final_model

June 27, 2024

1 Animal Competition (15%)

Indented block

For the non-competition mode, we will use the Animal (https://cloudstor.aarnet.edu.au/plus/s/cZYtNAeVhWD6uBX) dataset. This dataset contains images of 151 different animals.

The dataset contains a total of 6270 images corresponding to the name of animal types.

All images are RGB images of 224 pixels wide by 224 pixels high in .jpg format. The images are separated in 151 folders according to their respective class.

The task is to categorize each animal into one of 151 categories.

We provide baseline code that includes the following features:

- Loading and Analysing the dataset using torchvision.
- Defining a simple convolutional neural network.
- How to use existing loss function for the model learning.
- Train the network on the training data.
- Test the trained network on the testing data.

The following changes could be considered: 1. "Transfer" Learning (ie use a model pre-trained another dataset) 2. Change of advanced training parameters: Learning Rate, Optimizer, Batch-size, Number of Max Epochs, and Drop-out. 3. Use of a new loss function. 4. Data augmentation 5. Architectural Changes: Batch Normalization, Residual layers, etc. 6. Others

Your code should be modified from the provided baseline. A pdf report of a maximum of two pages is required to explain the changes you made from the baseline, why you chose those changes, and the improvements they achieved.

Marking Rules: We will mark the competition based on the final test accuracy on testing images and your report.

Final mark (out of 50) = acc_mark + efficiency mark + report mark ###Acc_mark 10:

We will rank all the submission results based on their test accuracy. Zero improvement over the baseline yields 0 marks. Maximum improvement over the baseline will yield 10 marks. There will be a sliding scale applied in between.

###Efficiency mark 10:

Efficiency considers not only the accuracy, but the computational cost of running the model (flops: https://en.wikipedia.org/wiki/FLOPS). Efficiency for our purposes is defined to be the ratio of accuracy (in %) to Gflops. Please report the computational cost for your final model and include the efficiency calculation in your report. Maximum improvement over the baseline will yield 10 marks. Zero improvement over the baseline yields zero marks, with a sliding scale in between.

###Report mark 30: Your report should comprise: 1. An introduction showing your understanding of the task and of the baseline model: [10 marks]

2. A description of how you have modified aspects of the system to improve performance. [10 marks]

A recommended way to present a summary of this is via an "ablation study" table, eg:

Method1	Method2	Method3	Accuracy
N	N	N	60%
Y	N	N	65%
Y	Y	N	77%
Y	Y	Y	82%

3. Explanation of the methods for reducing the computational cost and/or improve the trade-off between accuracy and cost: [5 marks]

4. Limitations/Conclusions: [5 marks]

[1]:
```
##########################################################################################################
### Subject: Computer Vision
### Year: 2023
### Student Name: Lok Yiu Savannah Fung, Pragya Kaushik
### Student ID: a1844641, a1840097
### Comptetion Name: Animal Classification Competition
### Final Results:
### ACC: 92.81250238418579% FLOPs: 0.002964856103062629 7G
### --- THIS IS THE FINAL MODEL ---
##########################################################################################################
```

2 Setup

[2]:
```
# Importing libraries.
import os
import random
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from tqdm.notebook import tqdm

import warnings
```

```
warnings.filterwarnings('ignore')

from torchvision import datasets, transforms, models
from torchvision.datasets import ImageFolder
from torchvision.transforms import ToTensor
from torchvision.utils import make_grid
from torch.utils.data import random_split
from torch.utils.data.dataloader import DataLoader
import matplotlib.pyplot as plt
%matplotlib inline
```

[3]:
```
# Mounting G-Drive to get your dataset.
# To access Google Colab GPU; Go To: Edit >>> Netebook Settings >>> Hardware␣
 ↪Accelarator: Select GPU.
# Reference: https://towardsdatascience.com/
 ↪google-colab-import-and-export-datasets-eccf801e2971
# from google.colab import drive
# drive.mount('/content/drive')
```

[4]:
```
# import sys
# sys.path.append('/content/drive/MyDrive/cv-assignment4')

from FLOPs_counter import print_model_parm_flops
```

[5]:
```
# To check wether Google Colab GPU has been assigned/not.
def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available():
        return torch.device('cuda')
    else:
        return None

def to_device(data, device):
    """Move tensor(s) to chosen device"""
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
```

```
            yield to_device(b, self.device)

    def __len__(self):
        """Number of batches"""
        return len(self.dl)
```

