          [ 2.5760e-03,  5.8702e-03,  4.9625e-03]],

         [[-1.9039e-02, -2.3542e-02, -1.6056e-02],
          [-4.1921e-03, -7.8027e-03, -3.3006e-03],
          [ 2.0770e-03,  2.2319e-04,  8.3657e-04]],

         ...,

         [[-7.4215e-03, -2.0545e-03, -1.1564e-02],
          [ 9.2881e-03,  1.6700e-02,  8.3780e-03],
          [ 3.3920e-03, -9.2834e-04, -4.5887e-04]],

         [[-1.6098e-02, -2.8438e-02, -2.5930e-02],
          [-3.9759e-03, -3.6386e-03, -3.8171e-03],
          [ 4.3336e-03,  6.7967e-04,  1.1529e-02]],

         [[-2.3248e-03, -1.4166e-02, -1.8058e-02],
          [ 1.0351e-03,  3.3221e-03, -3.1710e-03],
          [ 1.0952e-02,  1.9481e-02,  1.8481e-02]]],
```

```
[[[-1.3514e-02, -1.1935e-02, -1.5257e-02],
  [-1.2897e-02, -7.5891e-03, -1.3457e-02],
  [ 6.9504e-03,  4.9677e-03,  1.3058e-02]],

 [[ 8.9302e-03,  2.3822e-03,  4.9573e-03],
  [-5.2736e-03, -1.1190e-02, -1.1000e-03],
  [-4.2038e-03, -9.7826e-03, -2.3187e-03]],

 [[-1.5214e-02, -7.8989e-03, -2.0982e-03],
  [-2.5875e-02, -7.3971e-03, -2.1355e-02],
  [-2.3546e-02, -1.2924e-02, -2.7403e-02]],

 ...,

 [[ 1.2762e-02,  8.8231e-03,  1.7850e-02],
  [-4.8772e-03, -1.3807e-03, -1.0239e-02],
  [-9.4362e-04, -1.2310e-02, -7.1959e-03]],

 [[ 4.1539e-03,  6.6424e-03,  5.4072e-03],
  [-1.3305e-03,  3.8714e-03, -3.3220e-03],
  [-5.5890e-03, -1.2763e-02, -3.8614e-03]],

 [[-1.7634e-02, -1.1848e-02, -4.4337e-03],
  [ 2.9421e-03,  1.4789e-02,  1.4411e-02],
  [ 2.3922e-02,  2.7559e-02,  2.0104e-02]]],


[[[-2.3808e-02, -2.6596e-02, -2.7560e-02],
  [-7.4544e-03, -3.1470e-03, -8.5883e-03],
  [-8.5287e-03,  9.9268e-03, -1.1172e-03]],

 [[-2.6734e-04,  1.4564e-02,  1.2071e-02],
  [-2.7874e-04,  1.9151e-03,  1.1798e-02],
  [ 6.2957e-03,  1.2427e-02,  1.4782e-02]],

 [[-6.2003e-03,  3.4780e-03, -5.4696e-03],
  [ 1.7483e-03,  1.1673e-02, -9.7089e-04],
  [-5.7651e-03,  8.4405e-03, -8.8213e-03]],

 ...,

 [[ 1.7338e-03,  1.0280e-02, -2.0273e-04],
  [ 1.5215e-02,  2.2370e-02,  5.7679e-03],
  [-2.2840e-02,  2.4333e-03, -1.1762e-02]],

 [[ 1.3086e-02,  1.5013e-02,  1.4482e-02],
```

```
   [ 1.6500e-02, -6.2759e-03,  1.1913e-02],
   [ 1.6804e-02,  1.1445e-02,  1.5374e-02]],

  [[-2.3272e-03, -2.3383e-02, -8.3432e-03],
   [ 4.7369e-03, -1.7125e-02, -2.4316e-04],
   [ 7.9313e-03, -5.5280e-03,  6.1683e-03]]],


 ...,


 [[[ 3.8137e-03,  2.9065e-03,  3.4190e-03],
   [ 1.8116e-02,  1.8500e-02,  1.7258e-02],
   [ 1.9281e-02,  2.6543e-02,  2.2384e-02]],

  [[-2.7628e-02, -1.7255e-02, -2.0568e-02],
   [-4.2052e-03, -1.0963e-02, -6.2025e-03],
   [-8.3327e-03, -7.2367e-03, -5.5557e-03]],

  [[ 2.9410e-03,  2.7539e-03,  7.5567e-04],
   [ 1.3926e-02,  1.3429e-02,  1.6151e-02],
   [ 1.3563e-02,  1.5468e-02,  8.2742e-05]],

  ...,

  [[-5.8591e-03, -1.6226e-02, -1.4216e-02],
   [-1.6068e-02, -1.8017e-02, -1.7485e-02],
   [-5.9158e-03, -1.2450e-02, -9.3119e-03]],

  [[ 1.0208e-02,  8.3072e-03,  1.0482e-02],
   [-1.3361e-03, -1.0791e-02, -1.6688e-03],
   [-6.8615e-03, -9.4051e-03, -2.2868e-03]],

  [[ 3.1561e-02,  4.0782e-02,  3.0635e-02],
   [ 2.8463e-02,  3.2792e-02,  2.6994e-02],
   [ 1.4220e-02,  1.5681e-02,  5.0713e-03]]],


 [[[-3.3335e-03, -1.4957e-02, -7.4181e-03],
   [ 1.2418e-03, -4.0884e-03, -3.0930e-03],
   [ 4.5214e-03,  9.3549e-03,  6.2267e-03]],

  [[-1.9159e-02, -1.8707e-03, -1.6536e-02],
   [-1.2396e-02, -3.7607e-03, -1.0645e-02],
   [-1.4080e-02, -4.0730e-03, -6.5975e-03]],

  [[ 3.9216e-03,  1.0261e-02, -9.5820e-03],
   [-9.2228e-04,  2.6833e-03, -9.6669e-03],
```

```
          [-1.2602e-03,  8.0214e-03, -3.9008e-03]],

         ...,

         [[-1.0237e-02,  2.6940e-03,  3.3629e-03],
          [ 3.7897e-03,  9.8050e-03,  8.7898e-03],
          [-1.0030e-02, -4.5401e-03, -5.0849e-03]],

         [[ 1.1201e-02,  8.6087e-03,  7.0782e-03],
          [ 1.2091e-02,  8.4379e-03,  1.0173e-02],
          [ 1.6496e-02,  1.4824e-02,  1.6357e-02]],

         [[ 1.3709e-03,  2.9717e-03,  1.0069e-02],
          [-1.5573e-03, -8.2055e-03,  8.9460e-04],
          [-6.5944e-03, -1.7473e-02, -1.2148e-02]]],


        [[[ 2.5753e-03,  1.4569e-02,  4.6645e-03],
          [-5.1740e-03,  5.3076e-03, -6.1024e-03],
          [-7.1079e-03,  6.9591e-03, -1.1201e-02]],

         [[-4.4523e-03, -1.9660e-03,  2.5528e-03],
          [-5.5293e-03, -5.7334e-03,  4.8866e-03],
          [ 3.0081e-03, -5.6081e-03,  7.1931e-03]],

         [[ 7.6981e-03,  7.3136e-03,  1.1914e-02],
          [ 5.5834e-04,  2.0922e-03,  4.6882e-03],
          [-1.3255e-03,  3.6234e-03, -9.0617e-04]],

         ...,

         [[ 2.4073e-02,  1.3982e-03,  2.9523e-02],
          [ 2.8556e-02,  1.0556e-03,  2.3742e-02],
          [ 9.5749e-03, -1.5119e-02,  8.2984e-04]],

         [[ 1.3951e-02,  1.7825e-02,  1.0829e-02],
          [ 2.0035e-02,  3.0684e-02,  1.5525e-02],
          [-1.3623e-02, -3.6152e-03, -6.9655e-03]],

         [[-2.1046e-03, -6.0506e-04,  1.9632e-04],
          [ 1.8261e-03, -2.6193e-03, -1.6851e-04],
          [ 6.6605e-03, -2.7160e-03, -3.0763e-03]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.6001, -0.5206, -0.4369, -0.5746, -0.6743, -0.4760, -0.5254,
        -0.5069, -0.5150, -0.5927, -0.3728, -0.5359, -0.7406, -0.6074,
        -0.5547, -0.5533, -0.6429, -0.5384, -0.4530, -0.6169, -0.5226,
        -0.5405, -0.8200, -0.5199, -0.5710, -0.4472, -0.4758,  0.2821,
        -0.6543, -0.3756, -0.4740, -0.6038, -0.6622, -0.7704, -0.4778,
        -0.7547, -0.4260, -0.4309, -1.0574, -0.5875, -0.0459, -0.3680,
        -0.6745, -0.5630, -0.5240, -0.6012, -0.5970, -0.4795, -0.5182,
        -0.7644, -0.6476, -0.6205, -0.7234, -0.5438, -0.6516, -0.7475,
        -0.5897, -0.6255, -0.4756, -0.6504, -0.6241, -0.4801, -0.5919,
        -0.3620, -0.6752, -0.7267, -0.7532, -0.8159, -0.5379, -0.4556,
        -0.5513, -0.6667, -0.4108, -0.7163, -0.5539, -0.7339, -0.6195,
        -0.8868, -0.8781, -0.4520, -0.4888, -0.5490, -0.4253, -0.3771,
        -1.0587, -0.6921, -0.6199, -0.7087, -0.4993, -0.7620, -0.6007,
        -0.8095, -0.7127,  0.8581, -0.3819, -0.4376, -0.4576, -0.6933,
        -0.6241, -0.5690, -0.4927, -0.5022, -0.5085, -0.6311, -0.6091,
        -0.5940, -0.5316, -0.3713, -0.3225, -0.6250, -0.5891, -0.4813,
        -0.4458, -0.6768, -0.4694, -0.7022, -0.8861, -0.4603, -0.6275,
        -0.4276, -0.6600, -1.3299, -0.5605, -0.5874, -0.4007, -0.5027,
        -0.9096, -0.5743, -0.9334, -0.6903, -0.6616, -0.5724, -0.7156,
        -0.5046, -0.5466, -0.4928,  0.0847, -0.4051, -0.5163, -0.5740,
        -0.5576, -0.7650, -0.5671, -0.5783, -0.8043, -0.6228, -0.5903,
        -0.5528, -0.5020,  0.0102, -0.3600, -0.7569, -0.6912, -0.5092,
```

```
          -0.8780, -0.5310, -0.7062, -0.5945, -0.6520, -0.7231, -0.5778,
          -0.6434, -0.3757, -0.4580, -0.3944, -0.3618, -0.6168, -0.7730,
          -0.5217, -0.4723, -0.5607, -0.5854, -0.4589, -0.4644, -0.8175,
          -0.7404, -0.6211, -0.5196, -0.4943, -0.5333, -0.6333, -0.7870,
          -0.4319, -0.4181, -0.0896, -0.4976, -0.8866, -0.5164, -0.2816,
          -0.4708, -0.4762, -0.4920, -0.5880, -0.3896, -0.4078, -1.0062,
          -0.4489, -0.5059, -0.4591, -1.0910, -0.4842, -0.6258, -0.5529,
          -0.3561, -0.6263, -0.7487, -0.6574, -1.0658, -0.4072, -0.1111,
          -0.4001, -0.7596, -0.6457, -0.3378, -0.5879, -0.4192, -0.4009,
          -0.3539, -0.5000, -0.5169, -0.7876, -0.5733, -0.5855, -0.6332,
          -0.4338, -0.4955, -0.6602, -0.3191, -0.5092, -0.4203, -0.7349,
          -0.3774, -0.6859, -0.4712, -0.5626, -0.3252, -0.5156, -0.5901,
          -0.6239, -0.5787, -0.5526, -0.5467, -0.6838, -0.5045, -0.7069,
          -0.5985, -0.6266, -0.5393, -0.5399, -0.5691, -0.5992, -0.7296,
          -0.7420, -0.7045, -0.5616, -0.5297, -0.6941, -0.7764, -0.7101,
          -0.6416, -0.4652, -0.6830, -0.4880, -0.7115, -0.6478, -0.4814,
          -0.3753, -0.7993, -0.6217, -0.6302, -0.6508, -0.7611, -0.5096,
          -0.5011, -0.5222, -0.6975, -0.5299, -0.5674, -0.4489, -0.6429,
          -0.5587, -0.5334, -0.4686, -0.6158, -0.4464, -0.5098, -0.6810,
          -0.4755, -0.6054, -0.5092, -0.5512, -0.6793,  0.2587, -0.7314,
          -0.6757, -0.5939, -0.7092, -0.9162, -0.7168, -0.5202, -0.5147,
          -0.6513,  0.8764, -0.5541, -0.5418, -0.5951, -0.5644, -0.5112,
          -0.7271, -0.6492, -0.6791, -0.5249, -0.4774, -0.6875, -0.6254,
          -0.7037, -0.5318, -0.4668, -0.5232, -0.4439, -0.4777, -0.5759,
          -0.6983, -0.6110, -0.5206, -0.5929, -0.5163, -0.5353, -0.4664,
          -0.5284, -0.6271, -0.5947, -0.2911, -0.7214, -0.6441, -0.5889,
          -0.5912, -0.7229, -0.5120, -0.5466, -0.6012, -0.4134, -0.5649,
          -0.7321, -0.7438, -0.6231, -0.6454, -0.5772, -0.6173, -0.7418,
          -0.7013, -0.8119, -0.5502, -0.5031, -0.5560, -0.5678, -0.5670,
          -0.4883, -0.5525, -0.7577, -0.6038, -0.3718, -0.3536, -0.5352,
          -0.7573, -0.3823, -0.5165, -0.6100, -0.4470, -0.6325, -0.8838,
          -0.4776, -0.4423, -0.5265, -0.3796, -0.5510, -0.5228, -0.6164,
          -0.6303, -0.5992, -0.4628, -0.5518, -0.7748, -0.7340]), Parameter containing:
tensor([[[[ 1.4293e-02,  1.1453e-02,  1.2483e-02]],

         [[ 1.0529e-02,  1.5616e-02,  2.0105e-02]],

         [[-3.2694e-02, -3.4445e-02, -3.7687e-02]],

         ...,

         [[ 1.7592e-03,  4.4924e-03,  7.8485e-04]],

         [[ 3.3042e-02,  2.3407e-02,  2.8435e-02]],

         [[ 3.0664e-03,  1.2282e-02,  1.0348e-02]]],
```

```
[[[ 6.4276e-03,  2.0842e-03,  4.2527e-03]],
 [[-1.6495e-02, -1.3379e-02, -1.4685e-02]],
 [[ 1.0081e-03,  6.1569e-03,  9.1649e-03]],
 ...,
 [[ 1.2101e-02,  2.6299e-03,  9.3606e-03]],
 [[-3.3539e-02, -3.0218e-02, -3.0950e-02]],
 [[ 6.9438e-04,  3.8232e-03, -2.2613e-03]]],


[[[-1.2725e-02, -8.6301e-03, -8.5087e-03]],
 [[ 1.3084e-02,  1.7037e-02,  1.3669e-02]],
 [[ 9.5974e-03,  1.9639e-02,  1.2763e-02]],
 ...,
 [[ 1.9531e-02,  2.4180e-03,  2.5173e-02]],
 [[-1.0897e-02, -1.3017e-02, -2.4772e-03]],
 [[-1.9763e-02, -2.2533e-02, -2.0880e-02]]],


...,


[[[ 5.2028e-04,  1.4682e-03,  1.1764e-02]],
 [[ 4.2781e-03,  7.1187e-03,  6.0565e-03]],
 [[-1.8378e-03,  5.6994e-03, -1.4455e-03]],
 ...,
 [[ 1.0636e-02, -2.4852e-03,  3.4463e-03]],
 [[-2.7807e-03,  5.0598e-03,  2.8364e-03]],
 [[-1.5084e-02, -7.0277e-03, -1.3594e-02]]],
```

```
         [[[-3.1782e-02, -3.2502e-02, -2.8461e-02]],

          [[ 1.7778e-02,  1.0040e-02,  1.8219e-02]],

          [[-1.8375e-02, -2.6408e-02, -2.1856e-02]],

          ...,

          [[-1.5279e-02, -1.5933e-02, -2.0250e-02]],

          [[ 1.3824e-04,  5.5676e-03, -6.2705e-04]],

          [[-2.0107e-02, -1.1297e-02, -1.6884e-02]]],


         [[[-1.2882e-02, -1.3436e-02, -1.9360e-02]],

          [[ 3.4361e-02,  4.2594e-02,  3.3729e-02]],

          [[-1.6079e-02, -1.3808e-02, -2.1692e-02]],

          ...,

          [[-1.3468e-02, -1.4289e-02, -2.2654e-02]],

          [[ 6.0345e-03,  5.4468e-03,  1.0921e-02]],

          [[-8.3261e-03, -1.7220e-02, -1.5621e-02]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-1.5367, -1.3317, -1.2959, -1.1244, -1.4916, -1.2747, -1.1335,
        -1.4808, -1.2275, -1.2689, -1.2877, -1.3093, -1.2734, -1.0496,
        -1.1658, -1.2763, -1.3407, -1.3235, -1.0612, -1.3279, -1.3028,
        -1.3304, -0.7784, -1.4885, -1.3614, -1.3993, -1.3568, -1.2747,
        -1.6632, -1.0852, -1.3097, -1.1552, -1.0327, -1.1773, -1.3515,
        -1.3330, -1.0346, -1.2044, -1.0098, -1.5757, -1.0253, -1.6359,
        -1.2713, -1.4836, -1.2582, -1.3078, -1.0830, -1.4526, -1.0848,
        -1.2362, -1.1297, -1.2019, -1.2719, -1.1974, -1.1880, -1.5706,
        -1.3142, -1.3036, -1.3183, -1.0939, -0.2450, -1.2159, -0.9519,
        -1.3529, -1.3687, -1.1258, -1.0392, -1.1717, -1.1358, -1.4286,
        -1.2610, -1.3096, -1.5306, -1.1399, -1.3505, -1.0008, -1.2623,
        -1.3475, -1.2358, -1.0787, -1.1296, -1.2481, -1.2404, -1.1335,
        -1.4933, -1.2810, -1.2262, -1.4682, -1.2553, -1.2613, -1.3558,
        -1.6220, -1.2677, -1.3425, -1.2118, -1.3410, -1.1071, -1.4099,
        -1.3027, -1.7532, -1.4894, -1.0199, -1.3867, -1.2247, -1.3197,
        -1.4358, -1.4501, -1.3089, -1.1542, -1.4121, -1.2978, -1.1225,
        -1.2642, -1.3097, -1.3739, -1.2397, -1.2170, -1.1789, -1.2025,
        -0.9935, -1.1155, -1.1934, -1.2892, -1.4732, -1.2692, -1.2959,
        -0.9764,  2.1231, -1.4200, -1.1260, -1.3440, -1.0427, -1.4285,
        -1.3562, -1.6653, -1.3606, -1.1579, -1.1131, -1.2988, -1.5410,
        -1.1828, -1.1886, -1.5684, -1.4623, -1.4771, -1.0964, -1.3826,
        -1.5260, -1.1350, -1.2396, -1.8090, -1.3460, -1.3246, -1.2672,
        -1.3985, -1.4052, -1.1576, -1.2277, -1.3039, -1.4184, -0.3964,
        -1.2190, -1.5517, -1.0551, -1.2187, -1.2564, -1.2466, -1.4193,
        -1.1674, -1.4058, -1.3830, -1.4318, -1.4792, -1.4060, -1.4941,
        -1.4987, -1.4581, -1.0912, -1.2818, -1.1240, -1.0493, -1.2244,
        -1.0859, -1.3700, -1.3353, -1.2118, -1.3654, -0.9901, -1.1794,
        -1.5970, -1.1235, -1.2637, -1.4344, -1.3379, -1.3817, -1.1773,
        -1.5256, -1.6575, -1.2940, -1.2532, -1.0937, -1.1866, -1.2201,
        -1.1395, -1.2014, -1.2120, -1.3558, -1.1273, -1.5407, -1.1382,
        -1.3236, -1.2494, -1.4441, -1.3416, -0.9902, -1.0423, -0.9981,
        -1.0645, -1.6268, -1.3610, -1.1455, -1.3685, -1.2742, -1.3451,
        -1.1414, -1.3404, -1.4559, -1.1942, -1.1649, -1.3324, -1.6066,
        -1.0774, -1.1113, -1.1770, -1.6886, -1.0239, -1.3906, -1.0897,
        -1.3960, -1.0741, -1.1783, -1.2729, -1.2481, -1.2168, -1.3417,
        -1.2324, -1.1466, -1.5006, -1.3903, -1.1749, -1.1554, -1.0329,
```

```
        -1.3948, -1.1661, -1.2516, -1.3732, -1.2760, -1.4443, -1.4744,
        -1.3527, -1.0413, -1.6163, -1.2186, -1.1194, -1.2040, -1.2858,
        -1.3171, -0.9953, -1.2038, -0.8914, -1.5464, -1.3528, -1.3497,
        -1.3634, -1.0120, -1.1855, -1.3085, -0.3816, -0.7468, -1.2022,
        -1.4282, -1.1175, -1.3774, -1.4070, -1.1977, -1.0101, -1.3074,
        -1.2933, -1.3813, -1.5206, -1.1710, -1.3490, -1.3420, -1.4194,
        -1.0888, -1.2372, -1.4389, -1.2385, -1.5128, -1.1677, -1.2224,
        -1.3687, -1.3462, -1.4472, -1.6251, -1.2107, -1.4532, -1.6694,
        -1.2161, -1.4354, -1.0472, -1.2740, -1.5246, -1.1216, -1.3486,
        -1.3641, -1.1361, -1.2356, -1.2232, -1.3752, -1.3994, -1.5771,
        -1.3084, -1.3656, -1.4377, -1.2083, -1.2240, -1.0102, -1.0552,
        -1.3348, -1.2637, -1.2486, -1.2333, -1.2564, -1.4201, -1.2519,
        -0.9719, -1.2280, -1.3788, -1.5255, -1.2779, -1.3335, -1.3529,
        -0.9617, -1.4265, -1.3680, -1.3945, -1.3271, -1.0329, -1.2188,
        -1.3532, -1.3638, -1.2036, -1.3030, -0.8859, -1.6914, -1.3328,
        -1.1345, -1.4558, -1.1626, -1.2644, -1.0500, -1.2268, -1.4052,
        -1.3787, -1.4458, -1.6113, -1.5134, -1.2223, -1.2333, -1.2597,
        -1.2215, -1.2255, -1.5154, -1.1231, -1.4700, -1.1718, -1.2111,
        -1.1229, -1.1680, -1.0414, -1.5882, -1.4051, -1.5987]), Parameter containing:
tensor([[[[ 5.8540e-03],
          [ 2.3726e-02],
          [ 4.6027e-03]],

         [[-2.6383e-04],
          [ 1.1567e-02],
          [ 3.0335e-02]],

         [[-2.5389e-02],
          [-2.4772e-02],
          [-1.4848e-02]],

         ...,

         [[ 1.1711e-02],
          [ 2.3046e-02],
          [ 9.4025e-03]],

         [[ 1.8385e-02],
          [-7.5887e-04],
          [-1.1359e-02]],

         [[ 1.9761e-02],
          [ 1.1220e-02],
          [ 2.0887e-02]]],


        [[[ 2.0329e-02],
          [ 1.2378e-02],
```

```
  [ 4.8640e-03]],

 [[ 1.6934e-04],
  [-1.4464e-02],
  [-1.2615e-02]],

 [[ 1.7580e-02],
  [-2.9090e-02],
  [ 5.1456e-03]],

 ...,

 [[-1.5883e-02],
  [ 2.9270e-03],
  [-4.6032e-03]],

 [[-1.3987e-02],
  [-4.0738e-02],
  [-1.4968e-02]],

 [[-1.0835e-02],
  [-8.9399e-03],
  [-5.4107e-03]]],


[[[-2.6138e-02],
  [-1.3466e-03],
  [ 5.1831e-04]],

 [[ 1.0977e-02],
  [ 3.3770e-03],
  [-2.6333e-02]],

 [[ 1.5419e-02],
  [-5.4128e-03],
  [-6.1314e-04]],

 ...,

 [[ 1.6125e-02],
  [-2.2993e-02],
  [-3.9658e-02]],

 [[-2.6120e-02],
  [-3.3812e-02],
  [-2.8895e-02]],

 [[-1.4299e-02],
```

```
  [-1.7758e-02],
  [-8.1425e-03]]],


...,


[[[-1.5417e-02],
  [-1.1824e-02],
  [-2.7278e-02]],

 [[-3.1135e-02],
  [-1.7323e-02],
  [-4.3765e-02]],

 [[ 2.3970e-02],
  [ 3.5028e-02],
  [ 2.8163e-02]],

 ...,

 [[ 2.7946e-02],
  [-8.0751e-03],
  [-3.0835e-03]],

 [[ 2.1000e-02],
  [ 6.4451e-03],
  [ 3.1359e-02]],

 [[-1.2222e-02],
  [-1.2756e-02],
  [-8.5261e-03]]],


[[[ 3.5515e-03],
  [ 9.7480e-03],
  [ 8.9351e-03]],

 [[ 1.3720e-02],
  [-1.0618e-02],
  [ 1.5037e-03]],

 [[ 1.9439e-02],
  [ 1.5061e-03],
  [-1.2722e-02]],

 ...,
```

```
          [[-1.9397e-03],
           [-2.2805e-02],
           [-2.6616e-02]],

          [[ 1.4606e-02],
           [ 1.0279e-02],
           [-1.4576e-02]],

          [[ 1.2007e-02],
           [ 1.4098e-03],
           [ 7.9914e-03]]],


         [[[ 2.1844e-03],
           [ 1.0453e-02],
           [-2.7647e-03]],

          [[-1.5704e-02],
           [-3.7088e-03],
           [-1.7033e-03]],

          [[-6.7287e-03],
           [-4.8820e-03],
           [-2.1472e-02]],

          ...,

          [[ 1.0694e-02],
           [ 1.1032e-02],
           [ 6.2708e-03]],

          [[-3.4753e-02],
           [-2.9671e-02],
           [-2.5844e-02]],

          [[ 8.2142e-03],
           [ 8.1308e-03],
           [ 1.8898e-02]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-1.3436, -1.2938, -1.1346, -0.9481, -0.9995, -0.9560, -1.2585,
        -1.2038, -1.1268, -1.3244, -0.9531, -1.1987, -1.3177, -1.3318,
        -1.1684, -1.3141, -1.1972, -1.3417, -1.4542, -1.4359, -1.0324,
        -0.9687, -0.9776, -1.1858, -0.9727, -1.3239, -1.1707, -1.0785,
        -0.9011, -1.0712, -1.3241, -1.1490, -1.1015, -1.2981, -1.1559,
        -1.1687, -1.3318, -1.2442, -1.2024, -0.2154, -1.0789, -1.3364,
        -1.1484, -0.7845, -1.0827, -1.2238, -1.2874, -0.7683, -1.2087,
        -1.2482, -1.0536, -1.2813, -1.2044, -0.8747, -1.3136,  0.2000,
        -1.1619, -1.2891, -1.3054, -1.1999, -1.1055, -1.0845, -1.1604,
        -1.2611, -1.4118, -0.9947, -1.1757, -0.9907, -1.1388, -1.3224,
        -1.1201, -1.2020, -0.9295, -1.0200, -1.0813, -1.1631, -1.2680,
        -1.0779, -1.2673, -1.3106, -1.1635, -1.0271, -0.7399, -1.2299,
        -1.1362, -1.3329, -1.0971, -0.7702, -1.2093, -1.1697, -1.1043,
        -1.0278, -1.1882, -1.1986, -1.1945, -1.4124, -1.0523, -1.3490,
        -1.3235, -1.2580, -1.0153, -1.0319, -1.0172, -1.2235, -0.8770,
        -1.0995, -1.1195, -0.9131, -0.9801, -1.3953, -1.1325, -1.3213,
        -1.3433, -0.9478, -1.5397, -1.2713, -1.2310, -1.1508, -0.7928,
        -1.3616, -1.0724, -1.0744, -1.3231, -1.2385, -1.2869, -0.8863,
        -0.7765, -1.0923, -1.0945, -0.6767, -1.0335, -1.2569, -0.9905,
        -1.1182, -1.4875, -1.1230, -1.0666, -1.3656, -1.0572, -1.3272,
        -1.1019, -1.0544, -1.2634, -1.0940, -1.0653, -1.2488, -1.1671,
        -1.1408, -1.3298, -1.2333, -1.0047, -1.1991, -1.1156, -1.2864,
        -1.2524, -1.1438, -0.9755, -1.3128, -1.4412, -1.2243, -1.1363,
        -1.2991, -0.8077, -1.2225, -1.2677, -0.1808, -1.1974, -1.3554,
        -1.3748, -0.9808, -1.1598, -1.2266, -1.2922, -0.8898, -1.1075,
        -1.1205, -1.0822, -1.0612, -1.1476, -1.0566, -1.2550, -1.0748,
```

```
        -1.2801, -0.9274, -1.1309, -0.9304, -1.0920, -0.9673, -1.2977,
        -1.1650, -1.0545, -1.1200, -0.9669, -1.3175, -0.9693, -0.9600,
        -0.9420, -1.0781, -0.9815,  0.3066, -1.2401, -1.1441, -0.9845,
        -1.2125, -0.8715, -1.5790, -1.3316, -0.9806, -1.0678, -1.2698,
        -1.0259, -1.0913, -1.1081, -1.2435, -1.1578, -1.2043, -1.1654,
        -1.3763, -0.9268, -1.1986, -0.9958, -1.2620, -1.2274, -1.4203,
        -0.7273, -1.1026, -1.1621, -1.1892, -1.0590, -1.0322, -1.0485,
        -1.3155, -1.1108, -1.2280, -1.0164, -1.0218, -1.2234, -0.9812,
        -1.5053, -1.1868, -1.3396, -0.9880, -1.1429, -1.1904, -1.0100,
        -1.0332, -1.0982, -0.8395, -0.7589, -1.2664, -1.3122, -1.0611,
        -1.0332, -1.2034, -1.0542, -1.1036, -1.7787, -1.1243, -1.1831,
        -1.1127, -0.9673, -1.3537, -1.2862, -1.2931, -1.3128, -0.8756,
        -0.9774, -1.5744, -1.1275, -1.2884, -1.0612, -1.2954, -1.0039,
        -1.5291, -0.8795, -1.2620, -1.2729, -1.0323, -1.1291, -1.3627,
        -1.4106, -1.3762, -1.0605, -0.9687, -1.2059, -0.9770, -1.3293,
        -0.9619, -1.1511, -1.4575, -1.3310, -1.3646, -1.3335, -0.9513,
        -1.1665, -1.2202, -1.2543, -1.5247, -1.2699, -1.2720, -1.1225,
        -1.4395, -1.1449, -1.1241, -0.9931, -1.4942, -1.4269, -1.3774,
        -1.2818, -1.0487, -1.3975, -1.2434, -1.2182, -0.7648, -0.8123,
        -1.1328, -1.2191, -1.1301, -1.4729, -1.0721, -1.0846, -1.0660,
        -1.0459, -0.9472, -1.0646, -1.0609, -1.3151, -1.1259, -1.1906,
        -1.1802, -1.1080, -1.0211, -1.1587, -1.0913, -0.9737, -1.0839,
        -1.0570, -1.2001, -0.9517, -1.2597, -1.3306, -1.0410, -0.7461,
        -1.2536, -0.8803, -0.8938, -1.2273, -1.0084, -1.1660, -1.5286,
        -1.2160, -1.1550, -0.8931, -1.0848, -1.1750, -1.0173, -1.0662,
        -1.2040, -1.3323, -1.1539, -1.2441, -1.2390, -1.2682, -1.0675,
        -1.1156, -1.3475, -1.0250, -1.2233, -1.1793, -1.1723, -0.8054,
        -1.3874, -1.1389, -1.2216, -0.9053, -1.0814, -0.8903, -1.2409,
        -1.0042, -1.4325, -1.1953, -0.8456, -1.3655, -1.0765]), Parameter containing:
tensor([[[[ 1.1040e-02]],

         [[ 7.1928e-03]],

         [[-2.2424e-02]],

         ...,

         [[-9.7186e-03]],

         [[ 3.2637e-03]],

         [[-1.9972e-02]]],


        [[[-2.0232e-03]],

         [[-2.4652e-02]],
```

```
 [[-1.9717e-02]],

 ...,

 [[-2.0119e-02]],

 [[ 7.3641e-02]],

 [[ 1.3597e-02]]],


[[[-1.1607e-02]],

 [[-1.1467e-03]],

 [[ 3.5730e-02]],

 ...,

 [[ 1.7667e-02]],

 [[ 2.4087e-02]],

 [[ 1.0623e-03]]],


...,


[[[ 2.2917e-03]],

 [[-2.0347e-02]],

 [[ 3.4348e-02]],

 ...,

 [[-1.0718e-02]],

 [[ 4.6870e-03]],

 [[-4.0932e-02]]],


[[[ 1.4449e-04]],

 [[-1.9118e-02]],
```

```
          [[-6.7871e-03]],

          ...,

          [[-1.8144e-02]],

          [[ 2.2359e-02]],

          [[-2.2526e-02]]],


         [[[-4.2535e-03]],

          [[ 4.2356e-03]],

          [[-3.0034e-02]],

          ...,

          [[-2.8225e-02]],

          [[ 3.6158e-02]],

          [[-3.1027e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.9093, -0.9395, -0.9283, -0.8574, -0.9728, -0.9019, -0.8983,
        -0.8165, -0.7395, -0.9055, -1.0063, -0.5914, -0.9536, -0.8377,
        -0.8795, -0.9201, -0.7170, -0.7749, -1.0015, -0.8413, -0.7112,
        -0.8673, -1.1099, -1.0025, -0.9579, -0.9725, -0.9038, -0.9006,
        -0.8014, -0.7371, -0.8088, -0.9450, -1.0425, -0.8007, -0.8331,
        -0.8293, -0.9447, -0.8447, -0.8459, -1.0307, -0.9387, -0.8940,
        -1.0319, -0.8301, -0.6497, -0.9036, -0.9546, -0.8135, -0.7319,
        -0.7915, -0.7136, -0.9259, -0.8511, -0.8155, -0.8387, -0.9349,
```

```
        -0.8363, -0.6839, -0.8588, -0.9669, -0.8868, -0.9400, -0.9941,
        -0.8686, -0.8597, -0.9473, -1.1715, -0.8787, -0.6931, -0.7981,
        -0.9080, -0.9041, -0.7646, -0.8971, -0.7701, -0.6684, -0.8667,
        -0.6970, -0.9040, -0.8573, -0.6947, -0.8911, -0.8862, -0.8599,
        -0.8595, -0.9427, -0.9684, -0.8168, -0.8950, -0.9416, -0.8471,
        -0.9073, -0.8259, -0.8545, -0.9026, -1.0243, -0.8640, -0.8314,
        -1.0591, -0.9399, -0.8467, -0.9638, -0.8447, -0.9808, -0.9146,
        -0.9594, -0.8421, -0.9792, -0.4192, -1.1881, -0.8765, -0.7488,
        -1.0226, -0.8933, -0.8310, -0.7725, -0.9216, -0.8658, -0.9461,
        -0.7652, -0.9485, -0.8031, -0.9041, -0.8010, -0.8015, -0.7150,
        -0.9996, -0.9446, -0.9456, -0.8756, -0.9787, -0.6808, -0.9368,
        -1.0818, -0.8289, -0.8094, -0.8752, -0.8121, -0.8317, -0.7960,
        -0.9635, -0.9181, -1.1169, -0.9812, -0.9980, -0.8438, -0.7222,
        -0.8051, -0.9472, -0.8827, -0.8357, -0.9654, -0.9207, -0.9882,
        -0.7954, -0.9223, -0.9643, -0.7345, -0.8551, -0.7293, -1.0715,
        -0.9283, -0.8609, -0.9255, -0.7247, -0.8683, -0.7700, -0.8849,
        -0.9491, -0.8494, -0.9427, -0.8713, -0.9997, -0.9082, -0.8477,
        -0.7643, -0.9788, -0.7439, -0.8136, -0.7822, -0.7616, -0.8868,
        -0.9080, -0.9367, -0.7278, -0.9180, -0.8750, -0.8992, -0.9863,
        -0.8669, -0.6376, -0.7369]), Parameter containing:
tensor([[[[-1.6481e-02]],