[6]:
```
# Evaluate the model
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

def get_current_lr(optimizer):
    for param_group in optimizer.param_groups:
        return param_group['lr']

# Fit the model
def fit(epochs, lr, model, train_loader, val_loader, opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,␣
 ↪mode='min', factor=0.1, patience=1, verbose=True)
    for epoch in range(epochs):
        # Training Phase
        model.train()
        train_losses = []
        # Print the current learning rate
        current_lr = get_current_lr(optimizer)
        print(f'Epoch {epoch+1}/{epochs}, Learning Rate: {current_lr}')
        for batch in tqdm(train_loader):
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
        # Validation phase
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        model.epoch_end(epoch, result)
        history.append(result)
        # Step the scheduler
        scheduler.step(result['val_loss'])
    return history

# Plot the accuracies
def plot_accuracies(history):
```

```
    accuracies = [x['val_acc'] for x in history]
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs. No. of epochs')
    plt.show()

# Plot the losses
def plot_losses(history):
    train_losses = [x.get('train_loss') for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Validation'])
    plt.title('Loss vs. No. of epochs')
    plt.show()
```

[7]:
```
def accuracy(output, target, topk=(1,)):
    """
    Computes the accuracy over the k top predictions for the specified values␣
 ↪of k
    In top-3 accuracy you give yourself credit for having the right answer
    if the right answer appears in your top five guesses.
    """
    with torch.no_grad():
        maxk = 3
        batch_size = target.size(0)
        _, pred = output.topk(maxk, 1, True, True)
        pred = pred.t()
        correct = (pred == target.unsqueeze(dim=0)).expand_as(pred)
        correct_3 = correct[:3].reshape(-1).float().sum(0, keepdim=True)
        return correct_3.mul_(1.0 / batch_size)

class ImageClassificationBase(nn.Module):
    def training_step(self, batch):
        images, labels = batch
        out = self(images)                    # Generate predictions
        loss = F.cross_entropy(out, labels) # Calculate loss. Hints: the loss␣
 ↪function can be changed to improve the accuracy
        return loss

    def validation_step(self, batch):
        images, labels = batch
        out = self(images)                      # Generate predictions
        loss = F.cross_entropy(out, labels)   # Calculate loss
```

```
        acc = accuracy(out, labels, (5))           # Calculate accuracy
        return {'val_loss': loss.detach(), 'val_acc': acc}

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
        batch_accs = [x['val_acc'] for x in outputs]
        epoch_acc = torch.stack(batch_accs).mean()      # Combine accuracies
        return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

    def epoch_end(self, epoch, result):
        print("Epoch [{}], train_loss: {:.4f}, val_loss: {:.4f}, val_acc: {:.
↪4f}".format(
            epoch, result['train_loss'], result['val_loss'], result['val_acc']))
```

3 Load Data

[8]:
```
# Dataset path
# data_dir = '/content/drive/MyDrive/cv-assignment4/animal/dataset/dataset'
data_dir = 'animal/dataset/dataset/'
classes = os.listdir(data_dir)
```

[9]:
```
# Image Transformations
transform = transforms.Compose([
            transforms.Resize(112),
            transforms.RandomHorizontalFlip(),
            transforms.CenterCrop(112),
            transforms.ToTensor(),
            transforms.Normalize((0.488), (0.2172)),
        ])

# Load Data
dataset = ImageFolder(data_dir, transform=transform)

print('Size of training dataset :', len(dataset))
```

```
Size of training dataset : 6270
```

[10]:
```
# Splitting the dataset to training, validation, and testing category
torch.manual_seed(10)
val_size = len(dataset)//20
test_size = len(dataset)//10
train_size = len(dataset) - val_size - test_size

# Random Splitting
train_ds, val_ds, test_ds = random_split(dataset, [train_size, val_size,␣
↪test_size])
```

```
print('Size of training dataset:', len(train_ds))
print('Size of validation dataset:', len(val_ds))
print('Size of test dataset:', len(test_ds))
```

```
Size of training dataset: 5330
Size of validation dataset: 313
Size of test dataset: 627
```

4 Data Preprocessing

[11]:
```
# Image Transformations
train_transform = transforms.Compose([
    transforms.Resize(224),
    transforms.RandomHorizontalFlip(),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
])

train_ds.dataset.transform = train_transform
```

[12]:
```
# Data loaders
batch_size = 16
train_loader = DataLoader(train_ds, batch_size, shuffle=True, num_workers=2,␣
 ↪pin_memory=True)
val_loader = DataLoader(val_ds, batch_size, num_workers=2, pin_memory=True)
test_loader = DataLoader(test_ds, batch_size, num_workers=2, pin_memory=True)
```