         [[-8.4834e-03]],

         [[ 9.9553e-04]],

         ...,

         [[ 9.9746e-03]],

         [[ 8.7661e-03]],

         [[-1.7949e-02]]],


        [[[ 4.7680e-04]],

         [[-4.6105e-03]],

         [[ 4.9487e-03]],

         ...,

         [[ 2.2748e-03]],

         [[ 9.6272e-03]],

         [[-5.3682e-03]]],
```

```
[[[ 1.1906e-02]],

 [[ 3.2059e-04]],

 [[ 6.1321e-03]],

 ...,

 [[ 1.4564e-02]],

 [[-5.9070e-03]],

 [[-1.8257e-02]]],


...,


[[[ 1.5846e-02]],

 [[ 3.2400e-03]],

 [[ 6.3440e-03]],

 ...,

 [[-1.7038e-02]],

 [[-1.0921e-02]],

 [[ 9.9832e-03]]],


[[[ 4.3118e-03]],

 [[ 6.5980e-03]],

 [[-1.6134e-02]],

 ...,

 [[ 6.3357e-03]],

 [[-3.2150e-03]],

 [[-8.0903e-03]]],
```

```
         [[[-6.7840e-03]],

          [[-2.1129e-02]],

          [[-1.7892e-02]],

          ...,

          [[-5.8691e-03]],

          [[-1.0516e-02]],

          [[ 7.8326e-03]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter containing:
tensor([-0.1272, -0.1335, -0.1958, -0.1263, -0.0941, -0.1249, -0.1936,
        -0.1345, -0.0591, -0.0956, -0.1428, -0.1385, -0.1351, -0.1209,
        -0.1379, -0.1797, -0.1195, -0.0571, -0.0918, -0.0945, -0.1641,
        -0.1121, -0.1657, -0.1132, -0.1416, -0.1494, -0.0724, -0.1235,
        -0.1286, -0.1000, -0.1269, -0.1147, -0.1196, -0.1533, -0.1240,
        -0.0568, -0.1031, -0.0680, -0.1310, -0.2015, -0.0873, -0.0821,
```

```
          -0.1471, -0.1298, -0.1749, -0.2324, -0.1373, -0.1695, -0.0686,
          -0.1286, -0.0738, -0.0919, -0.0317, -0.1252, -0.2036, -0.1398,
          -0.1293, -0.1417, -0.1864, -0.1242, -0.1251, -0.1474, -0.1362,
          -0.1731, -0.1181, -0.1417, -0.1042, -0.1175, -0.1102, -0.1052,
          -0.1326, -0.1064, -0.1172, -0.0716, -0.1767, -0.1138, -0.1079,
          -0.1052, -0.1783, -0.1072, -0.1575, -0.1692, -0.1085, -0.1123,
          -0.1003, -0.1334, -0.1321, -0.1286, -0.1105, -0.1247, -0.1519,
          -0.1311, -0.0964, -0.1251, -0.1255, -0.1451, -0.1990, -0.1712,
          -0.1008, -0.1084, -0.1192, -0.0913, -0.0631, -0.0773, -0.0812,
          -0.1075, -0.1646, -0.0904, -0.1748, -0.0501, -0.1682, -0.1170,
          -0.2400, -0.1770, -0.1334, -0.1609, -0.1917, -0.1140, -0.3254,
          -0.2016, -0.1681, -0.0689, -0.0858, -0.1320, -0.1847, -0.1142,
          -0.2168, -0.1160, -0.0977, -0.2080, -0.0987, -0.1218, -0.1465,
          -0.1378, -0.1067, -0.1236, -0.1562, -0.1990, -0.0820, -0.1195,
          -0.1054, -0.1763, -0.2159, -0.0878, -0.0616, -0.1745, -0.1659,
          -0.0659, -0.1010, -0.1430, -0.1594, -0.1604, -0.0881, -0.1394,
          -0.2027, -0.1555, -0.1514, -0.1789, -0.0995, -0.0662, -0.0894,
          -0.0986, -0.0996, -0.1543, -0.1317, -0.1206, -0.1147, -0.0683,
          -0.2400, -0.1612, -0.1276, -0.1041, -0.0739, -0.1793, -0.1335,
          -0.0518, -0.1430, -0.1154, -0.1353, -0.0992, -0.0611, -0.1292,
          -0.1163, -0.0940, -0.1297, -0.0996, -0.1054, -0.0945, -0.2637,
          -0.0779, -0.1610, -0.1438, -0.1580, -0.2800, -0.1478, -0.2159,
          -0.2020, -0.1682, -0.0771, -0.1260, -0.0902, -0.1108, -0.1502,
          -0.1304, -0.1169, -0.2218, -0.1609, -0.1324, -0.1339, -0.1330,
          -0.0428, -0.0882, -0.1011, -0.1200, -0.0823, -0.1692, -0.1383,
          -0.2418, -0.0556, -0.1242, -0.0694, -0.0991, -0.0729, -0.1634,
          -0.0859, -0.0909, -0.1072, -0.1129, -0.0947, -0.1470, -0.1751,
          -0.1154, -0.1434, -0.0389, -0.2561, -0.1092, -0.1150, -0.0787,
          -0.1049, -0.2060, -0.0673, -0.1151, -0.1360, -0.1169, -0.1780,
          -0.1586, -0.1537, -0.0955, -0.1278, -0.1618, -0.0949, -0.1067,
          -0.1909, -0.1080, -0.1715, -0.1183, -0.1375, -0.1020, -0.1705,
          -0.1470, -0.0691, -0.1172, -0.1494, -0.1348, -0.0734, -0.1730,
          -0.0883, -0.0939, -0.1760, -0.1187, -0.0891, -0.1698, -0.1084,
          -0.4798, -0.0837, -0.2965, -0.1211, -0.0991, -0.0708, -0.1308,
          -0.1596, -0.1474, -0.1104, -0.1188, -0.3444, -0.1297, -0.0964,
          -0.0612, -0.1418, -0.1202, -0.2051, -0.1087, -0.1469, -0.1756,
          -0.1402, -0.0427, -0.1799, -0.1183, -0.1358, -0.0847, -0.1153,
          -0.1349, -0.0565, -0.1037, -0.1213, -0.1995, -0.0868, -0.1554,
          -0.0998, -0.0467, -0.1329, -0.1254, -0.1231, -0.0724, -0.1655,
          -0.1033, -0.0754, -0.1287, -0.1171, -0.0973]), Parameter containing:
tensor([[[[-2.1242e-02]],

         [[ 2.1673e-03]],

         [[-5.9891e-03]],

         ...,
```

```
 [[ 5.2235e-03]],

 [[-9.1408e-03]],

 [[ 1.5397e-02]]],


[[[ 1.2123e-02]],

 [[-1.4248e-02]],

 [[-1.4396e-02]],

 ...,

 [[ 4.8304e-02]],

 [[-2.8739e-02]],

 [[-1.4718e-03]]],


[[[-2.2353e-02]],

 [[-3.4103e-02]],

 [[ 1.9720e-03]],

 ...,

 [[-4.7109e-02]],

 [[ 6.2642e-03]],

 [[ 8.9672e-05]]],


...,


[[[-3.9126e-03]],

 [[-8.4841e-03]],

 [[-1.0735e-02]],

 ...,
```

```
           [[-1.3684e-02]],

           [[ 2.6895e-02]],

           [[ 2.9419e-03]]],


          [[[ 1.0151e-02]],

           [[-6.4467e-03]],

           [[-1.1453e-02]],

           ...,

           [[-8.2886e-03]],

           [[ 2.5767e-02]],

           [[-1.5567e-02]]],


          [[[-1.2238e-02]],

           [[ 2.1854e-03]],

           [[-2.0058e-02]],

           ...,

           [[ 1.1184e-02]],

           [[-2.4307e-02]],

           [[ 9.7547e-03]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]), Parameter conta
tensor([-1.3362, -1.2024, -1.4592, -1.5816, -1.2340, -1.1706, -1.0236,
        -1.2166, -1.5457, -1.3586, -1.1783, -1.1098, -1.6178, -1.0909,
        -1.9867, -1.2395, -1.4517, -1.3855, -1.2838, -1.4073, -1.4960,
        -1.2742, -1.2308, -1.3105, -1.3388, -1.1571, -1.3602, -1.1731,
        -1.4436, -1.8604, -1.5832, -1.4354, -1.2707, -1.4474, -1.1745,
        -1.4751, -1.0173, -1.3031, -1.4750, -1.1870, -1.3536, -1.1587,
        -1.2001, -1.2204, -1.1662, -1.5703, -1.2603, -1.2290, -1.1443,
        -1.3982, -1.4860, -1.2317, -1.2051, -1.1841, -1.3934, -1.2165,
        -1.5809, -1.2761, -1.2921, -1.3067, -1.2626, -1.4589, -1.5209,
        -1.1760, -1.2966, -1.4038, -1.1203, -1.5676, -1.1772, -1.1672,
        -1.2253, -1.6482, -1.3262, -1.3415, -1.1758, -1.2918, -1.6658,
        -1.2981, -1.3519, -1.2496, -1.2339, -1.1528, -1.1578, -1.2866,
        -1.3320, -1.2745, -1.2624, -1.2117, -1.3209, -1.2922, -1.5729,
        -1.3845, -1.3058, -1.3460, -1.1072, -1.5964, -1.3338, -1.2354,
        -1.3083, -1.2738, -1.4841, -1.3459, -1.1969, -1.2505, -1.3461,
        -1.3827, -1.2029, -1.4480, -1.1787, -1.1527, -1.2332, -1.2394,
        -1.4706, -1.4057, -1.4503, -1.2235, -1.2854, -1.3690, -1.4081,
        -1.2153, -1.1488, -1.3484, -1.2665, -1.5061, -1.3808, -1.2524,
        -1.3107, -1.3847, -1.3101, -1.2668, -1.3242, -1.1589, -1.4671,
        -1.3397, -1.4021, -1.2295, -1.3572, -1.4322, -1.2698, -1.2122,
        -1.2532, -1.2013, -1.4071, -1.3366, -1.2917, -1.2488, -1.2450,
        -1.3465, -1.3383, -1.3380, -1.3462, -1.3284, -1.2200, -1.2202,
        -1.2570, -1.1265, -1.3954, -1.2506, -1.3650, -1.3204, -1.2926,
         0.4017, -1.5999, -1.2411, -1.2077, -1.4907, -1.4605, -1.3595,
        -1.2914, -1.5158, -1.2213, -1.2825, -1.2761, -1.3486, -1.2291,
        -1.2310, -1.2776, -1.4175, -1.2723, -1.2161, -1.6757, -1.2175,
        -1.1852, -1.3845, -1.2500, -1.2214, -1.2819, -1.4192, -1.3409,
        -1.2097, -1.2612, -1.4244, -1.1402, -1.1750, -1.4858, -1.4272,
        -1.2935, -1.2994, -0.8881, -1.3312, -1.3255, -1.3397, -1.1864,
```

```
          -1.3221, -1.2797, -1.1618, -1.3633, -1.2451, -1.2873, -1.2445,
          -1.2851, -1.3736, -1.2492, -1.2187, -1.1721, -1.2288, -1.2403,
          -1.1473, -1.1465, -1.1755, -1.1446, -1.4730, -1.3415, -1.3326,
          -1.2752, -1.4262, -1.2473, -1.2673, -1.3418, -1.2826, -1.5259,
          -1.4222, -1.2612, -1.2258, -1.1907, -1.1688, -1.3407, -1.5994,
          -1.2794, -1.4395, -1.1902, -1.3165, -1.2241, -1.1688, -1.2417,
          -1.2272, -1.3152, -1.3258, -1.3514, -1.0784, -1.2663, -1.3533,
          -1.4002, -1.3458, -1.3559, -1.3347, -1.3773, -1.2407, -1.2109,
          -1.2017, -1.4682, -1.4041, -1.2746, -1.2770, -1.1973, -1.3807,
          -1.3950, -1.2720, -1.1705, -1.4477, -1.3861, -1.1092, -1.2576,
          -1.4028, -1.6080, -1.3335, -1.2912, -1.1748, -1.2589, -1.2988,
          -1.1719, -1.3128, -1.3060, -1.1774, -1.4407, -1.1425, -1.1672,
          -1.3440, -1.2009, -1.4517, -1.3698, -1.3562, -1.4344, -1.2757,
          -1.1674, -1.3080, -1.1825, -1.2411, -1.3325, -1.2549, -1.1952,
          -1.3113, -1.2218, -1.2220, -1.3749, -1.3423, -1.1099, -1.3167,
          -1.4661, -1.1943, -1.2980, -1.3032, -1.3403, -1.1890, -1.5091,
          -1.2822, -1.1690, -1.2626, -1.3237, -1.3540, -1.4079, -1.3119,
          -1.3519, -1.2581, -1.2521, -1.2399, -1.1598, -1.2559, -1.2797,
          -1.3894, -1.2362, -1.2137, -1.4434, -1.3631, -1.4444, -1.0921,
          -1.2594, -1.2916, -1.3804, -1.4610, -1.2181, -1.5195, -1.2636,
          -1.3254, -1.3427, -1.2824, -1.1915, -1.5219, -1.4588, -1.2178,
          -1.2427, -1.3288, -1.2816, -1.2933, -1.4732, -1.2203, -1.4397,
          -1.3306, -1.5317, -1.2607, -1.3276, -1.1964, -1.2838, -1.2836,
          -1.1507, -1.4193, -1.3047, -1.2556, -1.3042, -1.2886, -1.3330,
          -1.4387, -1.5751, -1.2805, -1.6349, -1.3694, -1.4132, -1.2369,
          -1.2325, -1.3796, -1.1093, -1.4218, -1.2873, -1.4271]), Parameter containing:
tensor([[[[-1.6973e-02, -1.0854e-02, -1.4306e-02]],

          [[-2.3722e-03, -3.5441e-05, -3.6012e-03]],

          [[ 1.3146e-02,  1.2882e-02,  1.2950e-02]],

          ...,

          [[-1.0956e-03,  2.3370e-04, -1.1763e-03]],

          [[-1.3280e-02, -9.4774e-03, -1.4967e-02]],

          [[-9.2637e-03, -5.9417e-03, -9.5811e-03]]],


         [[[-1.5864e-02, -1.2477e-02, -1.6712e-02]],

          [[ 1.5572e-02,  1.2294e-02,  1.4978e-02]],

          [[-6.5854e-03, -3.7833e-03, -5.7769e-03]],

          ...,
```

```
 [[-1.3771e-02, -1.0183e-02, -1.0564e-02]],

 [[ 1.7171e-03,  1.6944e-03,  3.6774e-04]],

 [[ 7.6227e-03,  4.1652e-03,  3.7985e-03]]],


[[[ 1.0850e-02,  7.2258e-03,  1.0753e-02]],

 [[ 4.2998e-03,  2.6903e-03,  5.1138e-03]],

 [[-8.6447e-03, -6.7389e-03, -8.7766e-03]],

 ...,

 [[-2.2981e-03, -5.3260e-03, -1.7064e-03]],

 [[ 5.1735e-03,  2.6904e-03,  2.3070e-03]],

 [[-4.3419e-03, -3.8874e-03, -5.3172e-03]]],


...,


[[[-3.2647e-03, -1.2362e-03, -4.7068e-03]],

 [[-4.7774e-03, -1.9984e-03, -4.2529e-03]],

 [[ 1.8579e-03, -8.4120e-05,  1.7220e-03]],

 ...,

 [[ 8.0405e-03,  1.0918e-02,  9.0519e-03]],

 [[-1.8632e-02, -1.6008e-02, -2.1520e-02]],

 [[-4.5682e-04,  1.6965e-03,  4.1407e-04]]],


[[[ 3.8238e-04, -3.4248e-03, -1.7031e-03]],

 [[-3.4007e-03,  5.6699e-04, -4.1488e-03]],

 [[-9.5093e-03, -7.7018e-03, -8.0058e-03]],

 ...,
```

```
          [[ 4.4991e-03,  3.3261e-03,  1.9938e-03]],

          [[ 5.5970e-03,  6.4490e-03,  7.1239e-03]],

          [[-8.7829e-03, -7.6489e-03, -7.8866e-03]]],


         [[[-3.3940e-03, -1.5090e-03, -4.6967e-03]],

          [[ 4.6161e-03,  2.8763e-03,  5.4698e-03]],

          [[-2.4075e-03, -1.5851e-03, -3.3571e-03]],

          ...,

          [[-5.7203e-03, -8.0488e-03, -7.1160e-03]],

          [[ 1.6382e-03,  1.4756e-03, -2.1473e-04]],

          [[ 9.8030e-04,  3.2939e-03,  6.7626e-04]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.1381, -0.1494, -0.0957, -0.1602, -0.1279, -0.0529, -0.0689,
        -0.0602, -0.1242, -0.1531, -0.1006, -0.2459, -0.0794, -0.1041,
        -0.1366, -0.0622, -0.1425, -0.1614, -0.0960, -0.1480, -0.1785,
        -0.1052, -0.1196, -0.1719, -0.1567, -0.1433, -0.1902, -0.1593,
        -0.1167, -0.1129, -0.1248, -0.1438, -0.0940, -0.1470, -0.1232,
        -0.1284, -0.2558, -0.1963, -0.1837, -0.1311, -0.1688, -0.1077,
        -0.2216, -0.0756, -0.1348, -0.1672, -0.1696, -0.1058, -0.1769,
        -0.1284, -0.0808, -0.1318, -0.1196, -0.0829, -0.0969, -0.1366,
        -0.1126, -0.1863, -0.1037, -0.1524, -0.1569, -0.1702, -0.1406,
        -0.1465, -0.1442, -0.0780, -0.2040, -0.0905, -0.2120, -0.2483,
        -0.1637, -0.1264, -0.1162, -0.1564, -0.1367, -0.1239, -0.1239,
        -0.1134, -0.1234, -0.1308, -0.1760, -0.2423, -0.0914, -0.0969,
        -0.1014, -0.1260, -0.2249, -0.1624, -0.1535, -0.1452, -0.1283,
        -0.1412, -0.1055, -0.1256, -0.2093, -0.1689, -0.1088, -0.1920,
        -0.1460, -0.1813, -0.1499, -0.1411, -0.1079, -0.1562, -0.1412,
        -0.2029, -0.0677, -0.0589, -0.1069, -0.1393, -0.0941, -0.1384,
        -0.0795, -0.2218, -0.1581, -0.1381, -0.0904, -0.1541, -0.0949,
        -0.1263, -0.1612, -0.1802, -0.1193, -0.1029, -0.0879, -0.1873,
        -0.1703, -0.1833, -0.1406, -0.1305, -0.1565, -0.1599, -0.2093,
        -0.1700, -0.0775, -0.1674, -0.1161, -0.1155, -0.0946, -0.1408,
        -0.1175, -0.1110, -0.1297, -0.1527, -0.0971, -0.2695, -0.1480,
        -0.1761, -0.1959, -0.0824, -0.1633, -0.1815, -0.1009, -0.1376,
        -0.0452, -0.1145, -0.1272, -0.1202, -0.0710, -0.1427, -0.1415,
        -0.1855, -0.1282, -0.1648, -0.2458, -0.2277, -0.1511, -0.1863,
        -0.0910, -0.0430, -0.1581, -0.1232, -0.1068, -0.1266, -0.1814,
        -0.1549, -0.2846, -0.1269, -0.1025, -0.0594, -0.0813, -0.1317,
        -0.1476, -0.0884, -0.0905, -0.0720, -0.1203, -0.1317, -0.1524,
        -0.1077, -0.1107, -0.1076, -0.0715, -0.1287, -0.1348, -0.1241,
        -0.1996, -0.0981, -0.1781, -0.0975, -0.1155, -0.2211, -0.2071,
        -0.0288, -0.0992, -0.1571, -0.1991, -0.1524, -0.1652, -0.0924,
        -0.1023, -0.1356, -0.0939, -0.0803, -0.1177, -0.1152, -0.1410,
        -0.1201, -0.2693, -0.1443, -0.1340, -0.1197, -0.1084, -0.1270,
        -0.1303, -0.2187, -0.1014, -0.0998, -0.1867, -0.1174, -0.2149,
        -0.1112, -0.1475, -0.0888, -0.1955, -0.0982, -0.3582, -0.1068,
        -0.0777, -0.1424, -0.1679, -0.1523, -0.1074, -0.0684, -0.0816,
        -0.1336, -0.1324, -0.0985, -0.1331, -0.1241, -0.1379, -0.3111,
        -0.1573, -0.1015, -0.1539, -0.0943, -0.1099, -0.1756, -0.0654,
        -0.1785, -0.1030, -0.1558, -0.1828, -0.1392, -0.2240, -0.1257,
        -0.1336, -0.1751, -0.1068, -0.2839, -0.1337, -0.2099, -0.0930,
        -0.0502, -0.1533, -0.1407, -0.2111, -0.1081, -0.1239, -0.1434,
        -0.2737, -0.1069, -0.1442, -0.1045, -0.1531, -0.1548, -0.2265,
        -0.2061, -0.1269, -0.1175, -0.1574, -0.1707, -0.1954, -0.1259,
        -0.1377, -0.1931, -0.1036, -0.1483, -0.1505, -0.1602, -0.0371,
```

```
        -0.1278, -0.1326, -0.1138, -0.1665, -0.1291, -0.1040, -0.1544,
        -0.0413, -0.0871, -0.1336, -0.0806, -0.0977, -0.1392, -0.1409,
        -0.1242, -0.1959, -0.1319, -0.1707, -0.1183, -0.1192, -0.1649,
        -0.1081, -0.1448, -0.0867, -0.1436, -0.3396, -0.1649, -0.1133,
        -0.0722, -0.1543, -0.1396, -0.1318, -0.0974, -0.1504, -0.2730,
        -0.1426, -0.1167, -0.1362, -0.1280, -0.0184, -0.1579, -0.1010,
        -0.1864, -0.0592, -0.1483, -0.1329, -0.1666, -0.0559, -0.1637,
        -0.1451, -0.1408, -0.1118, -0.0790, -0.0724, -0.1217, -0.1392,
        -0.1581, -0.0590, -0.1750, -0.0933, -0.1947, -0.0686, -0.1419,
        -0.1934, -0.1532, -0.1671, -0.0588, -0.0704, -0.1522, -0.2155,
        -0.0963, -0.0814, -0.1885, -0.1006, -0.0767, -0.1736, -0.1923,
        -0.1765, -0.0971, -0.1965, -0.1642, -0.1306, -0.1164]), Parameter containing:
tensor([[[[-1.9943e-03],
          [-5.5834e-03],
          [-6.5551e-03]],

         [[-1.0420e-02],
          [-1.0676e-02],
          [-1.3219e-02]],

         [[ 8.4481e-02],
          [ 6.5587e-02],
          [ 7.5719e-02]],

         ...,

         [[-1.7724e-02],
          [-1.4688e-02],
          [-1.4949e-02]],

         [[ 3.5402e-02],
          [ 3.3456e-02],
          [ 4.4383e-02]],

         [[-1.3677e-02],
          [-1.1111e-02],
          [-1.1218e-02]]],


        [[[-5.6817e-03],
          [ 1.5918e-03],
          [-2.9875e-03]],

         [[-7.1060e-03],
          [-8.1014e-03],
          [-1.1185e-02]],

         [[-2.9124e-03],
```

```
  [-2.9695e-03],
  [-1.8567e-04]],

 ...,

 [[-6.4355e-03],
  [-3.9423e-03],
  [-5.5575e-03]],

 [[ 6.5356e-03],
  [ 3.7638e-03],
  [ 3.0129e-03]],

 [[-6.4614e-03],
  [-8.8833e-03],
  [-1.3119e-02]]],


[[[-9.2029e-03],
  [-6.8589e-03],
  [-9.0551e-03]],

 [[ 6.8762e-04],
  [-2.4154e-03],
  [-3.3605e-03]],

 [[ 3.0805e-03],
  [ 2.2300e-03],
  [ 4.0227e-03]],

 ...,

 [[-1.2177e-02],
  [-9.8301e-03],
  [-1.3028e-02]],

 [[-2.7422e-04],
  [ 2.6852e-03],
  [ 4.4928e-04]],

 [[ 1.0724e-03],
  [ 2.8576e-03],
  [ 2.5562e-03]]],


...,
```

```
[[[ 3.6444e-03],
  [-9.3060e-05],
  [-6.6402e-04]],

 [[-1.6946e-02],
  [-1.2707e-02],
  [-1.5114e-02]],

 [[ 2.9562e-02],
  [ 3.3025e-02],
  [ 3.3042e-02]],

 ...,

 [[ 1.0947e-02],
  [ 1.4076e-02],
  [ 9.4959e-03]],

 [[ 2.3267e-03],
  [-1.4445e-04],
  [ 2.2130e-04]],

 [[ 7.5538e-03],
  [ 7.3094e-03],
  [ 7.6962e-03]]],


[[[-8.3983e-03],
  [-7.7284e-03],
  [-5.8137e-03]],

 [[-5.2059e-03],
  [-1.6222e-03],
  [-3.2867e-04]],

 [[-1.3400e-04],
  [-1.1557e-03],
  [-1.9525e-03]],

 ...,

 [[ 4.7677e-02],
  [ 4.1358e-02],
  [ 4.3895e-02]],

 [[ 9.3156e-03],
  [ 7.1224e-03],
  [ 8.5837e-03]],
```

```
         [[-3.5738e-03],
          [-1.6498e-03],
          [-6.8880e-03]]],


        [[[-8.9227e-03],
          [-3.1724e-03],
          [-5.0521e-03]],

         [[ 8.3967e-04],
          [-6.1827e-04],
          [ 6.4603e-04]],

         [[ 7.3134e-03],
          [ 6.7668e-03],
          [ 5.8515e-03]],

         ...,

         [[-1.0183e-03],
          [-3.6632e-03],
          [-1.1377e-02]],

         [[-2.2542e-03],
          [-2.2873e-03],
          [ 1.4935e-03]],

         [[ 1.8382e-03],
          [ 2.5526e-03],
          [ 3.1015e-03]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.0305, -0.1517, -0.0755, -0.0281, -0.1651, -0.1134, -0.0279,
        -0.0867, -0.1722, -0.0670, -0.0894, -0.0696, -0.0772, -0.0929,
        -0.0298, -0.0349, -0.1295, -0.0748, -0.0949, -0.0750, -0.0702,
        -0.0485, -0.0770, -0.0882, -0.1826, -0.1940, -0.1033, -0.0735,
        -0.0748, -0.2429, -0.0749, -0.1152, -0.0827, -0.1551, -0.2893,
        -0.1077, -0.1649, -0.1115, -0.0873, -0.2706, -0.0496, -0.0856,
        -0.0713, -0.1028, -0.0997, -0.0549, -0.0968, -0.0853, -0.1530,
        -0.1007, -0.0903, -0.0870, -0.1414, -0.2139, -0.0858, -0.1236,
        -0.0870, -0.0512, -0.1802, -0.1234, -0.0767, -0.1522, -0.1527,
        -0.1338, -0.0897, -0.0742, -0.1488, -0.1594, -0.1220, -0.1597,
        -0.0949, -0.1562, -0.0624, -0.1092, -0.0635, -0.0942, -0.0588,
        -0.0919, -0.1026, -0.1189, -0.0979, -0.0416, -0.0593, -0.1282,
        -0.0375, -0.2028, -0.1358, -0.1118, -0.1070, -0.1732, -0.1879,
        -0.1749, -0.1145, -0.0776, -0.1225, -0.0566, -0.1407, -0.0968,
        -0.1014, -0.1295, -0.0238, -0.0966, -0.1108, -0.1229, -0.1222,
        -0.0957, -0.0845, -0.0571, -0.1308, -0.0560, -0.1475, -0.0736,
        -0.0986, -0.1076, -0.0769, -0.1158, -0.0774, -0.1015, -0.1996,
        -0.1456, -0.0484, -0.0441, -0.0801, -0.1406, -0.1579, -0.0748,
        -0.1077, -0.1087, -0.1296, -0.0720, -0.1276, -0.0428, -0.1308,
        -0.0610, -0.1450, -0.0920, -0.1242, -0.1243, -0.0748, -0.1370,
        -0.1743, -0.1056, -0.0533, -0.2154, -0.1562, -0.0577, -0.0933,
        -0.1575, -0.0752, -0.0888, -0.1188, -0.1396, -0.1116, -0.0891,
        -0.1540, -0.1199, -0.1493, -0.0455, -0.1187, -0.1199, -0.0983,
        -0.1031, -0.1960, -0.1327, -0.1042, -0.0807, -0.0960, -0.0308,
        -0.1742, -0.2146, -0.0538, -0.0655, -0.0699, -0.0672, -0.1745,
        -0.1321, -0.0561, -0.1103, -0.0707, -0.0802, -0.1003, -0.1343,
        -0.0922, -0.0910, -0.0875, -0.1450, -0.1627, -0.0967, -0.1126,
        -0.0877, -0.0989, -0.1081, -0.0765, -0.1346, -0.1241, -0.1538,
        -0.0753, -0.2352, -0.0616, -0.0618, -0.0978, -0.1207, -0.1016,
        -0.1860, -0.1477, -0.0736, -0.1468, -0.0743, -0.1737, -0.0669,
        -0.1015, -0.2333, -0.1401, -0.1847, -0.1333, -0.1197, -0.1266,
        -0.1211, -0.0735, -0.0554, -0.0830, -0.1350, -0.1522, -0.1749,
        -0.0921, -0.0808, -0.1046, -0.1645, -0.1170, -0.0785, -0.1216,
```

```
        -0.1378, -0.1426, -0.1198, -0.0712, -0.1036, -0.0964, -0.1390,
        -0.0775, -0.1389, -0.0731, -0.0466, -0.1811, -0.0788, -0.0716,
        -0.1037, -0.0658, -0.0836, -0.0200, -0.1342, -0.0460, -0.1035,
        -0.0878, -0.0947, -0.1290, -0.0835, -0.0622, -0.1217, -0.1122,
        -0.1124, -0.0433, -0.0890, -0.0820, -0.1014, -0.1307, -0.1732,
        -0.1276, -0.1219, -0.0922, -0.1275, -0.1855, -0.1293, -0.1124,
        -0.0451, -0.1235, -0.1046, -0.1180, -0.1562, -0.1360, -0.1313,
        -0.1003, -0.1325, -0.0845, -0.0985, -0.1167, -0.1235, -0.0713,
        -0.0651, -0.2332, -0.1050, -0.2489, -0.1647, -0.0645, -0.1277,
        -0.1815, -0.1826, -0.0792, -0.0920, -0.0825, -0.1530, -0.1034,
        -0.0610, -0.1701, -0.0446, -0.0663, -0.0810, -0.1659, -0.1023,
        -0.0299, -0.1487, -0.0938, -0.1273, -0.0696, -0.1146, -0.1195,
        -0.1041, -0.0316, -0.0875, -0.1031, -0.0793, -0.0705, -0.1190,
        -0.0862, -0.0647, -0.0601, -0.0597, -0.0734, -0.0837, -0.0922,
        -0.1461, -0.0869, -0.1243, -0.1366, -0.1181, -0.1027, -0.1359,
        -0.0894, -0.0585, -0.0985, -0.0983, -0.1258, -0.0818, -0.1612,
        -0.0639, -0.1439, -0.1216, -0.0831, -0.0646, -0.1859, -0.1322,
        -0.0724, -0.1224, -0.1037, -0.1314, -0.1906, -0.1203, -0.0696,
        -0.0851, -0.0807, -0.1673, -0.0685, -0.1089, -0.1317, -0.0652,
        -0.0940, -0.1048, -0.0902, -0.0700, -0.1554, -0.0681, -0.0634,
        -0.0661, -0.0792, -0.1234, -0.1053, -0.1073, -0.1034, -0.1290,
        -0.0734, -0.0806, -0.0804, -0.1377, -0.1123, -0.1165]), Parameter containing:
tensor([[[[ 5.3577e-03]],