[13]:
```
# Move data to CPU (for QAT)
device = None
train_loader = DeviceDataLoader(train_loader, device)
val_loader = DeviceDataLoader(val_loader, device)
test_loader = DeviceDataLoader(test_loader, device)
```

5 Model Initialization

[14]:
```
class QuantizableMobileNetV3Model(ImageClassificationBase):
    def __init__(self, num_classes, freeze_layers=True):
        super(QuantizableMobileNetV3Model, self).__init__()
        self.model = models.quantization.mobilenet_v3_large(pretrained=True,␣
 ↪quantize=False)
        if freeze_layers:
            for param in self.model.parameters():
                param.requires_grad = False
        for param in self.model.features[-8:].parameters():
```

```
            param.requires_grad = True
        num_ftrs = self.model.classifier[3].in_features
        self.model.classifier[3] = nn.Linear(num_ftrs, num_classes)

    def forward(self, x):
        x = self.model(x)
        return x
```

[15]:
```
num_classes = len(classes) # 151 classes

# Initialize the model
model = QuantizableMobileNetV3Model(num_classes, freeze_layers=True)
```

6 Model Training

[16]:
```
train_dl = DeviceDataLoader(train_loader, device)
val_dl = DeviceDataLoader(val_loader, device)
model = to_device(model, device)

# Evaluate the model before training
history = [evaluate(model, val_dl)]
history_not_trained = history
print("Before training: ", history)
```

```
Before training:  [{'val_loss': 5.087038040161133, 'val_acc':
0.02812499925494194}]
```

[17]:
```
# Prepare for QAT
model.train()
model.qconfig = torch.quantization.get_default_qat_qconfig('fbgemm')
torch.quantization.prepare_qat(model, inplace=True)
```