         [[-3.4625e-02]],

         [[ 2.5828e-02]],

         ...,

         [[ 3.0822e-02]],

         [[ 1.5625e-02]],

         [[ 1.2460e-02]]],


        [[[-5.4169e-03]],

         [[ 2.8093e-02]],

         [[ 3.7669e-03]],

         ...,

         [[ 5.0112e-04]],

         [[ 1.1331e-02]],
```

```
 [[-8.3160e-03]]],


[[[-8.4391e-03]],

 [[ 1.0246e-02]],

 [[ 1.3914e-02]],

 ...,

 [[ 1.7275e-02]],

 [[-2.8492e-02]],

 [[-1.6045e-03]]],


...,


[[[ 1.9567e-03]],

 [[-1.2806e-02]],

 [[-2.9563e-02]],

 ...,

 [[-3.3913e-02]],

 [[-1.5829e-02]],

 [[-1.0699e-02]]],


[[[-2.6812e-02]],

 [[-1.2849e-02]],

 [[-4.8005e-02]],

 ...,

 [[-2.3991e-02]],

 [[-8.5986e-03]],
```

```
          [[-1.1969e-02]]],


         [[[ 2.6675e-02]],

          [[ 1.0047e-02]],

          [[-2.5981e-02]],

          ...,

          [[-8.6819e-03]],

          [[-4.5009e-03]],

          [[-3.0824e-02]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
          1.,  1.,  1.,  1.]), Parameter containing:
tensor([-0.9146, -1.1579, -1.2072, -1.0143, -1.0000, -0.9638, -1.4655,
        -1.1017, -0.7426, -0.6465, -1.1102, -0.9768, -0.3746, -0.8242,
        -0.9616, -1.0340, -0.8636, -1.1413, -0.9459, -0.9675, -0.9410,
        -1.2753, -0.9616, -0.2083, -1.2046, -1.0777, -1.1793, -1.9137,
        -1.0437, -1.1435, -1.2569, -0.9986, -1.0228, -0.9966, -1.1951,
        -0.9622, -0.9417, -1.1181, -0.7334, -0.7886, -1.3020, -0.9532,
        -0.9016, -1.2462, -1.2494, -1.1211, -0.9002, -1.0411, -1.0014,
        -1.1185, -1.0431, -1.0478, -0.2735, -1.1187, -1.0952, -0.6776,
        -0.8115, -0.9088, -0.8857, -0.8175, -0.8350, -0.9763, -0.9500,
        -0.8151, -1.1435, -0.8564, -1.0557, -0.5980, -0.8307, -0.4456,
        -1.2547, -1.0345, -0.9190, -1.1097, -0.9260, -1.1849, -0.9877,
        -0.6789,  0.3496, -0.5780, -1.1812, -1.0589, -1.0700, -0.9613,
        -1.2043, -1.2013, -0.2661,  0.2993, -0.7786, -1.0011, -0.8953,
        -0.7429, -0.7282, -0.9864, -0.8847,  0.4076, -1.0686, -0.9056,
        -0.8864, -0.4801, -1.0159, -1.0309, -0.9697, -0.7593, -0.7748,
         0.0408, -0.8352, -1.1208, -0.8253, -0.8297, -0.9502, -1.0682,
        -1.1517, -0.8636, -0.9195, -1.0556, -0.9002,  0.2132, -0.9957,
        -0.9847, -1.1316, -0.7284, -1.1058, -1.0532, -1.3043, -0.9483,
        -0.9182, -1.0255, -0.9147, -0.9650, -0.8423, -0.9056, -0.4654,
        -0.8484, -0.9129, -0.8800, -0.8183, -0.9434, -0.8885, -0.9637,
        -0.4989, -1.3229, -1.0033, -0.9780, -1.0372, -0.5438, -0.8494,
        -1.0628, -1.1046, -0.9727, -0.8924, -0.6046, -0.9524, -0.7640,
        -0.6777, -1.1549, -1.1969, -0.8193, -1.1228, -0.9634, -1.0619,
        -0.7448, -1.2002, -0.9141, -0.8299, -1.2229, -1.0111, -0.8530,
        -1.0132, -0.9307, -0.9246, -1.2290, -1.1048, -0.9632, -0.8022,
        -0.7823, -1.1014, -0.9780, -1.0124, -0.9745, -0.8487, -1.1960,
        -0.9080, -0.9437, -0.8220, -0.9484, -0.9373, -0.7813, -0.9687,
        -0.7944, -0.8358, -0.7984, -0.9775, -1.0098, -0.8681, -0.5370,
        -0.9419, -1.0369, -0.8435, -1.1069, -1.1275, -0.9794, -0.9455,
        -1.2697, -1.0603, -1.0255, -1.0617, -0.7108, -0.9137, -0.7578,
        -0.7304, -1.0232, -0.9533, -1.0341, -1.0257, -1.1644, -1.1422,
        -1.0492, -0.9667, -0.9277, -1.1016, -0.6365, -0.8893, -0.7554,
        -0.7658, -1.0547, -1.1569, -0.9941, -1.1312, -0.9291, -0.8720,
        -0.8824, -1.0268, -1.2437, -0.6872, -1.0191, -0.8872, -0.1143,
        -1.0432, -0.9582, -0.6640, -1.0206, -0.9041, -1.0237, -0.6540,
        -0.8126, -0.9310, -1.1296, -0.6714, -1.1727, -0.8790, -1.1162,
        -0.9282, -0.9137, -1.0615, -1.2764, -0.1121, -0.7623, -0.8721,
        -1.1032, -0.6943, -0.9304, -1.0056, -0.9679, -0.9721, -0.9500,
        -1.2303, -0.9800, -1.0610, -1.1046, -1.0258, -0.8879, -1.0612,
        -1.0376, -0.9375, -0.9428, -0.8081, -0.8193, -1.0035, -1.4241,
        -0.8989, -0.9575, -0.6428, -1.3659, -0.6940, -0.7608, -1.2162,
```

```
          -0.9620, -1.0867, -1.0654, -0.9497, -0.9791, -1.1646, -0.9137,
          -0.8254, -0.8508, -1.0024, -1.0253, -1.0088, -1.0326, -0.1287,
          -1.3117, -0.4493, -0.3625, -1.1436, -1.0419, -0.9105, -0.8124,
          -1.2708, -1.0467, -0.3074, -1.1350, -1.3580, -0.9467, -1.1329,
          -0.7557, -0.7991, -1.0807, -1.0606, -0.2396, -0.7915, -1.0884,
          -0.9204, -1.0123, -0.9319, -1.0876, -0.7276, -0.9660, -0.8273,
          -1.2371, -0.9925, -1.0683, -1.0024, -0.9139, -1.0532, -1.0101,
          -0.7409, -0.7965, -0.7756, -0.9673, -0.9277, -0.8946, -0.3311,
          -0.7254, -0.9968, -1.0752, -0.8900, -1.0571, -0.9808, -0.9846,
          -0.8964, -0.9523, -0.9081, -1.2229, -1.2044, -0.9767, -1.0021,
          -1.0425, -0.9662, -0.8346, -0.9225, -0.9819, -1.0297, -1.3272,
          -0.9450, -0.8525, -1.0049, -0.6202, -0.5944, -0.9156, -1.0919,
          -0.8458, -0.9515, -0.7406, -0.8607, -0.7926, -0.9279, -0.8758,
          -1.0658, -1.2114, -0.6382, -0.7221, -1.0319, -0.3515, -0.7058,
          -1.1520, -0.6847, -0.8079, -1.2106, -0.9703, -0.7628, -1.0122,
          -1.3091, -0.7417, -1.2583, -0.4017, -0.0062, -0.9446, -1.0984,
          -0.8280, -1.1431, -0.7489, -0.8761, -0.9514, -0.5791, -0.9031,
          -1.0043, -0.8914, -0.8402, -0.9834, -1.2468, -1.1131, -0.8591,
          -0.9569, -0.4680, -1.0471, -0.8094, -1.0354, -1.0990, -0.9125,
          -0.9285, -0.9400, -0.7784, -0.7596, -1.0970, -1.0236, -0.9058,
          -1.0805, -0.8134, -0.8677, -0.9021, -0.9851, -0.8704, -0.3764,
          -0.7805, -1.0453, -1.2256, -0.6075, -1.0354, -1.2142, -0.5059,
          -1.3275, -0.8705, -1.0787, -0.6843, -1.1096, -0.9296, -1.1707]), Parameter cc
tensor([[[[-1.5987e-02, -1.3626e-02, -1.8936e-02],
          [-5.8332e-03, -3.9005e-04, -7.3720e-03],
          [-1.8429e-02, -1.2142e-02, -1.9871e-02]],

         [[-1.0251e-02, -1.1986e-02, -9.0889e-03],
          [-6.6207e-03, -9.7569e-03, -8.2758e-03],
          [-3.5155e-03, -6.6197e-03, -2.7147e-03]],

         [[-2.7332e-02, -2.0634e-02, -2.5751e-02],
          [-2.0198e-02, -1.3786e-02, -2.1812e-02],
          [-2.8465e-02, -2.6326e-02, -2.7157e-02]],

         ...,

         [[-1.0716e-02, -8.2076e-03, -1.0479e-02],
          [-1.1458e-02, -4.8359e-03, -1.2320e-02],
          [-1.3013e-02, -5.6467e-03, -1.1736e-02]],

         [[-1.8773e-02, -1.5951e-02, -2.2171e-02],
          [-1.0197e-02, -3.3050e-03, -1.2100e-02],
          [-8.4425e-03, -3.8412e-03, -1.4820e-02]],

         [[ 1.7455e-03,  5.5479e-04,  1.8122e-03],
          [ 1.5785e-03,  4.7689e-03,  1.6180e-03],
          [ 1.3683e-03, -4.5729e-03,  1.8421e-03]]],
```

```
[[[-1.8021e-02, -1.5802e-02, -1.6284e-02],
  [-1.3637e-02, -1.2621e-02, -1.2809e-02],
  [-1.2460e-02, -1.0533e-02, -9.0523e-03]],

 [[ 1.5359e-03,  2.5444e-03,  2.6890e-03],
  [ 2.2054e-03,  5.7701e-04,  3.4508e-03],
  [-1.1316e-03,  2.4089e-03,  2.0398e-04]],

 [[ 5.2848e-03,  5.2997e-03,  1.0597e-02],
  [ 7.8456e-03,  2.1233e-03,  1.1602e-02],
  [ 1.2832e-02,  9.4424e-03,  1.4659e-02]],

 ...,

 [[-1.0004e-02, -1.2146e-02, -1.4461e-02],
  [-7.1869e-03, -7.3439e-03, -8.2180e-03],
  [-7.9186e-03, -1.0221e-02, -1.1031e-02]],

 [[-5.6239e-03,  3.3366e-03, -3.9714e-03],
  [ 8.6960e-03,  1.3944e-02,  8.0065e-03],
  [ 3.9127e-03,  7.4053e-03,  3.9748e-03]],

 [[ 6.0168e-03,  9.1664e-03,  2.2161e-03],
  [-5.0634e-03, -1.9295e-03,  1.4419e-03],
  [-5.8542e-03,  4.1630e-03, -7.4936e-04]]],


[[[ 7.2920e-03, -2.1639e-03,  7.6573e-04],
  [ 9.1991e-03,  5.7013e-03,  5.1328e-03],
  [ 7.5157e-03,  4.0701e-03,  9.1065e-03]],

 [[ 1.3418e-03, -1.8614e-03,  2.8061e-03],
  [ 3.1898e-03,  1.3898e-03,  4.9745e-03],
  [ 4.9160e-03,  2.0129e-03,  2.5805e-03]],

 [[-1.4945e-02, -1.1269e-02, -1.3489e-02],
  [-1.1887e-02, -1.0340e-02, -1.3874e-02],
  [-7.1468e-03, -1.3317e-02, -1.4082e-02]],

 ...,

 [[ 7.3992e-03,  4.1552e-03,  9.0437e-03],
  [ 2.4372e-03, -2.5152e-03,  1.1753e-03],
  [ 1.0133e-02,  8.1771e-03,  8.1848e-03]],

 [[ 1.1861e-02,  9.6554e-03,  9.6335e-03],
```

```
  [ 1.2410e-02,  7.5370e-03,  1.4293e-02],
  [ 1.9782e-02,  1.4961e-02,  1.4501e-02]],

 [[ 1.4282e-03,  4.4770e-03,  3.6091e-03],
  [-7.9330e-03, -4.9275e-03, -2.7714e-03],
  [-6.7231e-03, -7.7545e-03, -6.4745e-03]]],


...,


[[[-1.9593e-03,  5.1678e-03, -4.3902e-03],
  [-5.5170e-03, -3.3662e-05, -7.5492e-03],
  [-9.8514e-03, -9.2743e-03, -1.7870e-02]],

 [[-1.2363e-02, -1.1515e-02, -1.5277e-02],
  [-9.4901e-03, -4.2107e-03, -1.0052e-02],
  [-7.2745e-03, -7.0162e-03, -1.3841e-02]],

 [[-5.2304e-03, -5.8978e-03, -3.7861e-03],
  [ 1.3982e-03, -5.6873e-03, -3.3836e-03],
  [ 1.5699e-03, -3.7531e-03, -1.7290e-03]],

 ...,

 [[-1.3987e-02, -9.7051e-03, -1.3832e-02],
  [-8.9281e-03, -5.0231e-03, -7.4884e-03],
  [-1.6157e-02, -1.2937e-02, -1.4552e-02]],

 [[-5.9354e-03, -1.5771e-03, -4.6145e-03],
  [-4.3691e-03,  5.6665e-03,  7.1428e-04],
  [ 7.6461e-04,  4.2281e-03,  2.2530e-03]],

 [[-3.8109e-03, -2.8061e-03, -4.1552e-04],
  [-5.9774e-03, -4.3133e-03, -2.2303e-03],
  [-1.1914e-02, -1.2056e-02, -1.1978e-02]]],


[[[-1.5880e-03, -4.0442e-03, -4.4847e-03],
  [-9.5915e-04,  3.1304e-03, -1.4415e-03],
  [ 2.5879e-04,  5.3582e-03,  7.2224e-04]],

 [[-1.7702e-02, -1.1212e-02, -1.1435e-02],
  [-1.6053e-02, -1.0177e-02, -1.2733e-02],
  [-1.5558e-02, -9.8903e-03, -1.5141e-02]],

 [[-7.3153e-03, -1.6670e-03, -7.8104e-03],
  [-3.3830e-03,  2.6180e-03, -4.2232e-03],
```

```
          [-1.3705e-02, -9.3777e-03, -1.4891e-02]],

         ...,

         [[-1.6689e-02, -8.2537e-03, -1.1986e-02],
          [-1.6957e-02, -7.7841e-03, -1.2930e-02],
          [-2.0085e-02, -1.1430e-02, -1.7161e-02]],

         [[-2.5288e-03, -9.0674e-04, -3.3729e-03],
          [-3.0682e-04, -9.5916e-04, -7.4434e-03],
          [-1.6086e-03, -4.2635e-03, -1.0360e-02]],

         [[-3.6552e-04,  8.0308e-03,  3.3731e-03],
          [ 5.4448e-03,  1.2094e-02,  7.3604e-03],
          [-1.1239e-03,  2.4105e-04, -2.9925e-03]]],


        [[[ 1.0837e-02, -1.5605e-03,  8.0852e-03],
          [ 8.7566e-03,  1.2543e-03,  1.0621e-02],
          [ 9.8338e-03,  5.4113e-04,  8.4283e-03]],

         [[ 1.1586e-02,  1.0564e-02,  9.9971e-03],
          [ 7.3519e-03,  9.0359e-03,  6.2167e-03],
          [ 6.5084e-03,  7.5243e-03,  4.0713e-03]],

         [[-8.2081e-03, -2.5669e-03, -5.0134e-03],
          [-1.4744e-02, -8.3309e-03, -9.5169e-03],
          [-1.0602e-02, -1.0909e-02, -1.2926e-02]],

         ...,

         [[-1.3842e-02, -1.0576e-02, -1.4215e-02],
          [-1.5837e-02, -7.9796e-03, -1.6836e-02],
          [-1.6280e-02, -1.4748e-02, -2.0137e-02]],

         [[-1.7092e-02, -8.0272e-03, -1.8144e-02],
          [-1.3358e-02, -9.8318e-03, -1.6867e-02],
          [-2.7013e-02, -2.2168e-02, -3.0106e-02]],

         [[ 9.5448e-03,  1.1252e-02,  8.1072e-03],
          [ 9.6362e-03,  4.1850e-03,  5.5939e-03],
          [ 1.1708e-02,  6.8185e-03,  1.0697e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
            1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([ 0.1046, -0.7660, -0.6495, -0.5410, -0.5314, -0.4233, -0.8389,
        -0.4672, -0.4776, -0.5677, -0.7552, -0.7415, -0.8548, -0.8423,
        -0.9132, -0.9167, -0.5288, -0.3333, -0.9825, -0.6092, -0.7415,
        -0.7215, -0.6145, -0.5146, -0.4937, -0.6225, -0.4667, -0.5475,
        -0.5557, -0.8771, -0.8906, -0.7167, -0.6289, -0.5425, -0.0802,
        -0.1535, -0.5316, -0.4532, -0.6222, -0.4681, -0.1126, -0.4954,
        -0.6195, -0.6577, -0.4533, -0.5278, -0.3347, -0.6355, -0.4006,
        -0.9593, -1.0176, -0.4522, -0.5660, -0.4387, -0.9807, -0.6196,
        -0.4658, -0.7617, -0.5301, -0.5534, -1.0963, -1.0889, -0.4642,
        -0.6586, -0.6014, -0.9241, -0.4870, -0.2266, -0.6072, -0.0203,
        -0.4288, -0.5455, -0.5436, -0.5005, -0.5667, -0.5977, -0.5541,
        -0.6521, -0.6545, -0.6647, -0.5045, -1.2384, -0.7118, -0.6701,
        -0.7889, -0.7833, -0.2352, -1.2433, -0.4091, -0.4497, -0.5924,
        -1.2080, -0.5087, -0.6503, -0.5022, -0.4652, -1.1559, -0.3670,
        -0.4279, -0.1172, -0.6854, -0.6896, -0.4485, -0.4333, -0.7947,
        -0.8033, -0.2505, -0.1674, -0.6105, -0.0754, -0.4305, -0.7074,
        -0.4238, -0.9875, -0.6898, -0.6162, -0.5382, -0.4944, -0.7504,
        -0.1110, -0.4111, -0.6765, -0.4214, -0.5531, -0.5938, -0.4972,
        -0.4396, -0.4861, -0.5849, -0.5679, -0.3717, -0.5771, -0.1688,
        -0.7795, -0.7846, -0.3811, -0.6475, -0.5544, -0.1213, -0.5717,
        -0.6522, -0.9380, -0.4973, -1.0005, -0.8650, -0.6035, -0.3793,
        -0.5224, -0.4893, -0.5876, -0.2592, -0.4927, -0.6431, -0.5947,
```

```
        -0.4534, -0.7252, -0.5966, -0.5530, -0.4188,  0.5896, -0.2978,
        -0.4043, -0.1621, -0.7224, -0.4487, -0.7697, -0.4669, -0.4918,
        -0.7817, -0.3115, -0.3936, -0.1057, -0.4342, -0.7306, -0.3269,
        -0.4221, -0.7245, -0.6543, -0.7053, -0.6132, -0.6390, -0.7107,
        -0.8439, -0.8063, -0.5957, -0.7527, -0.3259, -0.4244, -0.7419,
        -0.4453, -0.3909, -0.6805,  0.0650,  0.3319, -0.9632, -0.4717,
        -0.7220, -0.5047, -0.3348, -0.5146, -0.4947, -0.8264, -0.1396,
        -0.3629, -1.0243, -0.4399, -0.8467, -0.4977, -0.4843, -0.7681,
        -0.1265, -0.5827, -0.5369, -0.8698, -0.5992, -0.3910, -0.5605,
        -1.4793, -0.6298, -0.1243, -0.3471, -0.9261, -0.4048, -0.4558,
        -0.0685, -0.6148, -0.9045, -0.4365, -0.3755, -0.5372, -0.3815,
        -0.5757, -0.5505, -0.9268, -0.1799, -0.5102, -0.9880,  2.6487,
        -0.6928, -0.3387, -0.5004, -0.4082, -0.4680, -0.8971, -0.1777,
        -0.5664, -0.5985, -0.5848, -0.4071, -0.0602, -0.8769, -0.6336,
        -0.5277, -0.6312, -0.6737, -1.0326, -0.6136, -0.3950, -0.4394,
        -0.7084, -0.0733, -0.0739, -0.2774, -0.8733, -0.6159, -0.5188,
        -0.6182, -0.5132, -0.6340, -1.2306, -1.0563, -0.4518, -0.5751,
        -1.1228, -0.0835, -0.7090, -0.6023, -1.5022, -0.5744, -0.5878,
        -0.5534, -0.5765, -0.7286, -0.5908, -0.7578, -0.4613, -0.5361,
        -0.1979, -0.0654, -0.3808,  0.1358, -0.6767, -0.7595, -0.5761,
        -0.0735, -0.6799, -0.7131, -0.3634, -0.4837, -1.0312, -1.3189,
        -0.4514, -0.7161, -0.4986, -0.6241, -0.5466, -0.1715, -0.9932,
        -0.5841, -0.4716, -0.3719, -0.5836, -0.0449, -0.9769, -0.5770,
        -0.6495, -0.5816, -0.6544, -0.5186, -0.6534, -0.5538, -0.6545,
        -0.5628, -0.6505, -0.6352, -0.1216, -0.0358, -1.0793, -0.4156,
        -0.4018, -0.5689, -0.6325,  0.1514, -0.8512, -0.7641, -0.5729,
        -0.5701, -0.5104, -0.3681, -0.6615, -0.8940, -0.5479, -0.4858,
        -1.0318, -0.1067, -0.4417, -1.1564,  0.0180, -0.5336, -0.9196,
        -0.5910, -0.4741, -0.5879, -0.5496, -0.7334, -0.1046, -0.5798,
        -0.8525, -1.0855, -0.6076, -0.6009, -1.0196, -0.8224, -0.6142,
        -1.2385, -1.0958, -1.1762, -0.6968, -0.5270, -0.0454, -0.7040,
        -0.3043, -0.5436, -0.8179, -0.3916, -0.6517, -0.5480, -0.2837,
        -0.1931, -0.6079, -0.1614, -0.5155, -0.6281, -0.4676]), Parameter containing:
tensor([[[[ 2.9360e-04, -4.3746e-04, -1.9403e-03]],

         [[ 1.3531e-03,  2.6048e-03,  1.0066e-03]],

         [[-3.5284e-03,  3.6033e-03, -4.0084e-03]],

         ...,

         [[-1.1144e-02, -7.6368e-03, -1.0040e-02]],

         [[-2.4267e-02, -1.8555e-02, -2.4467e-02]],

         [[-7.7894e-03, -8.5697e-03, -7.9448e-03]]],
```

```
[[[ 1.4879e-02,  5.3628e-03,  1.4754e-02]],
 [[ 3.9978e-03,  2.7162e-03,  3.9743e-03]],
 [[-1.4003e-02, -1.1919e-02, -1.4965e-02]],
 ...,
 [[-1.5939e-02, -1.2153e-02, -1.7818e-02]],
 [[-5.7928e-03, -2.0894e-03, -4.3683e-03]],
 [[-2.5603e-02, -1.8142e-02, -2.2219e-02]]],

[[[-1.0281e-02, -8.2021e-03, -9.7709e-03]],
 [[ 1.2358e-03, -1.3313e-03, -2.7746e-03]],
 [[-4.8965e-04,  9.4620e-04, -1.9691e-03]],
 ...,
 [[-1.1624e-02, -5.4688e-03, -1.0978e-02]],
 [[ 1.7468e-02,  1.4232e-02,  1.6433e-02]],
 [[ 8.6488e-03,  3.1439e-03,  8.8398e-03]]],

...,

[[[-1.8470e-02, -1.2469e-02, -1.9476e-02]],
 [[ 1.6390e-02,  1.1238e-02,  1.6583e-02]],
 [[-7.2233e-03, -2.4244e-03, -9.0486e-03]],
 ...,
 [[-9.1918e-04,  7.0815e-03,  2.8322e-03]],
 [[-1.3070e-03,  3.0676e-03, -1.4618e-03]],
 [[ 1.7873e-02,  1.2845e-02,  1.8170e-02]]],
```

```
          [[[-1.4970e-02, -1.1306e-02, -1.3033e-02]],

           [[-1.0167e-02, -3.9795e-03, -1.0933e-02]],

           [[-9.0930e-04,  1.4723e-03, -2.4231e-03]],

           ...,

           [[ 7.2739e-03,  2.5501e-03,  8.4919e-03]],

           [[-2.3703e-03,  3.2121e-03, -2.8104e-03]],

           [[-9.4666e-03, -7.3635e-03, -7.8964e-03]]],


          [[[ 5.2283e-03,  2.4843e-03,  6.0533e-03]],

           [[-5.5906e-03, -3.5754e-03, -3.9850e-03]],

           [[-8.6692e-03, -1.0215e-03, -1.0563e-02]],

           ...,

           [[-6.4960e-04,  1.8311e-03, -3.4584e-04]],

           [[ 1.5083e-03,  4.4509e-03,  1.0990e-03]],

           [[ 1.4569e-02,  9.9396e-03,  1.1701e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
              1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.1458, -0.1680, -0.1793, -0.1400, -0.2904, -0.1361, -0.2150,
        -0.2363, -0.2159, -0.2282, -0.1311, -0.2668, -0.2034, -0.1718,
        -0.1595, -0.2902, -0.2720, -0.1159, -0.1629, -0.1896, -0.1588,
        -0.2182, -0.1619, -0.1441, -0.2742, -0.1571, -0.1859, -0.1747,
        -0.1914, -0.2069, -0.1572, -0.2331, -0.2567, -0.1300, -0.1198,
        -0.1635, -0.1925, -0.1817, -0.1743, -0.1693, -0.1912, -0.1687,
        -0.2058, -0.1892, -0.1511, -0.2122, -0.2530, -0.1559, -0.0878,
        -0.1406, -0.1968, -0.2851, -0.1299, -0.2303, -0.1476, -0.2205,
        -0.2628, -0.1711, -0.1378, -0.1755, -0.1654, -0.2318, -0.0796,
        -0.2393, -0.2082, -0.1833, -0.1672, -0.2153, -0.1425, -0.1665,
        -0.1761, -0.2359, -0.1750, -0.1946, -0.1107, -0.1282, -0.1946,
        -0.2366, -0.1601, -0.2154, -0.2106, -0.1216, -0.1879, -0.1777,
        -0.1790, -0.2693, -0.1991, -0.0847, -0.1672, -0.1507, -0.2914,
        -0.1461, -0.2127, -0.2231, -0.1493, -0.2324, -0.1568, -0.1681,
        -0.1747, -0.2155, -0.3282, -0.1616, -0.1040, -0.1560, -0.3144,
        -0.1237, -0.1420, -0.2414, -0.2535, -0.1665, -0.2188, -0.1800,
        -0.1223, -0.1777, -0.0882, -0.0893, -0.1890, -0.0931, -0.1876,
        -0.1725, -0.2314, -0.2021, -0.1524, -0.3632, -0.1699, -0.1555,
        -0.2669, -0.1952, -0.1172, -0.1717, -0.2510, -0.2312, -0.1556,
        -0.1385, -0.1402, -0.1676, -0.2292, -0.2122, -0.1801, -0.1730,
        -0.3067, -0.1621, -0.2646, -0.1316, -0.2695, -0.1754, -0.1451,
        -0.1274, -0.1451, -0.1673, -0.1297, -0.1766, -0.1722, -0.1582,
        -0.1056, -0.2098, -0.1655, -0.1694, -0.2374, -0.1822, -0.2846,
        -0.1438, -0.1114, -0.1041, -0.1908, -0.1738, -0.2833, -0.1870,
        -0.1945, -0.2333, -0.1727, -0.1555, -0.1857, -0.2460, -0.1976,
        -0.3812, -0.2045, -0.2450, -0.1173, -0.1321, -0.1768, -0.2182,
        -0.1987, -0.1312, -0.1982, -0.1009, -0.1617, -0.2544, -0.1139,
        -0.1750, -0.2213, -0.1655, -0.1561, -0.1276, -0.1725, -0.1613,
        -0.3589, -0.2221, -0.2225, -0.2777, -0.1445, -0.1782, -0.2364,
        -0.2233, -0.1043, -0.0905, -0.1821, -0.1228, -0.1575, -0.2111,
        -0.1290, -0.1177, -0.1390, -0.1378, -0.1805, -0.3181, -0.1793,
        -0.4029, -0.1893, -0.1321, -0.2275, -0.1237, -0.1260, -0.1485,
        -0.0823, -0.1852, -0.1374, -0.1348, -0.1219, -0.1202, -0.1808,
        -0.2144, -0.1656, -0.1490, -0.1360, -0.1891, -0.1856, -0.1470,
        -0.1487, -0.1415, -0.1278, -0.1344, -0.1104, -0.0907, -0.1591,
        -0.1855, -0.2640, -0.2009, -0.2047, -0.1420, -0.1459, -0.1749,
```

```
        -0.1075, -0.1737, -0.1277, -0.0776, -0.1571, -0.1438, -0.3068,
        -0.2225, -0.1807, -0.0678, -0.1894, -0.2113, -0.1281, -0.1908,
        -0.2552, -0.2126, -0.2120, -0.1713, -0.2224, -0.1397, -0.1557,
        -0.1529, -0.2079, -0.1717, -0.1738, -0.1348, -0.1620, -0.1692,
        -0.2515, -0.1197, -0.2154, -0.2132, -0.1470, -0.1331, -0.1447,
        -0.1874, -0.2151, -0.2093, -0.1651, -0.2114, -0.3049, -0.1750,
        -0.1954, -0.2349, -0.1617, -0.2817, -0.1089, -0.1680, -0.2577,
        -0.2861, -0.2054, -0.1985, -0.1551, -0.1407, -0.1325, -0.2113,
        -0.2127, -0.2616, -0.1937, -0.2125, -0.1151, -0.1287, -0.1184,
        -0.2349, -0.2201, -0.1726, -0.1584, -0.1925, -0.2438, -0.2050,
        -0.2315, -0.2340, -0.1536, -0.2442, -0.1325, -0.1247, -0.1370,
        -0.0857, -0.1179, -0.0740, -0.2222, -0.1955, -0.1968, -0.1892,
        -0.1356, -0.2274, -0.1548, -0.1555, -0.0929, -0.1629, -0.1660,
        -0.2314, -0.2527, -0.2090, -0.1736, -0.1297, -0.1840, -0.1907,
        -0.2490, -0.1806, -0.2014,  0.1164, -0.2535, -0.1456, -0.2459,
        -0.1496, -0.1327, -0.1309, -0.1978, -0.1855, -0.1366, -0.1772,
        -0.1721, -0.1608, -0.1949, -0.2526, -0.1771, -0.1288, -0.1402,
        -0.1782, -0.3105, -0.1428, -0.1665, -0.1857, -0.1859, -0.2438,
        -0.1820, -0.1653, -0.2020, -0.1663, -0.1866, -0.1177]), Parameter containing:
tensor([[[[-1.4825e-02],
          [-1.7334e-02],
          [-2.3733e-02]],

         [[-2.9694e-03],
          [-3.2465e-03],
          [-7.9251e-03]],

         [[-1.9919e-02],
          [-1.5216e-02],
          [-2.2658e-02]],

         ...,

         [[ 1.0305e-02],
          [ 1.0634e-02],
          [ 1.1351e-02]],

         [[-1.6468e-02],
          [-1.7959e-02],
          [-2.1188e-02]],

         [[ 6.9413e-03],
          [ 8.8362e-03],
          [ 5.0736e-03]]],


        [[[-2.4410e-02],
          [-1.6724e-02],
```

```
  [-1.6082e-02]],

 [[ 5.4371e-03],
  [ 9.6867e-03],
  [ 4.0779e-03]],

 [[-6.5353e-03],
  [-6.8616e-03],
  [-6.4105e-03]],

 ...,

 [[-9.6351e-03],
  [-5.3455e-03],
  [-1.2621e-02]],

 [[-1.3619e-02],
  [-6.7649e-03],
  [-1.7426e-02]],

 [[-2.0682e-02],
  [-1.4939e-02],
  [-1.7199e-02]]],


[[[-1.0460e-02],
  [-9.0318e-03],
  [-7.1457e-03]],

 [[ 1.3210e-02],
  [ 1.0221e-02],
  [ 1.5230e-02]],

 [[ 1.3685e-02],
  [ 1.5314e-02],
  [ 1.8922e-02]],

 ...,

 [[ 1.9799e-03],
  [ 3.1979e-03],
  [ 7.3868e-04]],

 [[ 1.0402e-02],
  [ 5.3221e-03],
  [ 8.8175e-03]],

 [[ 1.1923e-03],
```

```
  [-1.3214e-03],
  [ 3.9217e-03]]],


...,


[[[ 5.2892e-03],
  [ 2.9932e-03],
  [ 9.3245e-04]],

 [[ 8.5243e-04],
  [-1.7679e-03],
  [-1.9279e-03]],

 [[-2.0924e-03],
  [ 2.0314e-03],
  [-4.8265e-03]],

 ...,

 [[-2.7181e-02],
  [-1.4116e-02],
  [-2.0921e-02]],

 [[-1.7699e-02],
  [-1.1515e-02],
  [-1.7650e-02]],

 [[ 7.8842e-04],
  [-1.4942e-03],
  [ 1.4505e-03]]],


[[[-2.3853e-03],
  [-1.0850e-04],
  [ 1.6302e-03]],

 [[ 1.3942e-02],
  [ 1.2213e-02],
  [ 1.2737e-02]],

 [[ 8.7690e-03],
  [ 6.7067e-03],
  [ 1.3239e-02]],

 ...,
```

```
          [[-2.7293e-03],
           [-1.3653e-03],
           [-7.4112e-03]],

          [[-9.5696e-03],
           [-3.7065e-03],
           [-7.5343e-03]],

          [[-1.1504e-02],
           [-8.9500e-03],
           [-1.2662e-02]]],


         [[[ 8.1250e-05],
           [ 1.7887e-03],
           [-1.3469e-03]],

          [[-1.5411e-02],
           [-1.0970e-02],
           [-1.4819e-02]],

          [[-4.1440e-03],
           [-1.4157e-04],
           [-4.6004e-04]],

          ...,

          [[-2.6322e-03],
           [-4.0265e-03],
           [-1.1246e-02]],

          [[ 2.0508e-02],
           [ 1.9254e-02],
           [ 1.8411e-02]],

          [[ 7.0088e-03],
           [ 5.2396e-03],
           [ 2.4906e-03]]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
```

```
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([-0.1095, -0.1623, -0.2132, -0.2616, -0.1349, -0.1866, -0.1847,
        -0.1728, -0.2081, -0.0914, -0.1155, -0.1812, -0.1603, -0.2219,
        -0.1635, -0.0646, -0.2556, -0.1452, -0.1842, -0.1104, -0.1235,
        -0.1152, -0.1497, -0.1190, -0.1092, -0.1714, -0.1547, -0.1312,
        -0.1813, -0.1472, -0.2075, -0.0960, -0.2349, -0.1525, -0.3066,
        -0.1066, -0.1591, -0.1612, -0.1871, -0.0740, -0.1722, -0.1698,
        -0.1274, -0.1795, -0.1093, -0.1637, -0.2929, -0.1534, -0.2027,
        -0.1243, -0.1420, -0.2161, -0.1734, -0.1175, -0.1854, -0.0753,
        -0.1206, -0.1202, -0.1600, -0.1071, -0.0676, -0.1354, -0.2472,
        -0.1250, -0.1772, -0.1100, -0.1253, -0.1907, -0.1307, -0.1461,
        -0.1787, -0.1946, -0.0889, -0.2034, -0.1965, -0.1373, -0.2029,
        -0.1012, -0.2121, -0.0914, -0.1984, -0.4587, -0.1729, -0.1558,
        -0.1572, -0.1410, -0.1309, -0.1165, -0.1450, -0.1477, -0.1516,
        -0.2315, -0.1038, -0.1151, -0.1335, -0.1338, -0.1270, -0.2101,
        -0.1001, -0.1630, -0.2585, -0.1658, -0.0959, -0.1827, -0.1343,
        -0.2124, -0.1662, -0.3486, -0.1489, -0.2522, -0.1451, -0.1327,
        -0.1703, -0.1076, -0.1855, -0.1770, -0.1538, -0.0893, -0.1626,
        -0.1498, -0.1681, -0.0766, -0.1151, -0.2201, -0.1369, -0.2880,
        -0.1835, -0.1892, -0.2355, -0.2069, -0.1655, -0.0956, -0.1487,
        -0.1469, -0.1403, -0.2013, -0.1675, -0.2140, -0.1464, -0.1138,
        -0.1333, -0.2221, -0.1318, -0.2180, -0.0710, -0.0821, -0.1275,
        -0.1639, -0.1759, -0.1570, -0.1260,  0.0064, -0.0926, -0.1644,
        -0.2470, -0.2173, -0.2022, -0.1694, -0.1539, -0.1738, -0.1093,
        -0.1598, -0.1611, -0.1241, -0.2159, -0.1601, -0.1433, -0.2710,
        -0.1531, -0.1294, -0.1330, -0.1678, -0.1384, -0.2039, -0.1495,
        -0.1882, -0.1278, -0.1211, -0.2135, -0.1699, -0.1117, -0.1449,
```