[17]:
```
QuantizableMobileNetV3Model(
  (model): QuantizableMobileNetV3(
    (features): Sequential(
      (0): Conv2dNormActivation(
        (0): Conv2d(
          3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False
          (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
            (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
          )
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
```

```
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
        (1): BatchNorm2d(
          16, eps=0.001, momentum=0.01, affine=True, track_running_stats=True
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
        (2): Hardswish(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (1): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=16, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
```

```
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              16, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              16, 16, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              16, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
```

```
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (2): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              16, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              64, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
```

```
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
groups=64, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              64, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(
              64, 24, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
```

```
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              24, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (3): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              24, 72, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
```

```
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              72, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              72, 72, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=72, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
```

```
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              72, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(
              72, 24, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              24, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (4): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              24, 72, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              72, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
              )
            )
            (2): ReLU()
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              72, 72, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2),
groups=72, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              72, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              72, 24, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
```

```
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (fc2): Conv2d(
              24, 72, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
```

```
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              72, 40, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              40, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
```

```
max_val=-inf)
          )
        )
      )
      (5): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              40, 120, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              120, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              120, 120, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2),
groups=120, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
```

```
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              120, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              120, 32, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
              )
            )
            (fc2): Conv2d(
              32, 120, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              120, 40, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
```

```
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              40, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (6): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              40, 120, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
```

```
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              120, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              120, 120, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2),
groups=120, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
```

```
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              120, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): ReLU()
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              120, 32, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (fc2): Conv2d(
              32, 120, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
```

```
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              120, 40, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
```

```
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              40, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (7): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
```

```
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              240, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              240, 240, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
groups=240, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
```

```
            )
            (1): BatchNorm2d(
              240, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(
              240, 80, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (8): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              80, 200, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              200, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              200, 200, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=200, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              200, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
```

```
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(
              200, 80, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (9): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              80, 184, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              184, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
            )
          )
          (2): Hardswish(
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(
            184, 184, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=184, bias=False
            (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
              (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
            )
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (1): BatchNorm2d(
            184, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (2): Hardswish(
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(
              184, 80, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
```

```
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (10): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              80, 184, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              184, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
```

```
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              184, 184, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=184, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              184, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
              )
            )
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(
              184, 80, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              80, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
```

```
      )
      (11): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
```

```
            (0): Conv2d(
              480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=480, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              480, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              480, 120, kernel_size=(1, 1), stride=(1, 1)
```

```
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (fc2): Conv2d(
              120, 480, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              480, 112, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              112, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
```

```
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (12): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              672, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
```

```
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              672, 672, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
groups=672, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              672, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
            )
          )
        )
        (2): QuantizableSqueezeExcitation(
          (avgpool): AdaptiveAvgPool2d(output_size=1)
          (fc1): Conv2d(
            672, 168, kernel_size=(1, 1), stride=(1, 1)
            (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
              (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
            )
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (fc2): Conv2d(
            168, 672, kernel_size=(1, 1), stride=(1, 1)
            (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
              (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
            )
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (activation): ReLU()
          (scale_activation): Hardsigmoid(
            (activation_post_process): FixedQParamsFakeQuantize(
              fake_quant_enabled=tensor([1], dtype=torch.uint8),
```

```
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              112, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
            )
          )
        )
      )
      (skip_add): FloatFunctional(
        (activation_post_process): FusedMovingAvgObsFakeQuantize(
          fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
          (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
        )
      )
    )
    (13): QuantizableInvertedResidual(
      (block): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(
            112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False
            (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
              (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
            )
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (1): BatchNorm2d(
            672, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
```

```
            )
          )
          (2): Hardswish(
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(
            672, 672, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2),
groups=672, bias=False
            (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
              (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
            )
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (1): BatchNorm2d(
            672, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
            (activation_post_process): FusedMovingAvgObsFakeQuantize(
              fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
              (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
            )
          )
          (2): Hardswish(
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              672, 168, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (fc2): Conv2d(
              168, 672, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
```

```
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              672, 160, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (14): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
```

```
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              960, 960, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2),
groups=960, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
```

```
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              960, 240, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (fc2): Conv2d(
              240, 960, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
```

```
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
```

```
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (15): QuantizableInvertedResidual(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(
              160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
```

```
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (1): Conv2dNormActivation(
            (0): Conv2d(
              960, 960, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2),
groups=960, bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
```

```
            )
            (1): BatchNorm2d(
              960, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (2): Hardswish(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (2): QuantizableSqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(
              960, 240, kernel_size=(1, 1), stride=(1, 1)
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (fc2): Conv2d(
              240, 960, kernel_size=(1, 1), stride=(1, 1)
```

```
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (activation): ReLU()
            (scale_activation): Hardsigmoid(
              (activation_post_process): FixedQParamsFakeQuantize(
                fake_quant_enabled=tensor([1], dtype=torch.uint8),
observer_enabled=tensor([1], dtype=torch.uint8), scale=tensor([0.0039]),
zero_point=tensor([0], dtype=torch.int32), dtype=torch.quint8, quant_min=0,
quant_max=255, qscheme=torch.per_tensor_affine
                (activation_post_process): FixedQParamsObserver()
              )
            )
            (skip_mul): FloatFunctional(
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
          (3): Conv2dNormActivation(
            (0): Conv2d(
              960, 160, kernel_size=(1, 1), stride=(1, 1), bias=False
              (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
                (activation_post_process):
```

```
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
              )
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
            (1): BatchNorm2d(
              160, eps=0.001, momentum=0.01, affine=True,
track_running_stats=True
              (activation_post_process): FusedMovingAvgObsFakeQuantize(
                fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
                (activation_post_process):
MovingAverageMinMaxObserver(min_val=inf, max_val=-inf)
              )
            )
          )
        )
        (skip_add): FloatFunctional(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
      (16): Conv2dNormActivation(
        (0): Conv2d(
          160, 960, kernel_size=(1, 1), stride=(1, 1), bias=False
          (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
            (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
          )
```

```
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
        (1): BatchNorm2d(
          960, eps=0.001, momentum=0.01, affine=True, track_running_stats=True
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
        (2): Hardswish(
          (activation_post_process): FusedMovingAvgObsFakeQuantize(
            fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
            (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
          )
        )
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=1)
    (classifier): Sequential(
      (0): Linear(
        in_features=960, out_features=1280, bias=True
        (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
          fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
          (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
        )
        (activation_post_process): FusedMovingAvgObsFakeQuantize(
          fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
```

```
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
          (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
        )
      )
      (1): Hardswish(
        (activation_post_process): FusedMovingAvgObsFakeQuantize(
          fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
          (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
        )
      )
      (2): Dropout(p=0.2, inplace=True)
      (3): Linear(
        in_features=1280, out_features=152, bias=True
        (weight_fake_quant): FusedMovingAvgObsFakeQuantize(
          fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.qint8, quant_min=-128, quant_max=127,
qscheme=torch.per_channel_symmetric, reduce_range=False
          (activation_post_process):
MovingAveragePerChannelMinMaxObserver(min_val=tensor([]), max_val=tensor([]))
        )
        (activation_post_process): FusedMovingAvgObsFakeQuantize(
          fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
          (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
        )
      )
    )
    (quant): QuantStub(
      (activation_post_process): FusedMovingAvgObsFakeQuantize(
        fake_quant_enabled=tensor([1]), observer_enabled=tensor([1]),
scale=tensor([1.]), zero_point=tensor([0], dtype=torch.int32),
dtype=torch.quint8, quant_min=0, quant_max=127, qscheme=torch.per_tensor_affine,
reduce_range=True
        (activation_post_process): MovingAverageMinMaxObserver(min_val=inf,
max_val=-inf)
      )
    )
```

```
    (dequant): DeQuantStub()
  )
)
```

[18]:
```
num_epochs = 10
opt_func = torch.optim.Adam
lr = 0.001

# Train the model
history += fit(num_epochs, lr, model, train_dl, val_dl, opt_func)
```

```
Epoch 1/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [0], train_loss: 2.7840, val_loss: 1.9499, val_acc: 0.7639
Epoch 2/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [1], train_loss: 1.1562, val_loss: 1.4118, val_acc: 0.8288
Epoch 3/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [2], train_loss: 0.7086, val_loss: 1.4011, val_acc: 0.8413
Epoch 4/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [3], train_loss: 0.4988, val_loss: 1.5409, val_acc: 0.8608
Epoch 5/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [4], train_loss: 0.4220, val_loss: 1.3071, val_acc: 0.8569
Epoch 6/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [5], train_loss: 0.3465, val_loss: 1.3940, val_acc: 0.8500
Epoch 7/10, Learning Rate: 0.001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [6], train_loss: 0.2750, val_loss: 1.3142, val_acc: 0.8545
Epoch 8/10, Learning Rate: 0.0001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [7], train_loss: 0.1540, val_loss: 0.8367, val_acc: 0.8969
Epoch 9/10, Learning Rate: 0.0001
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [8], train_loss: 0.0839, val_loss: 0.7505, val_acc: 0.9312
Epoch 10/10, Learning Rate: 0.0001
```

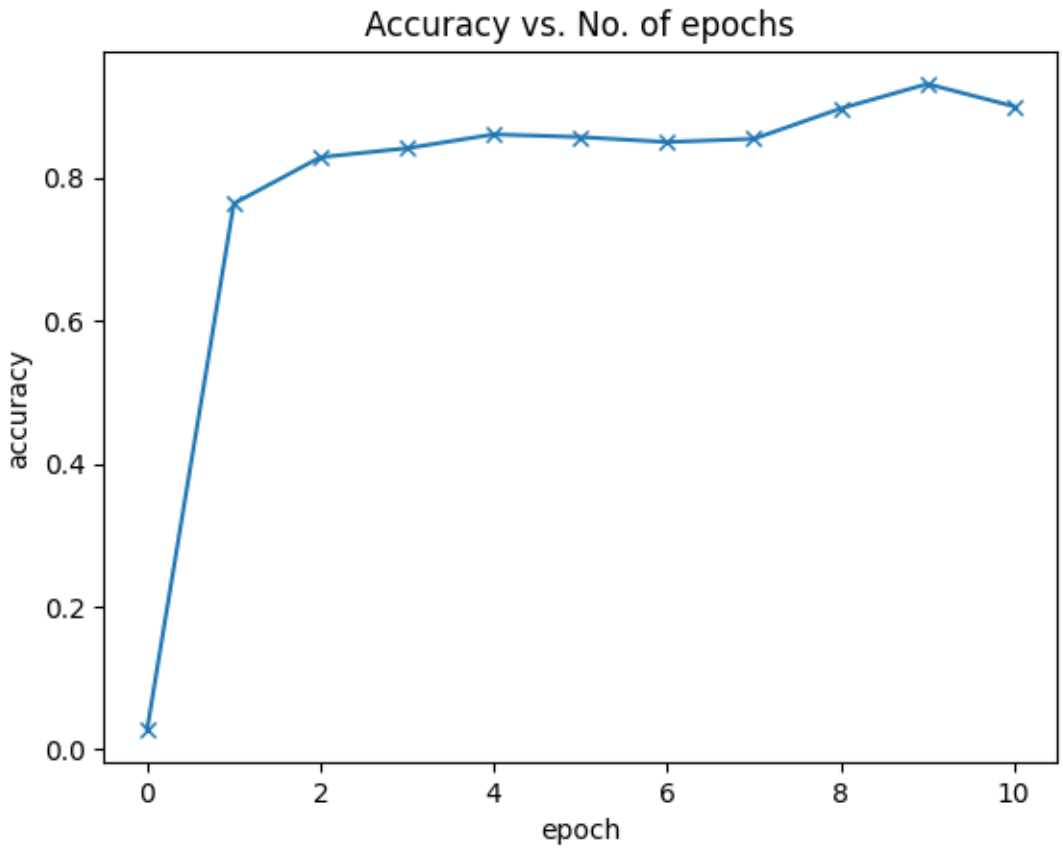
```
  0%|          | 0/334 [00:00<?, ?it/s]
Epoch [9], train_loss: 0.0618, val_loss: 0.8773, val_acc: 0.9000
```

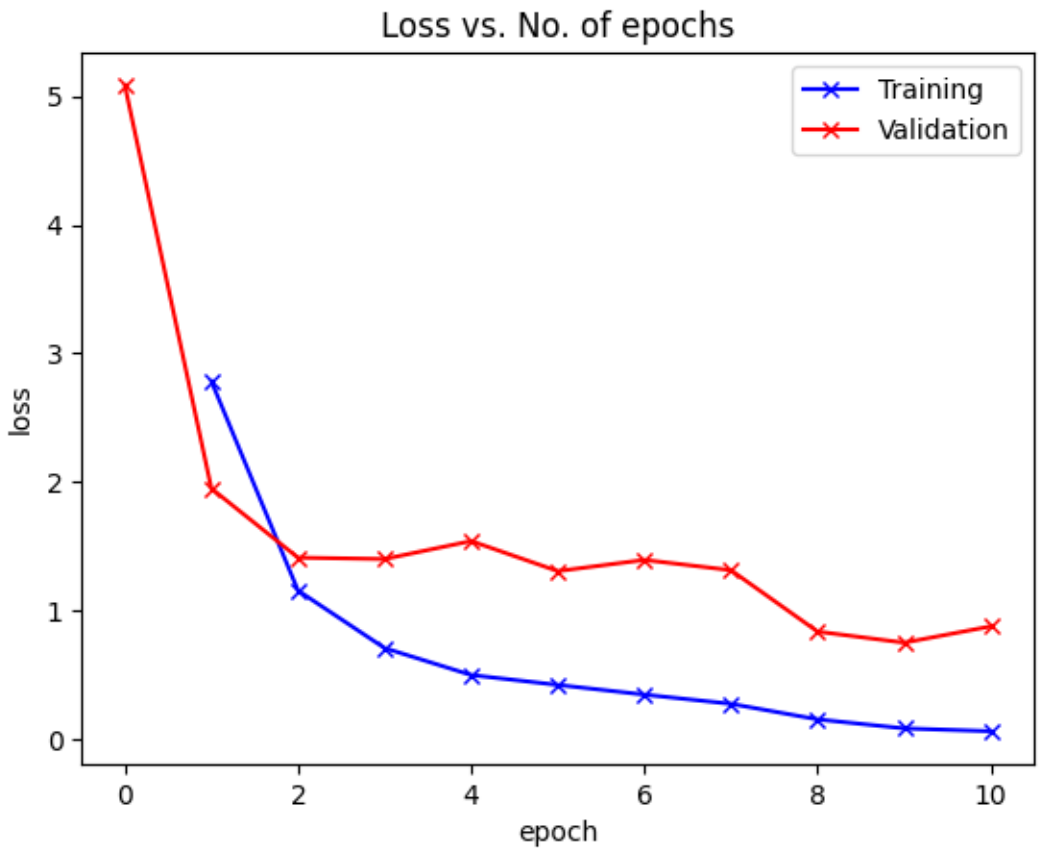
[19]:
```
# Set the quantization backend
torch.backends.quantized.engine = 'qnnpack'

# Convert to quantized model for inference
model.eval()
model.to('cpu')
model_int8 = torch.quantization.convert(model, inplace=False)
```

7 Model Evaluation

[20]:
```
# Plot the accuracies and losses
plot_accuracies(history)
plot_losses(history)
```





[21]:
```
# Evaluate the model after training
val = evaluate(model_int8, test_loader)
print("Accuracy:", val['val_acc'])
```

```
Accuracy: 0.9281250238418579
```

[22]:
```
# Number of FLOPs
input = torch.randn(1, 3, 224, 224)
flops = print_model_parm_flops(model_int8, input, detail=False)
print("Number of FLOPs:", flops)
```

```
 + Number of FLOPs: 0.00G
Number of FLOPs: 0.0029648561030626297
```