```
        -0.1827, -0.1346, -0.2517, -0.1383, -0.1025, -0.3495, -0.1278,
        -0.1098, -0.1543, -0.1046, -0.1493, -0.1821, -0.0804, -0.1323,
        -0.1441, -0.1755, -0.1486, -0.1383, -0.0993, -0.0993, -0.1474,
        -0.0899, -0.1527, -0.2207, -0.1599, -0.1962, -0.1997, -0.1358,
        -0.2149, -0.0895, -0.1790, -0.1169, -0.1720, -0.1703, -0.1108,
        -0.1383, -0.1826, -0.1068, -0.1008, -0.0878, -0.1779, -0.1230,
        -0.1448, -0.1349, -0.2371, -0.1712, -0.1608, -0.1731, -0.1545,
        -0.0805, -0.1172, -0.1492, -0.1831, -0.2131, -0.1681, -0.2718,
        -0.1417, -0.1789, -0.1532, -0.1828, -0.1272, -0.2399, -0.2137,
        -0.1429, -0.1380, -0.1512, -0.1398, -0.1149, -0.1401, -0.2467,
        -0.1482, -0.0773, -0.0972, -0.0319, -0.1318, -0.1605, -0.0979,
        -0.0861, -0.1034, -0.1457, -0.1193, -0.1177, -0.1945, -0.1688,
        -0.2336, -0.1048, -0.1792, -0.1133, -0.1061, -0.1060, -0.2788,
        -0.1109, -0.1770, -0.1209, -0.1281, -0.1822, -0.1351, -0.2239,
        -0.2466, -0.0712, -0.1809, -0.1368, -0.1013, -0.1350, -0.0938,
        -0.1498, -0.1831, -0.1432, -0.1002, -0.1221, -0.2218, -0.1885,
        -0.1142, -0.1415, -0.1440, -0.1574, -0.1392, -0.1453, -0.1836,
        -0.1711, -0.0919, -0.1279, -0.1024, -0.1176, -0.1308, -0.2404,
        -0.1671, -0.1640, -0.1132, -0.1216, -0.1627, -0.1493, -0.1196,
        -0.1381, -0.2075, -0.1150, -0.1778, -0.2349, -0.2484, -0.2132,
        -0.2423, -0.2031, -0.1624, -0.0891, -0.1473, -0.1694, -0.1125,
        -0.1106, -0.1675, -0.1415, -0.1619, -0.1743, -0.1085, -0.1188,
        -0.1792, -0.1074, -0.1590, -0.2272, -0.1729, -0.1350, -0.1238,
        -0.1338, -0.1228, -0.3183, -0.0973, -0.1276, -0.1620, -0.1326,
        -0.1747, -0.1828, -0.1252, -0.1619, -0.1508, -0.2823, -0.2007,
        -0.1585, -0.1069, -0.3199, -0.1918, -0.1263, -0.1341, -0.1081,
        -0.1269, -0.1033, -0.1036, -0.1434, -0.1517, -0.1145, -0.1167,
        -0.2437, -0.1150, -0.1091, -0.1500, -0.0786, -0.1552, -0.2557,
        -0.1162, -0.1378, -0.1919, -0.1891, -0.1608, -0.1811]), Parameter containing:
tensor([[[[ 1.3783e-04]],

         [[ 1.5900e-02]],

         [[-2.3943e-02]],

         ...,

         [[-1.8657e-02]],

         [[ 1.2976e-02]],

         [[-1.3959e-02]]],


        [[[-2.5143e-03]],

         [[-2.5841e-03]],
```

```
 [[-7.6451e-03]],

 ...,

 [[-3.5836e-03]],

 [[ 2.2346e-03]],

 [[-9.2025e-03]]],


[[[ 2.2355e-02]],

 [[-6.2734e-03]],

 [[-8.9533e-03]],

 ...,

 [[-2.3423e-02]],

 [[ 6.5233e-03]],

 [[ 1.0619e-02]]],


...,


[[[-1.4097e-02]],

 [[-3.6005e-02]],

 [[-1.7314e-02]],

 ...,

 [[ 9.3584e-03]],

 [[-7.6261e-03]],

 [[ 2.3696e-02]]],


[[[ 3.4010e-03]],

 [[ 1.2201e-02]],
```

```
          [[ 3.5685e-02]],

          ...,

          [[ 3.6786e-03]],

          [[-1.3602e-02]],

          [[ 4.1993e-04]]],


         [[[-7.5281e-03]],

          [[-1.2542e-02]],

          [[ 9.1895e-03]],

          ...,

          [[-9.7625e-03]],

          [[-1.1295e-02]],

          [[-1.2911e-02]]]]), Parameter containing:
tensor([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
         1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.]), Parameter conta
tensor([ 2.6121e-02,  2.6375e-02, -1.0253e-02,  9.2580e-03, -4.5947e-03,
         2.4830e-02,  2.7164e-02, -2.7295e-03,  4.4443e-02, -1.1259e-02,
        -4.0700e-03, -1.1370e-02, -9.8100e-03, -1.0354e-03,  2.3506e-02,
         3.5719e-02,  2.4881e-02, -5.9601e-03, -4.1265e-03, -3.9713e-02,
        -7.9208e-03,  2.5036e-02, -1.0418e-02, -1.0588e-02, -1.7724e-02,
        -1.1561e-01, -1.1733e-02,  4.4575e-02, -2.0677e-01,  4.8964e-02,
        -1.0354e-02,  3.0609e-02,  4.3466e-02,  1.4928e-02, -3.6054e-02,
         6.0750e-02,  4.6276e-03, -2.4563e-01, -7.9396e-04,  2.8367e-06,
```

```
        -1.2176e-01, -3.0645e-02,  2.4224e-03,  3.0632e-02, -6.2193e-03,
         1.6761e-02, -1.6513e-02, -3.7833e-02, -8.1768e-03,  1.3451e-02,
        -2.1606e-02,  2.3638e-02,  3.3766e-02,  2.9197e-02, -7.7040e-02,
        -2.1619e-02, -2.3710e-02, -9.4954e-03,  3.9319e-02, -1.0141e-01,
        -8.7131e-03,  4.3399e-02,  2.7680e-02, -2.5667e-02,  1.9747e-02,
         5.7067e-03, -4.5472e-02,  2.5851e-02,  4.4995e-02, -1.7784e-03,
         1.6150e-02, -7.2420e-02,  5.7845e-03,  7.6838e-03, -3.1690e-03,
         1.8840e-02,  1.4834e-02,  3.3057e-02,  1.7268e-02,  7.7465e-04,
         2.4301e-02,  3.8171e-03,  1.9890e-03,  6.6573e-02,  3.4774e-02,
        -9.6466e-03, -2.5591e-01,  2.6954e-02,  2.1214e-02,  2.4643e-02,
        -2.2106e-01, -4.3731e-02, -1.6206e-02,  5.1864e-03, -1.5775e-02,
        -6.5154e-03, -6.6732e-03,  1.3874e-02,  7.0942e-03, -5.0388e-03,
         1.7943e-02, -2.8549e-02,  6.8641e-02, -1.1394e-01,  5.4728e-03,
        -2.4633e-03, -6.4307e-02,  2.0658e-04, -1.7134e-02,  6.5402e-03,
        -1.7103e-02,  2.0720e-02, -4.8667e-03,  2.9304e-02,  1.5870e-02,
         1.0313e-02,  1.1080e-02,  1.4476e-02, -2.5932e-02, -1.5959e-02,
         2.4390e-02,  1.1967e-02, -3.0477e-02,  2.3300e-02, -1.4065e-02,
         4.4882e-02,  2.3594e-02,  3.9413e-02, -3.9255e-02,  3.7116e-02,
        -2.9142e-02, -1.6127e-02,  3.0124e-03, -3.8458e-02,  3.6049e-02,
        -1.7454e-04, -2.2774e-02, -3.6551e-02,  1.5607e-03,  7.9036e-03,
        -2.4948e-02, -1.1698e-02, -1.4900e-02, -2.1454e-02, -5.2513e-02,
        -4.8735e-02, -1.3189e-02,  1.5072e-02, -2.4456e-02, -3.1935e-02,
        -8.8924e-03, -1.0274e-02,  1.7929e-02,  2.2282e-02,  2.5226e-02,
         2.9967e-02,  3.3035e-02,  4.1258e-02, -1.3459e-02,  1.9177e-03,
         3.4584e-02,  1.2405e-02,  4.7933e-02, -4.9171e-03,  2.7066e-03,
         4.7785e-03, -5.4232e-03, -6.5759e-02,  1.8349e-02,  2.5758e-02,
        -1.9629e-02,  2.0176e-02,  9.4566e-03, -6.8721e-03, -1.0168e-03,
        -2.3699e-02,  4.7564e-04, -7.0149e-03, -7.1443e-02, -2.1543e-02,
        -4.1964e-03, -8.7552e-03, -9.6640e-03,  6.6622e-04,  3.4198e-03,
        -1.1982e-02,  1.0850e-02,  7.9774e-03,  2.5505e-02,  2.4405e-02,
        -2.1212e-02,  6.8545e-03]), Parameter containing:
tensor([[-8.5680e-03, -3.9787e-02, -1.8505e-02,  ...,  5.4861e-02,
         -3.9264e-03, -1.8079e-02],
        [ 2.8369e-02, -1.3823e-02,  1.7401e-02,  ..., -3.5623e-02,
         -2.2936e-02, -1.9412e-02],
        [ 2.2932e-02,  4.3835e-02,  1.3674e-03,  ..., -1.2666e-02,
         -9.5695e-03,  5.1593e-02],
        ...,
        [ 1.6579e-02, -3.1619e-02,  2.6921e-02,  ...,  6.0088e-02,
         -1.7905e-02,  7.7380e-02],
        [-1.9857e-02,  3.4046e-02, -1.1144e-02,  ..., -4.6939e-02,
          1.3986e-02, -4.8510e-03],
        [-1.0640e-03,  9.1229e-02, -6.0601e-03,  ..., -1.3793e-03,
          1.2735e-02,  1.5703e-02]]), Parameter containing:
tensor([-0.2239, -0.4473, -0.0314,  0.0991,  0.5365,  0.0507,  0.1138,
        -0.4573, -0.4628, -0.1953, -0.1230, -0.2094, -0.5255, -0.0784,
         0.0194, -0.1215, -0.0666, -0.3526,  0.4558, -0.3745,  0.4689,
         0.9118,  0.2481,  0.2292, -0.0322,  0.1597,  0.2580,  0.2105,
```

```
 0.0131, -0.6989, -0.4246, -0.1333, -0.0416, -0.2805, -0.0438,
-0.2052, -0.4710, -0.2326,  0.0433, -0.4453,  0.0878,  0.1793,
 0.4447, -0.5760,  0.3600,  0.1292,  0.4366, -0.5819,  0.0795,
 0.1656,  0.0655, -1.0882,  0.5087,  0.2543,  0.7223,  0.3124,
 0.5726,  0.2244,  0.2249,  0.2692,  0.4403, -0.1219, -0.0062,
-0.1746, -0.0918,  0.3662,  0.3913,  0.4352,  0.6040, -0.1042,
 0.1185, -0.1813,  0.1021,  0.0589,  0.3005,  0.4141, -0.3580,
 0.1987,  0.3289,  0.1699, -0.1087,  0.2942,  0.0814, -0.5194,
-0.5703, -0.1084, -0.1920, -0.6297, -0.3263,  0.2017, -0.2583,
-0.2614,  0.7391, -0.2390, -0.4079,  0.2127, -0.0495, -0.2154,
 0.0347, -0.1420, -0.1168, -0.1250,  0.3770, -0.1309,  0.1746,
-0.3185,  0.3417, -0.3162, -0.1402,  0.1158, -0.2930,  0.5254,
-0.4531, -0.7501, -0.1519, -0.4162, -0.3422, -0.6219, -0.3437,
-0.0327, -0.3186, -0.3916, -0.3837, -0.5635, -0.0275, -0.5762,
-0.2842,  0.4454,  0.0596,  0.2129, -0.0944,  0.3604,  0.4571,
 0.0290,  0.0760, -0.2237, -0.0637, -0.0070,  0.2465, -0.1401,
-0.1259,  0.2718, -0.0751,  0.2878, -0.2195, -0.4534,  0.1910,
 0.2728,  0.0069,  0.2462, -0.4380, -0.3749,  0.1191,  0.5800,
 0.0950,  0.0564, -0.0108, -0.3806, -0.1527, -0.0560, -0.6822,
-0.0751,  0.4419,  0.1484, -0.2662, -0.0795,  0.1861, -0.3837,
 0.1045, -0.3742, -0.1586, -0.0241, -0.1497,  0.0642, -0.1752,
 0.0864, -0.2327,  0.0769,  0.2835,  0.0328, -0.0597, -0.2992,
-0.1114, -0.1940,  0.0738,  0.2027, -0.0731, -0.1487, -0.1621,
 0.2681,  0.2087,  0.0219,  0.5848, -0.2561,  0.0797,  0.0932,
 0.2866,  0.0624, -0.1716,  0.1306,  0.3635,  0.0659,  0.1223,
 0.3866,  0.4068, -0.1561, -0.3760, -0.1041, -0.0662,  0.1487,
-0.2355,  0.0603, -0.1016, -0.1126, -0.0822, -0.2036, -0.1716,
 0.3664,  0.0966,  0.0251,  0.0461, -0.3221,  0.0225,  0.1298,
-0.2917,  0.1741,  0.1720,  0.1020, -0.2244,  0.3213, -0.0309,
 0.0282,  0.2124,  0.2936,  0.0810,  0.0306,  0.0791, -0.0884,
 0.3530,  0.5294,  0.1096,  0.0754, -0.1030, -0.2635, -0.1216,
 0.2566, -0.0840,  0.0615,  0.0702,  0.0868,  0.0787, -0.0905,
 0.3464, -0.5345, -0.2175, -0.2319,  0.0071, -0.1887,  0.3552,
 0.1506, -0.0224,  0.1192, -0.0640,  0.1663, -0.2730, -0.0444,
 0.3491, -0.2373, -0.6047,  0.0572,  0.4471,  0.1062,  0.1987,
 0.4181,  0.1735, -0.0170,  0.3368,  0.1724,  0.1492,  0.1179,
 0.0455, -0.1205, -0.0241, -0.0828,  0.0237,  0.0966, -0.3344,
 0.3570,  0.3380,  0.4253,  0.2787, -0.2585, -0.1987,  0.0525,
-0.0727,  0.0959, -0.2647,  0.3408, -0.0250,  0.1882, -0.0129,
-0.0204, -0.1062, -0.2103,  0.0627,  0.4876,  0.0832, -0.0518,
-0.0372, -0.2693,  0.1043,  0.0048,  0.2581,  0.2837,  0.1190,
 0.1058, -0.0730,  0.1878,  0.4971, -0.2056,  0.0530, -0.2705,
-0.3237, -0.3382, -0.2563, -0.2603, -0.1006, -0.4722, -0.3041,
-0.0220,  0.1499,  0.0646,  0.1028, -0.4500,  0.0601, -0.3039,
 0.6534, -0.2660, -0.7818, -0.1705, -0.1425, -0.6362, -0.0796,
 0.2187, -0.0683, -0.4103,  0.0923,  0.0359, -0.2346, -0.1638,
-0.0765,  0.3374,  0.0962,  0.1335, -0.4515, -0.7218, -0.2000,
-0.1647,  0.3197,  0.0332, -0.2599,  0.1179,  0.1085, -0.0015,
```

```
-0.2368, -0.0126,  0.1045,  0.1494,  0.1416,  0.0747, -0.1155,
-0.0810,  0.0499,  0.1962, -0.1555, -0.2222, -0.1721, -0.3270,
 0.1665,  0.2496, -0.1026,  0.2116, -0.1749, -0.0817, -0.4208,
-0.0260,  0.0621, -0.0586, -0.1669, -0.0966, -0.5856, -0.2318,
-0.6634, -0.4916, -0.2495, -0.3463, -0.8607, -0.4322, -0.1512,
 0.1826,  0.1013, -0.3569,  0.2005,  0.1850,  0.1283,  0.2091,
-0.1758,  0.1871, -0.5070, -0.0771, -0.1124, -0.3046, -0.1739,
 0.0016, -0.2653, -0.1910,  0.0854, -0.0651,  0.5646,  0.3962,
-0.1899,  0.1253,  0.2957,  0.1530,  0.4531,  0.2264, -0.2172,
-0.3577, -0.3203, -0.1118, -0.3181, -0.0078,  0.2973, -0.1371,
 0.2219,  0.0799,  0.1838,  0.0774, -0.0005, -0.2671,  0.2516,
 0.2669,  0.0753, -0.0905,  0.2169,  0.3970,  0.2983, -0.7014,
-0.3127,  0.1155, -0.3450, -0.1142, -0.3292,  0.2966,  0.1662,
 0.1831, -0.4048,  0.2133, -0.0425,  0.0959,  0.5987, -0.4774,
-0.1433, -0.0477,  0.0326,  0.3368, -0.0984, -0.1224, -0.0366,
-0.4135,  0.0678, -0.2036,  0.0255,  0.3769, -0.2030, -0.4228,
-0.9055, -0.4680, -0.0741,  0.2359,  0.3095, -0.0512,  0.0769,
-0.0356,  0.4957, -0.0699,  0.1661,  0.2639,  0.2302,  0.0156,
-0.1694, -0.6490, -0.1494,  0.3279, -0.3650,  0.2804, -0.5180,
-0.0842,  0.0890, -0.0458,  0.3399,  0.0302, -0.5038, -0.1707,
 0.1840, -0.2864, -0.0633, -0.1058,  0.4799, -0.3035, -0.0848,
 0.3916,  0.3580,  0.0720, -0.3049, -0.0920, -0.3395, -0.1559,
-0.3305,  0.0360, -0.1036, -0.1429, -0.1195,  0.1933, -0.6371,
-0.5547,  0.3216,  0.4647, -0.1682,  0.4156,  0.3149, -0.2757,
 0.2112,  0.7734,  0.0436, -0.2366,  0.2704, -0.3279,  0.0716,
 0.3913, -0.1126, -0.3596,  0.1882, -0.1130,  0.0234, -0.3563,
-0.3659, -0.0612,  0.0850,  0.1949, -0.1693,  0.1243,  0.1933,
 0.4525,  0.0146, -0.1554, -0.5297,  0.2446, -0.0065,  0.1530,
-0.4800, -0.2270, -0.7888,  0.1138, -0.3214, -0.1461, -0.1626,
 0.4296, -0.3454, -0.3413, -0.2050, -0.3237,  0.1906,  0.2316,
 0.2014,  0.1130, -0.4391, -0.5672,  0.0603, -0.0776,  0.0570,
-0.2282,  0.1726, -0.2208, -0.0358, -0.0743, -0.3754,  0.1717,
-0.5669, -0.2972, -0.0707,  0.1264, -0.0064,  0.2045, -0.4224,
-0.2499, -0.0860, -0.1705,  0.1082, -0.1256,  0.0824,  0.0907,
 0.2465,  0.0728, -0.1525,  0.1719, -0.4150, -0.0961,  0.3729,
-0.0867,  0.1263, -0.7021, -0.3920, -0.1146, -0.5244,  0.2281,
-0.2314,  0.4672,  0.2564, -0.1377,  0.1830, -0.0150,  0.6340,
 0.5324,  0.4955, -0.0768,  0.5202,  0.1555, -0.2371,  0.2908,
-0.0873,  0.0292,  0.4831,  0.0252, -0.2666, -0.1985, -0.0467,
-0.4213,  0.2363, -0.0925,  0.4788, -0.0071, -0.2643, -0.4537,
 0.4576, -0.0149,  0.2310, -0.2233,  0.6984,  1.0384,  0.5813,
 0.2217,  0.1540,  0.0075,  0.0632,  0.2132, -0.0371, -0.0130,
-0.1458,  0.3285,  0.2481, -0.0096,  0.2884, -0.0205,  0.2151,
 0.0784, -0.0652,  0.0834, -0.0756, -0.1585, -0.2984,  0.3067,
-0.0796,  0.2982, -0.4434, -0.1184, -0.5549,  0.2904, -0.1236,
 0.1343,  0.4402,  0.8869,  0.3632,  0.4249, -0.2677,  0.1790,
 0.0181, -0.2462, -0.2388, -0.0427, -0.2923, -0.3024, -0.2442,
-0.1680, -0.5931, -0.1815, -0.3332, -0.2856,  0.2796, -0.3165,
```

```
 0.0387,  0.3267,  0.1899,  0.1916, -0.2846,  0.0863,  0.0373,
-0.5293,  0.2855, -0.1247, -0.4626,  0.1003, -0.1781,  0.4136,
 0.1089, -0.5584,  0.2983, -0.0154,  0.1965, -0.0442,  0.2595,
-0.0495,  0.3204, -0.4179, -0.1303, -0.0332, -0.0632, -0.6080,
-0.2371, -0.2001, -0.0422,  0.0634, -0.1080,  0.5642,  0.3090,
-0.0935,  0.1881,  0.2095, -0.1299, -0.0740, -0.3746,  0.1328,
 0.0240,  0.2362,  0.0855,  0.1634,  0.3198, -0.2484, -0.4479,
-0.1844,  0.0866, -0.1427,  0.0912,  0.0587,  0.4959,  0.0010,
-0.5889, -0.3907, -0.4883,  0.3062,  0.3498,  0.3523,  0.1559,
-0.0308,  0.1929, -0.0685, -0.6261,  0.0044,  0.4700,  0.4781,
-0.5767, -0.2262, -0.2079,  0.0160, -0.1046, -0.1330, -0.1273,
 0.0310, -0.3842, -0.3832,  0.0912, -0.2381,  0.1052,  0.1996,
 0.2105,  0.4014, -0.4431, -0.6533, -0.1047,  0.3862, -1.0717,
-0.1804,  0.0127, -0.0915,  0.1865,  0.0437, -0.2155,  0.0790,
 0.0973,  0.7931, -0.6601, -0.1545, -0.0225, -0.3021,  0.0060,
 0.2761,  0.0156, -0.2437, -0.2394,  0.4232,  0.1031, -0.5399,
-0.3889, -0.0229,  0.2159,  0.3450, -0.6137,  0.0026,  0.0095,
-0.2752, -0.5640, -0.2938, -0.3720,  0.3574, -0.1109,  0.7501,
-0.0808, -0.3174,  0.4206, -0.3122, -0.1209, -0.0375, -0.2896,
-0.1293,  0.7486, -0.6319, -0.2057,  0.0555,  0.3015, -0.0710,
-0.1825, -0.0760, -0.2376, -0.1106,  0.6283,  0.4598,  0.1422,
-0.1952,  0.1641, -0.1565, -0.7445, -0.1856,  0.1310, -0.0930,
-0.0285, -0.1674, -0.4556, -0.6375,  0.2527, -0.6240,  0.1104,
-0.0213, -0.1651, -0.1757, -0.1530, -0.4082, -0.1692, -0.3374,
-0.0507,  0.2472, -0.1212,  0.0068,  0.1385,  0.1037, -0.0538,
 0.3732,  0.1448, -0.1156, -0.2203, -0.0426,  0.0180,  0.2079,
-0.1493, -0.1629,  0.2149,  0.6102, -0.3049, -0.6497,  0.1138,
-0.2299, -0.0986, -0.6853,  0.3830, -0.2512,  0.2667,  0.5090,
 0.3124,  0.1725,  0.2510,  0.1068,  0.3136,  0.3421,  0.4562,
 0.5376,  0.5573,  0.6012,  0.4194,  0.0189,  0.6494,  0.0983,
 0.1506,  0.4846,  0.0564,  0.2430,  0.1324, -0.0935,  0.1296,
-0.7217, -0.0506,  0.1920, -0.1049, -0.5905,  0.1951,  0.0652,
-0.2395,  0.1030,  0.1950, -0.0222, -0.3406,  0.2971,  0.3087,
-0.1659, -0.1407, -0.0437, -0.1251,  0.4205, -0.0101, -0.2944,
 0.0713, -0.3985,  0.0777, -0.1455, -0.1891, -0.5079, -0.2376,
 0.1452,  0.2436, -0.2307,  0.1170,  0.2599,  0.4578,  0.1793,
-0.2000, -0.3301, -0.1826, -0.3331, -0.3359,  0.0806,  0.4725,
 0.0898, -0.3807,  0.1566, -0.2583,  0.3414, -0.0031, -0.1465,
 0.2953,  0.5217,  0.5306,  0.0343,  0.6397, -0.3678,  0.5849,
 0.3462,  0.7571,  0.7399,  1.0331,  1.2305,  0.5972,  0.7071,
 0.6412,  0.0770, -0.1237,  0.1340,  0.8507,  0.4015, -0.0361,
-0.0660, -0.4562,  0.0738, -0.1665, -0.0394,  0.0169, -0.3657,
-0.0837, -0.2734, -0.0724, -0.4848, -0.1534,  0.2427])]
```

In [126]:
```
num_classes = 133
model_transfer.AuxLogits.fc = nn.Linear(768, num_classes)
model_transfer.fc = nn.Linear(2048, num_classes)
```

In [127]:
```
if use_cuda:
```

```
model_transfer = model_transfer.cuda()
```

Question 5: Outline the steps you took to get to your final CNN architecture and your reasoning at each step. Describe why you think the architecture is suitable for the current problem.

Answer: I chose the Inception v3 network, since this includes kernels of multiple sizes to detect a variety of salient features. The model is flexible and each inception module can capture salient features at different levels. I froze all layers except for Layer 7, the last layer and the AuxLogits layer, so the network can learn the parameters in these layers only. I also resized the images to 299 as Inception expects this as input.

1.1.14 (IMPLEMENTATION) Specify Loss Function and Optimizer

Use the next code cell to specify a loss function and optimizer. Save the chosen loss function as `criterion_transfer`, and the optimizer as `optimizer_transfer` below.

```
In [128]: criterion_transfer = nn.CrossEntropyLoss()
          optimizer_transfer = optim.SGD(params_to_update, lr=0.05)
```

1.1.15 (IMPLEMENTATION) Train and Validate the Model

Train and validate your model in the code cell below. Save the final model parameters at filepath `'model_transfer.pt'`.

```
In [129]: def inception_train(n_epochs, loaders, model, optimizer, criterion, use_cuda, save_pat
              """returns trained model"""
              # initialize tracker for minimum validation loss
              valid_loss_min = np.Inf

              for epoch in range(1, n_epochs+1):
                  print('Now in epoch ', epoch, '...')
                  # initialize variables to monitor training and validation loss
                  train_loss = 0.0
                  valid_loss = 0.0

                  ###################
                  # train the model #
                  ###################
                  model.train()
                  for batch_idx, (data, target) in enumerate(loaders['train']):
                      # move to GPU
                      if use_cuda:
                          data, target = data.cuda(), target.cuda()
                      ## find the loss and update the model parameters accordingly
                      ## record the average training loss, using something like
                      ## train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_l
                      optimizer.zero_grad()
                      outputs, aux_outputs = model(data)
                      loss1 = criterion(outputs, target)
                      loss2 = criterion(aux_outputs, target)
```

```
            loss = loss1 + 0.4*loss2
            loss.backward()
            optimizer.step()
#             train_loss += loss.item()*data.size(0)
            train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_loss

        ######################
        # validate the model #
        ######################
        model.eval()
        for batch_idx, (data, target) in enumerate(loaders['valid']):
            # move to GPU
            if use_cuda:
                data, target = data.cuda(), target.cuda()
            ## update the average validation loss



            optimizer.zero_grad()



            with torch.no_grad():
                output = model(data)
            loss = criterion(output, target)
#             valid_loss += loss.item()*data.size(0)
            valid_loss = valid_loss + ((1 / (batch_idx + 1)) * (loss.data - valid_loss

#         train_loss = train_loss/len(loaders['train'])
#         valid_loss = valid_loss/len(loaders['valid'])

        # print training/validation statistics
        print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.format(
            epoch,
            train_loss,
            valid_loss
            ))

        ## TODO: save the model if validation loss has decreased
        if valid_loss <= valid_loss_min:
            print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.f
            valid_loss_min,
            valid_loss))
            torch.save(model.state_dict(), 'model_transfer.pt')
            valid_loss_min = valid_loss

    # return trained model
    return model
```

```
In [132]: loaders_transfer = {
              'train': train_loader,
              'valid': valid_loader,
              'test': test_loader
          }

          # train the model
          model_transfer = inception_train(15, loaders_transfer, model_transfer, optimizer_trans

          # load the model that got the best validation accuracy (uncomment the line below)
          model_transfer.load_state_dict(torch.load('model_transfer.pt'))
```

```
Now in epoch  1 ...
Epoch: 1         Training Loss: 2.895938         Validation Loss: 1.359395
Validation loss decreased (inf --> 1.359395).  Saving model ...
Now in epoch  2 ...
Epoch: 2         Training Loss: 2.506086         Validation Loss: 1.193242
Validation loss decreased (1.359395 --> 1.193242).  Saving model ...
Now in epoch  3 ...
Epoch: 3         Training Loss: 2.221446         Validation Loss: 1.072665
Validation loss decreased (1.193242 --> 1.072665).  Saving model ...
Now in epoch  4 ...
Epoch: 4         Training Loss: 1.997028         Validation Loss: 0.997612
Validation loss decreased (1.072665 --> 0.997612).  Saving model ...
Now in epoch  5 ...
Epoch: 5         Training Loss: 1.804666         Validation Loss: 0.901323
Validation loss decreased (0.997612 --> 0.901323).  Saving model ...
Now in epoch  6 ...
Epoch: 6         Training Loss: 1.640287         Validation Loss: 0.733516
Validation loss decreased (0.901323 --> 0.733516).  Saving model ...
Now in epoch  7 ...
Epoch: 7         Training Loss: 1.510859         Validation Loss: 0.729790
Validation loss decreased (0.733516 --> 0.729790).  Saving model ...
Now in epoch  8 ...
Epoch: 8         Training Loss: 1.377893         Validation Loss: 0.677639
Validation loss decreased (0.729790 --> 0.677639).  Saving model ...
Now in epoch  9 ...
Epoch: 9         Training Loss: 1.274844         Validation Loss: 0.705721
Now in epoch  10 ...
Epoch: 10         Training Loss: 1.192047         Validation Loss: 0.644694
Validation loss decreased (0.677639 --> 0.644694).  Saving model ...
Now in epoch  11 ...
Epoch: 11         Training Loss: 1.101702         Validation Loss: 0.674278
Now in epoch  12 ...
Epoch: 12         Training Loss: 1.030284         Validation Loss: 0.564720
Validation loss decreased (0.644694 --> 0.564720).  Saving model ...
Now in epoch  13 ...
Epoch: 13         Training Loss: 0.964691         Validation Loss: 0.549696
```

```
Validation loss decreased (0.564720 --> 0.549696).  Saving model ...
Now in epoch  14 ...
Epoch: 14         Training Loss: 0.905367         Validation Loss: 0.618791
Now in epoch  15 ...
Epoch: 15         Training Loss: 0.855395         Validation Loss: 0.543069
Validation loss decreased (0.549696 --> 0.543069).  Saving model ...
```

1.1.16 (IMPLEMENTATION) Test the Model

Try out your model on the test dataset of dog images. Use the code cell below to calculate and print the test loss and accuracy. Ensure that your test accuracy is greater than 60%.

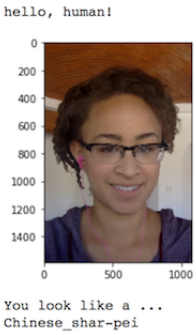
```
In [133]: test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
```

```
Test Loss: 0.552015


Test Accuracy: 88% (741/836)
```

1.1.17 (IMPLEMENTATION) Predict Dog Breed with the Model

Write a function that takes an image path as input and returns the dog breed (`Affenpinscher`, `Afghan hound`, etc) that is predicted by your model.

```
In [136]: ### TODO: Write a function that takes a path to an image as input
          ### and returns the dog breed that is predicted by the model.

          # list of class names by index, i.e. a name can be accessed like class_names[0]
          # class_names = [item[4:].replace("_", " ") for item in loaders_transfer['train'].clas
          class_names = [item[4:].replace("_", " ") for item in datasets.ImageFolder("/data/dog_

          def predict_breed_transfer(img_path):
              # load the image and return the predicted breed

              img_transform = transforms.Compose([
              transforms.Resize(size=320),
              transforms.CenterCrop(299),
              transforms.ToTensor(),
              transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
          ])

              img = Image.open(img_path)
              img = img_transform(img)
              img = img.unsqueeze(0)
              model_transfer.eval()
              if use_cuda:
                  img = img.cuda()
```



Sample Human Output

```
        with torch.no_grad():
            prediction = model_transfer(img)
#       prediction = prediction.data.numpy().argmax()
        prediction = prediction.cpu().numpy().argmax()

        return class_names[prediction] # predicted class index
```

In [ ]:

Step 5: Write your Algorithm

Write an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then, - if a **dog** is detected in the image, return the predicted breed. - if a **human** is detected in the image, return the resembling dog breed. - if **neither** is detected in the image, provide output that indicates an error.

You are welcome to write your own functions for detecting humans and dogs in images, but feel free to use the `face_detector` and `human_detector` functions developed above. You are **required** to use your CNN from Step 4 to predict dog breed.

Some sample output for our algorithm is provided below, but feel free to design your own user experience!

1.1.18 (IMPLEMENTATION) Write your Algorithm

In [144]:
```
### TODO: Write your algorithm.
### Feel free to use as many code cells as needed.

def run_app(img_path):
    ## handle cases for a human face, dog, and neither
    result = dog_detector(img_path)
    if result == True:
        print('\nDOG DETECTED. CLASSIFYING BREED...')
        breed = predict_breed_transfer(img_path)
        print('DOG BREED IS ', breed)
    if result == False:
        is_human = face_detector(img_path)
```

```
    if is_human == True:
        print('\nHUMAN DETECTED. CLASSIFYING RESEMBLING DOG BREED...')
        breed = predict_breed_transfer(img_path)
        print('YOU LOOK LIKE A ', breed)
    else:
        print('\nWE\'RE SORRY, NEITHER A DOG OR HUMAN WAS DETECTED. (PERHAPS YOU A
```

Step 6: Test Your Algorithm

In this section, you will take your new algorithm for a spin! What kind of dog does the algorithm think that *you* look like? If you have a dog, does it predict your dog's breed accurately? If you have a cat, does it mistakenly think that your cat is a dog?

1.1.19 (IMPLEMENTATION) Test Your Algorithm on Sample Images!

Test your algorithm at least six images on your computer. Feel free to use any images you like. Use at least two human and two dog images.

Question 6: Is the output better than you expected :) ? Or worse :(? Provide at least three possible points of improvement for your algorithm.

Answer: (Three possible points for improvement) The output is better than I expected. Some points for improvement: 1) Re-train with less layers frozen, 2) Improve data augmentation steps, 3) Collect more training data!

In [145]:
```
## TODO: Execute your algorithm from Step 6 on
## at least 6 images on your computer.
## Feel free to use as many code cells as needed.

## suggested code, below
for file in np.hstack((human_files[:3], dog_files[:3])):
    run_app(file)
```

```
HUMAN DETECTED. CLASSIFYING RESEMBLING DOG BREED...
YOU LOOK LIKE A  Cane corso

HUMAN DETECTED. CLASSIFYING RESEMBLING DOG BREED...
YOU LOOK LIKE A  Dogue de bordeaux

HUMAN DETECTED. CLASSIFYING RESEMBLING DOG BREED...
YOU LOOK LIKE A  Cocker spaniel

DOG DETECTED. CLASSIFYING BREED...
DOG BREED IS  Mastiff

DOG DETECTED. CLASSIFYING BREED...
DOG BREED IS  Mastiff
```

```
WE'RE SORRY, NEITHER A DOG OR HUMAN WAS DETECTED. (PERHAPS YOU ARE AN ALIEN?)
```

In [ ]: