# DL-Ops

Lab Assignment 6- Report

Debonil Ghosh
Roll No: M21AIE225
Executive MTech
Artificial Intelligence
Indian Institute of Technology, Jodhpur

## Problem Statement:

Q1. Based on the lecture by Dr. Anush on DLOPs, you have to perform the following experiments:

a. Load and pre-processing CIFAR100 dataset using standard augmentation and normalization techniques [10 Marks]
b. Train the following models for 50 epoch and at the same time profile the model using Tensorboard during the training step [5*4 = 20 Marks]
    a. ○ ResNet-34
    b. ○ DenseNet-121
    c. ○ EfficientNet-B0
    d. ○ ConvNeXt-T
c. Then perform the following model inferencing techniques on the above listed models
   [10*2 = 20 Marks]
    a. ○ ONNX ; ONNX Quantized
    b. ○ Torchscript
   Report the model size and average execution time before and after performing the
   above mentioned inferencing techniques on the test dataset [5*4 = 20 Marks]

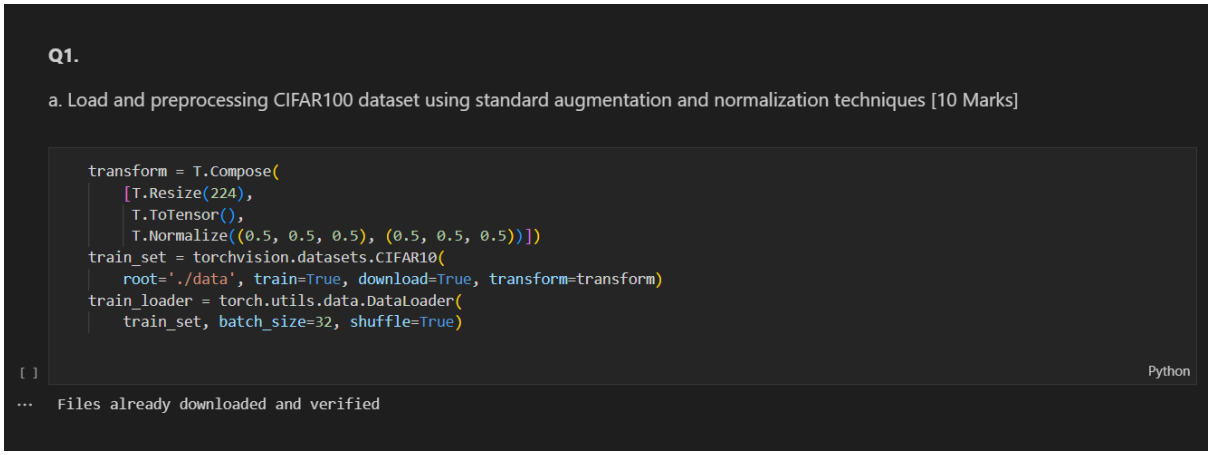
d. Analyze the models based on their architecture and inferencing techniques and write them in the report.

## Solution Google Colab Link:

https://drive.google.com/file/d/1d0-6YpeKdkFHGy0RQlR1IyBbbjlVTgnx/view?usp=sharing

## Q1.

### a. Load and pre-processing CIFAR100 dataset using standard augmentation and normalization techniques [10 Marks]



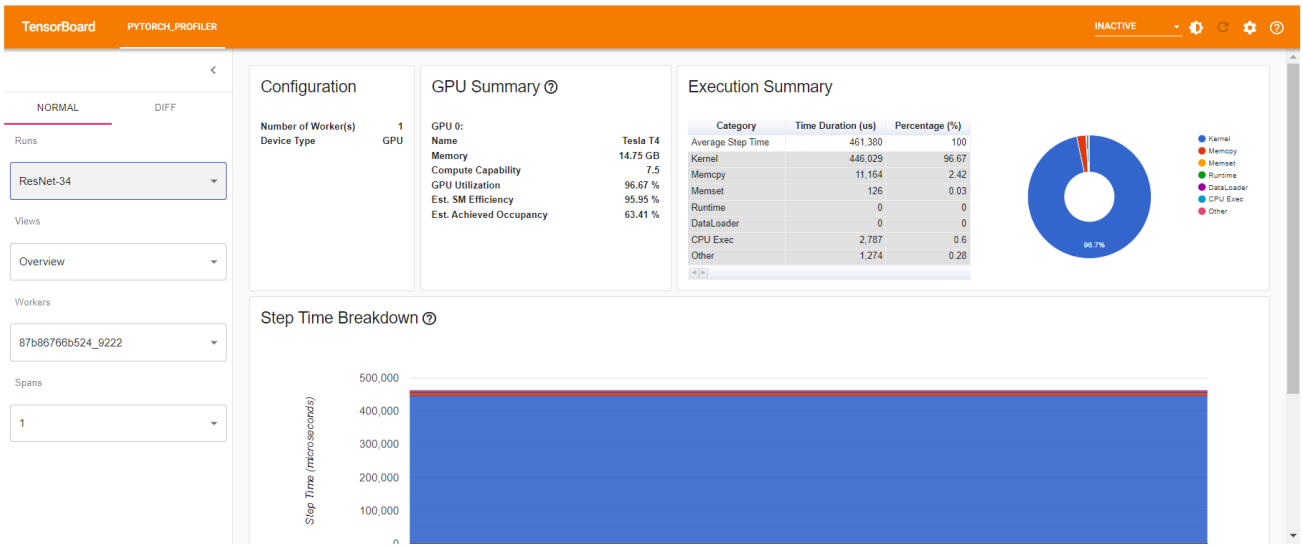

### b. Train the following models for 50 epoch and at the same time profile the model using Tensorboard during the training step [5*4 = 20 Marks]

- ResNet-34
- DenseNet-121
- EfficientNet-B0
- ConvNeXt-T

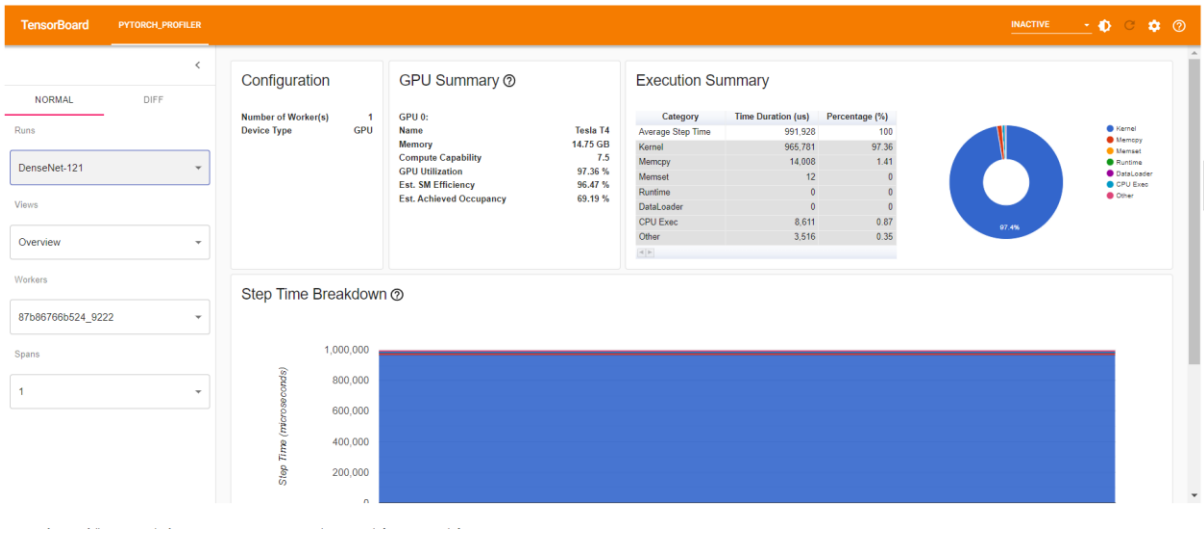
As asked in the question given models have been trained and profiled using the Tensor Board during the training process. Output from the Tensorboard is as below:

## Overall View:

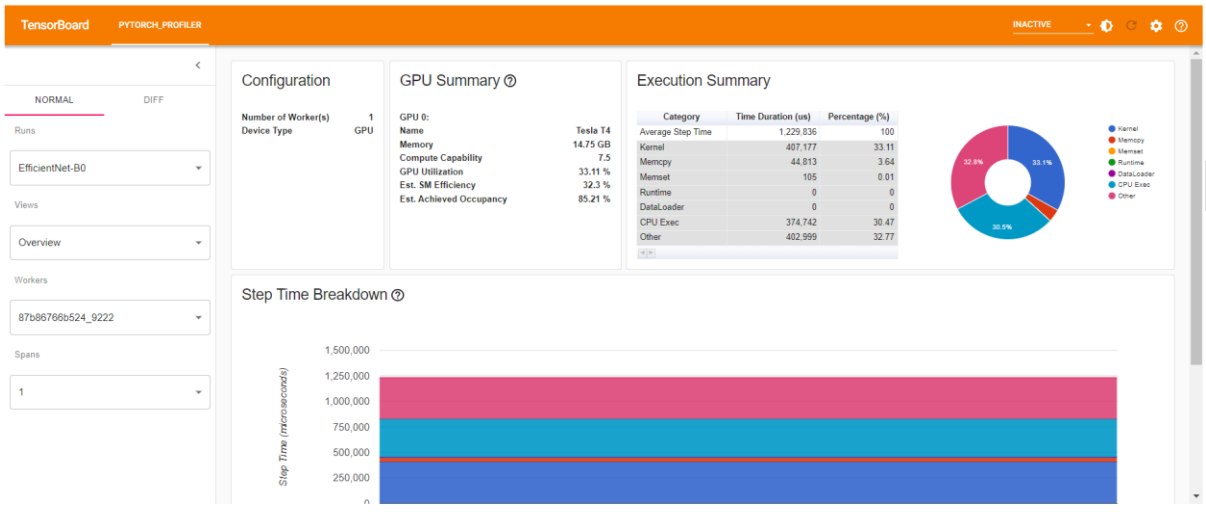
### ResNet – 34



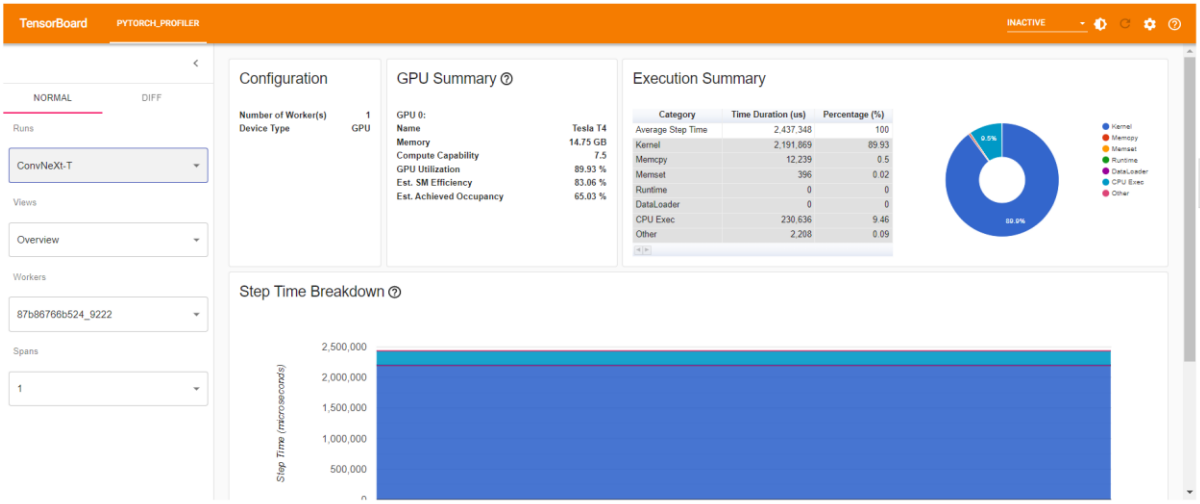

## DenseNet-121




## EfficientNet-B0

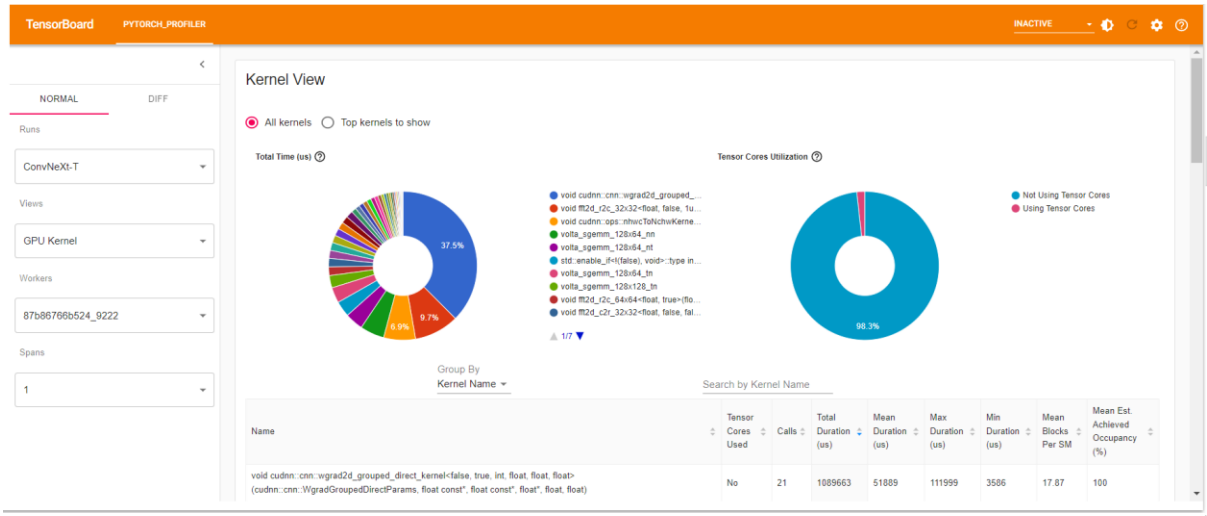


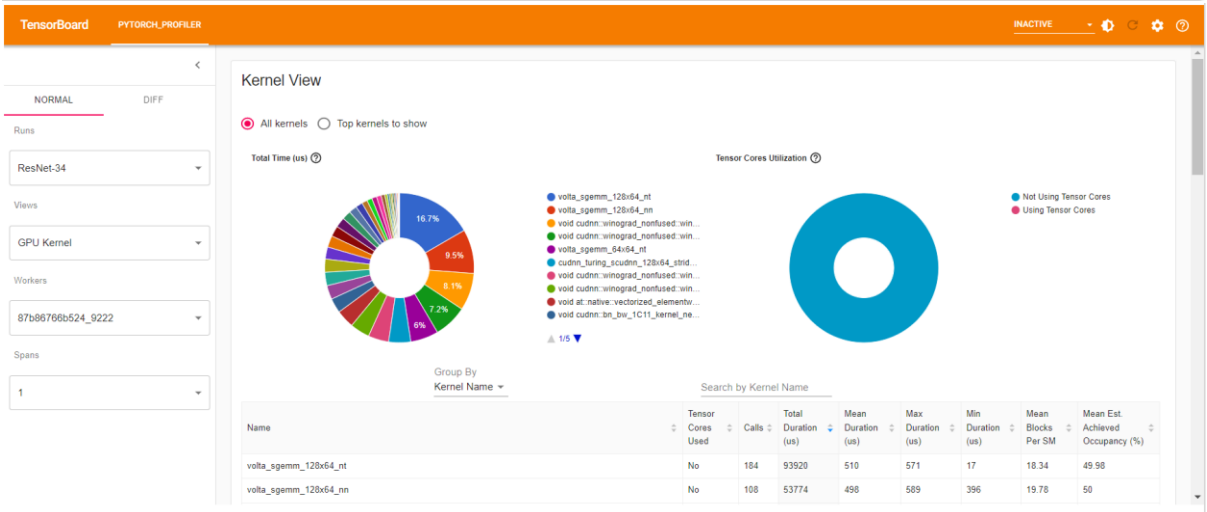
## ConvNeXt-T

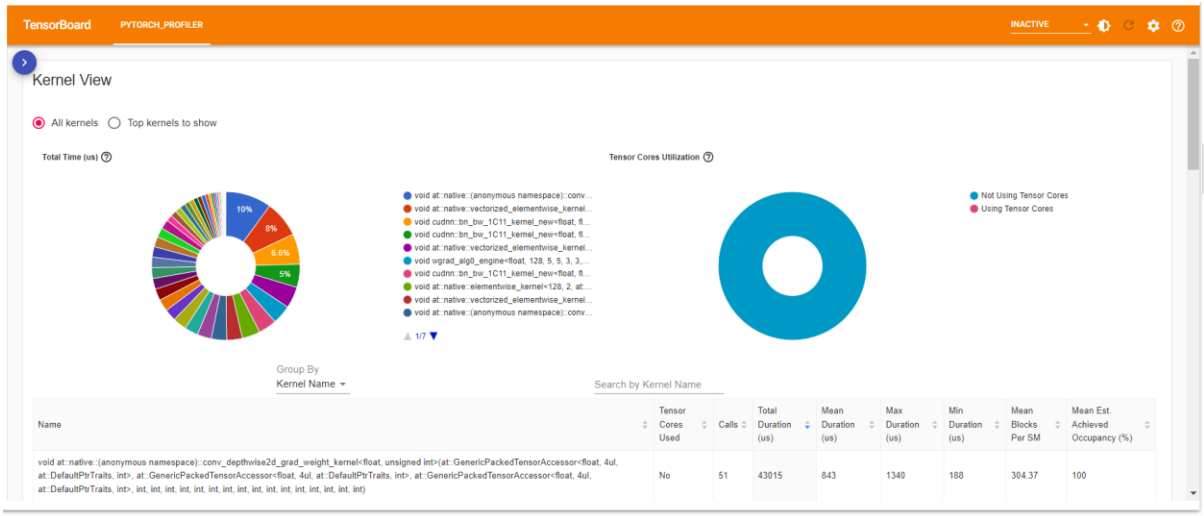


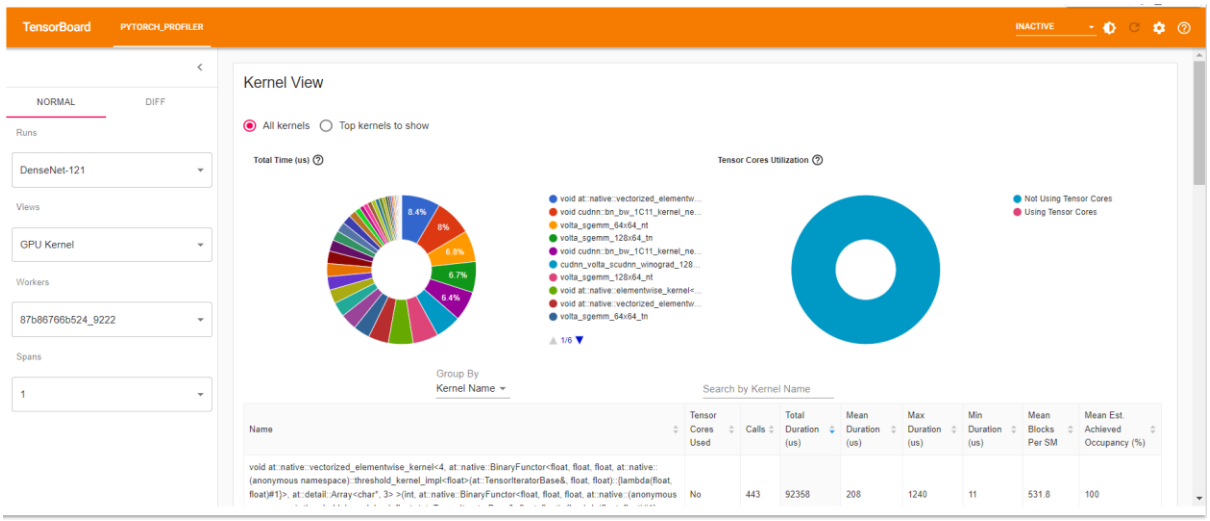

# GPU View:

## ConvNeXt-T




TensorBoard
PYTORCH_PROFILER
INACTIVE
Kernel View
All kernels
Top kernels to show
Total Time (us)
Tensor Cores Utilization
Not Using Tensor Cores
Using Tensor Cores
Group By
Kernel Name
Search by Kernel Name
Name
Tensor Cores Used
Calls
Total Duration (us)
Mean Duration (us)
Max Duration (us)
Min Duration (us)
Mean Blocks Per SM
Mean Est. Achieved Occupancy (%)
No
51
43015
843
1340
188
304.37
100


TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
DenseNet-121
Views
GPU Kernel
Workers
87b86766b524_9222
Spans
1
Kernel View
All kernels
Top kernels to show
Total Time (us)
Tensor Cores Utilization
Not Using Tensor Cores
Using Tensor Cores
Group By
Kernel Name
Search by Kernel Name
Name
Tensor Cores Used
Calls
Total Duration (us)
Mean Duration (us)
Max Duration (us)
Min Duration (us)
Mean Blocks Per SM
Mean Est. Achieved Occupancy (%)
No
443
92358
208
1240
11
531.8
100

Memory View:


TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
EfficientNet-B0
Views
Memory
Workers
87b86766b524_9222
Spans
1
Memory View
Device
GPU0
Peak Memory Usage: 3291.9MB
Memory Usage (MB)
Time (ms)
Allocated (MB)
Reserved (MB)
Search by Name
Min Size(KB)
45568
Max Size(KB)
182272
Operator
Size (KB)
Allocation Time (ms)
Release Time (ms)
Duration (ms)
aten::cudnn_convolution
131072
229.68
229.75
0.07
aten::cudnn_convolution
131072
230.08
230.13
0.05

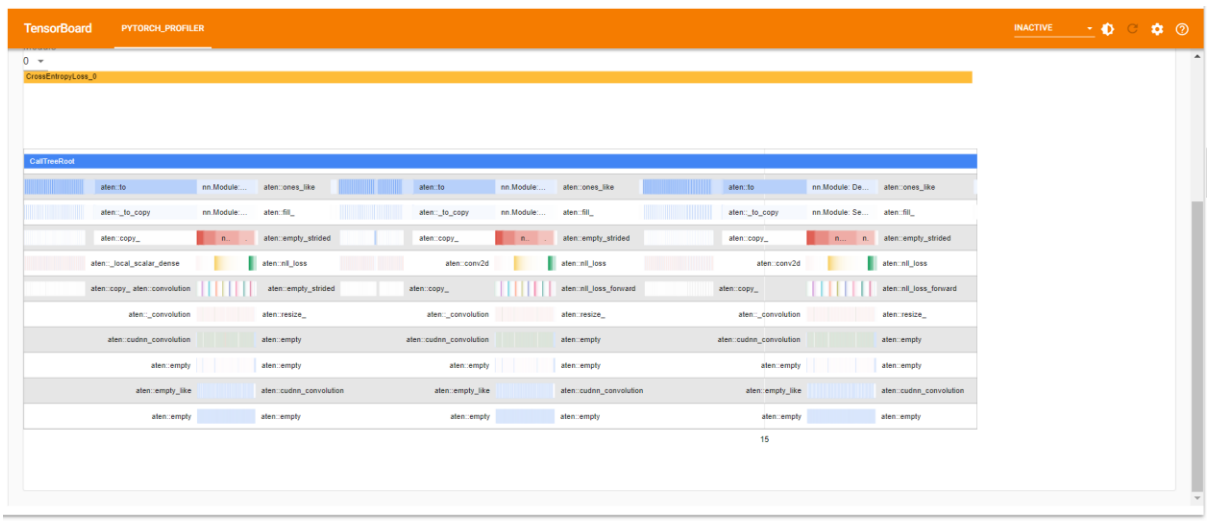

TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
DenseNet-121
Views
Memory
Workers
87b86766b524_9222
Spans
1
Memory View
Device
GPU0
Peak Memory Usage: 4515.8MB
Memory Usage (MB)
Time (ms)
Allocated (MB)
Reserved (MB)
Search by Name
Min Size(KB)
49152
Max Size(KB)
196608
Operator
Size (KB)
Allocation Time (ms)
Release Time (ms)
Duration (ms)
aten::cudnn_convolution
131072
183.59
183.65
0.06
aten::cudnn_convolution
141696
184.06
184.16
0.1

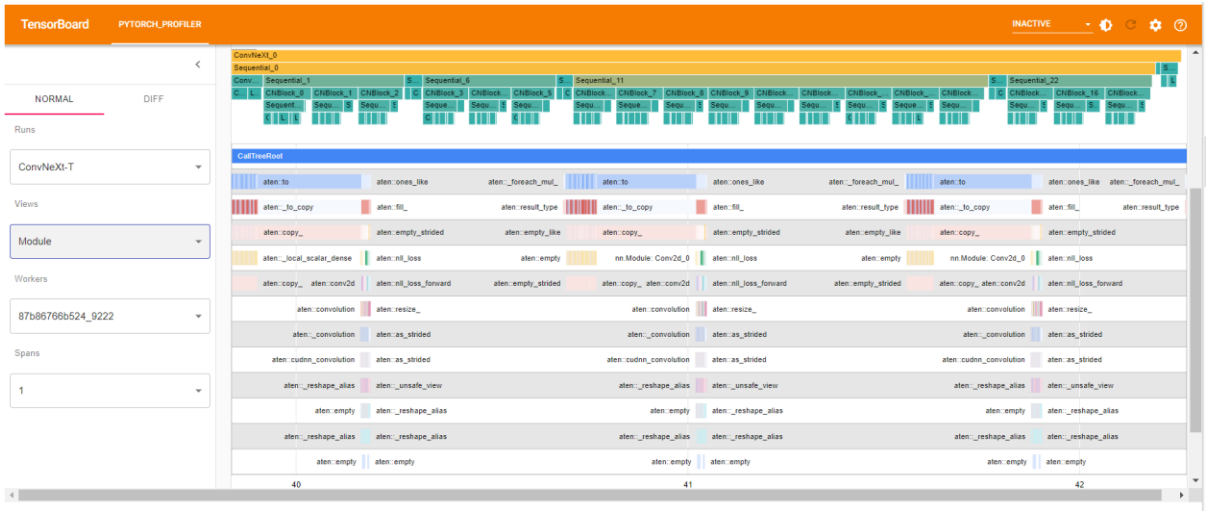

TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
ConvNeXt-T
Views
Memory
Workers
87b86766b524_9222
Spans
1
Memory View
Device
GPU0
Peak Memory Usage: 3978.1MB
Memory Usage (MB)
Time (ms)
Allocated (MB)
Reserved (MB)
Search by Name
Min Size(KB)
37888
Max Size(KB)
151552
Operator
Size (KB)
Allocation Time (ms)
Release Time (ms)
Duration (ms)
aten::gelu
75264
340.59
775.14
434.55
aten::mm
75264
772.93
775.34
2.41

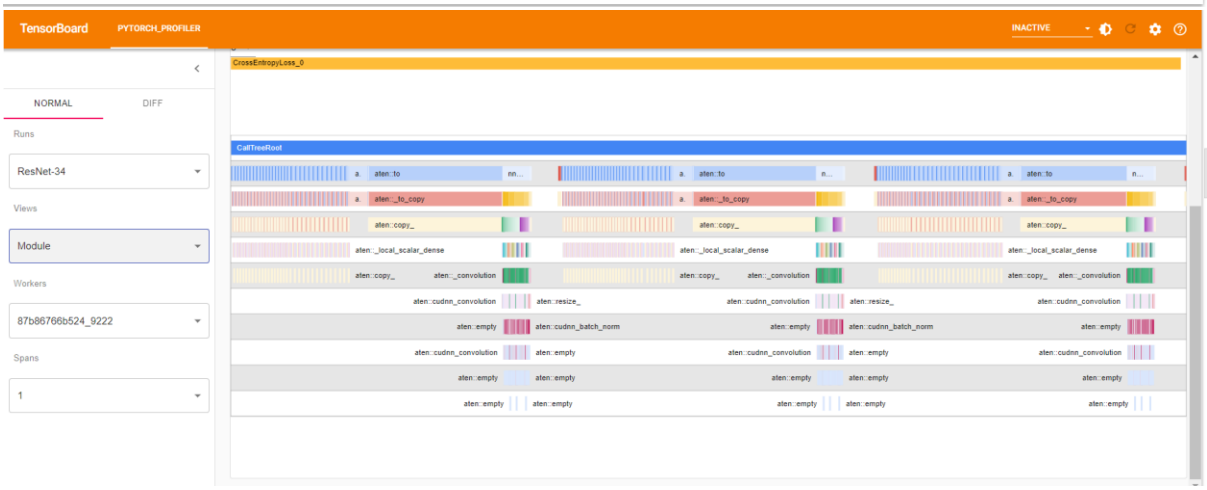

TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
ResNet-34
Views
Memory
Workers
87b86766b524_9222
Spans
1
Memory View
Device
GPU0
Peak Memory Usage: 1471.4MB
Memory Usage (MB)
Time (ms)
Allocated (MB)
Reserved (MB)
Search by Name
Min Size(KB)
28368
Max Size(KB)
113472
Operator
Size (KB)
Allocation Time (ms)
Release Time (ms)
Duration (ms)
aten::cudnn_convolution
113472
133.42
133.51
0.1
aten::cudnn_convolution
113472
133.85
133.92
0.07

# Module View:

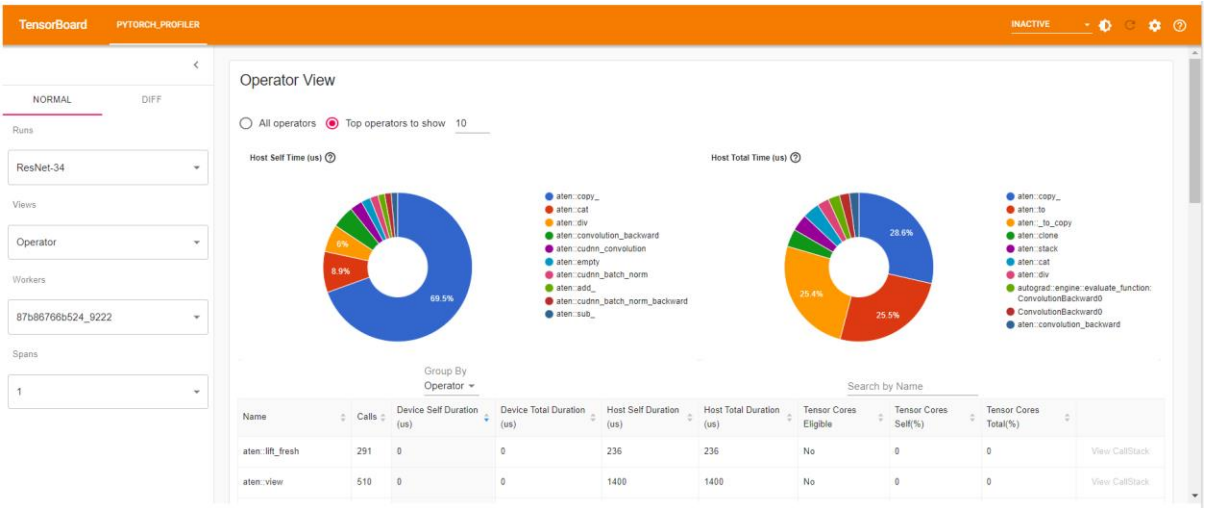

TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
EfficientNet-B0
Views
Module
Workers
87b86766b524_9222
Spans
1
CallTreeRoot
26
27

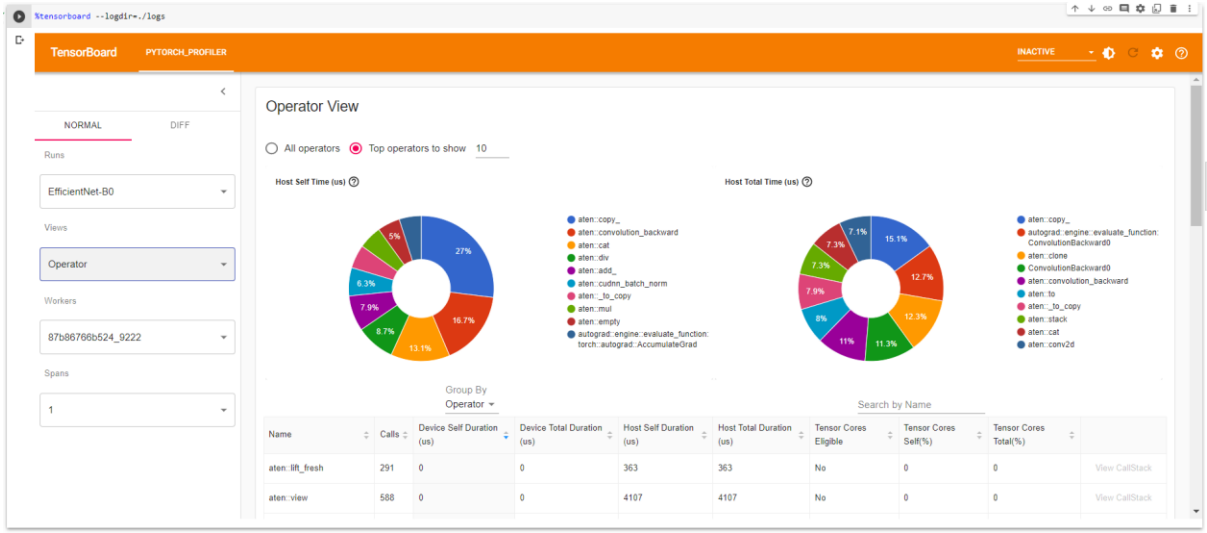

TensorBoard
PYTORCH_PROFILER
INACTIVE
CrossEntropyLoss_0
CallTreeRoot
15

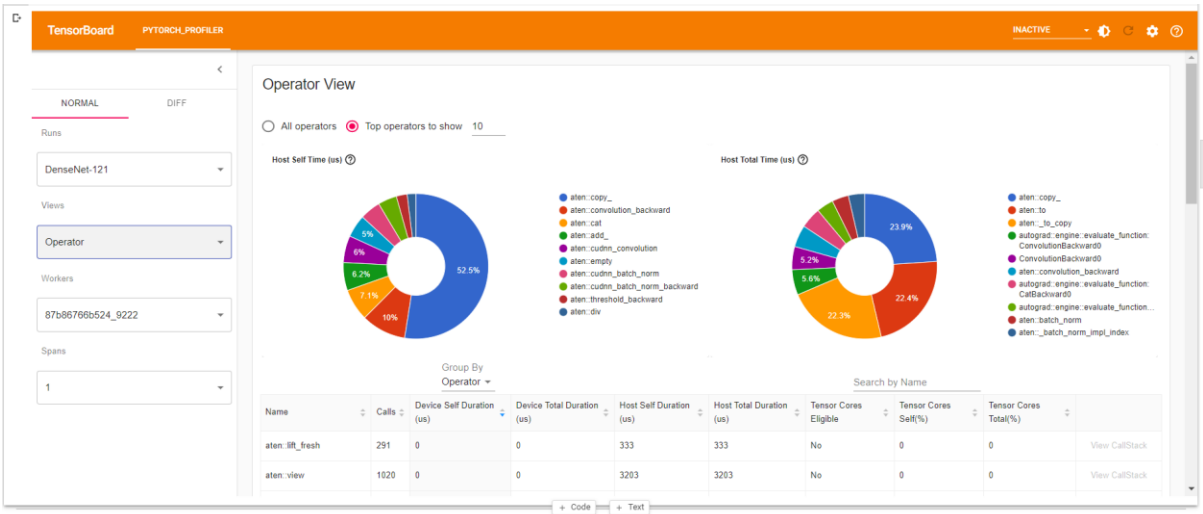

TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
ConvNeXt-T
Views
Module
Workers
87b86766b524_9222
Spans
1
ConvNeXt_0
Sequential_0
CallTreeRoot
40
41
42

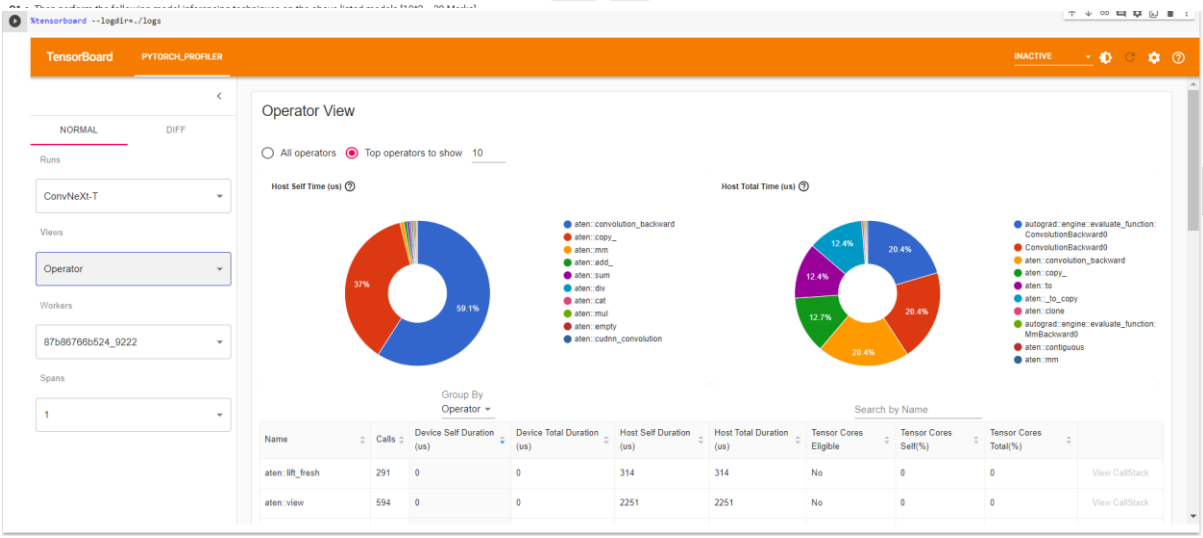

TensorBoard
PYTORCH_PROFILER
INACTIVE
NORMAL
DIFF
Runs
ResNet-34
Views
Module
Workers
87b86766b524_9222
Spans
1
CrossEntropyLoss_0
CallTreeRoot

# Operator View:






## C. Then perform the following model inferencing techniques on the above listed models [10*2 = 20 Marks]

# Torchscript:

| | Model | model_size | model_scripted_size | frozen_mod_size | avgRuntimePytorch | avgRuntimeTorchScript | avgRuntimeTorchScriptFrozen | unscripted_top5 | scripted_top5 | frozen_scripted_top5 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ResNet-34 | 83.286778 | 83.328921 | 83.142973 | 57.163395 | 136.099796 | 127.904972 | [640, 741, 539, 735, 458] | [640, 741, 539, 735, 458] | [640, 741, 539, 735, 458] |
| 1 | DenseNet-121 | 31.022118 | 31.195852 | 30.936444 | 22.560215 | 66.150334 | 2976.462460 | [490, 455, 539, 650, 446] | [490, 455, 539, 650, 446] | [490, 455, 539, 650, 446] |
| 2 | EfficientNet-B0 | 20.463761 | 20.659024 | 20.144644 | 15.952289 | 514.606910 | 353.644948 | [92, 127, 858, 22, 21] | [92, 127, 858, 22, 21] | [92, 127, 858, 22, 21] |
| 3 | ConvNeXt-T | 109.123593 | 109.177211 | 109.112747 | 10.349689 | 1371.275551 | 2310.848496 | [6, 3, 5, 4, 2] | [6, 3, 5, 4, 2] | [6, 3, 5, 4, 2] |

ResNet-34 Python model top 5 results: [640 741 539 735 458]
ResNet-34 TorchScript model top 5 results: [640 741 539 735 458]
ResNet-34 TorchScript Frozen model top 5 results: [640 741 539 735 458]

ResNet-34 Average runtime of Pytorch Model in cuda:0: 57.16339499995229
ResNet-34 Average runtime of TorchScript Model in cuda:0 : 136.0997959000997
ResNet-34 Average runtime of TorchScript Frozen Model in cuda:0 : 127.90497169999071

ResNet-34 Size of Pytorch Model in cuda:0: 83.2867784500122
ResNet-34 Size of TorchScript Model in cuda:0 : 83.3289213180542
ResNet-34 Size of TorchScript Frozen Model in cuda:0 : 83.14297294616699

DenseNet-121 Python model top 5 results: [490 455 539 650 446]
DenseNet-121 TorchScript model top 5 results: [490 455 539 650 446]
DenseNet-121 TorchScript Frozen model top 5 results: [490 455 539 650 446]

DenseNet-121 Average runtime of Pytorch Model in cuda:0: 22.56021480002346
DenseNet-121 Average runtime of TorchScript Model in cuda:0 : 66.15033400007633
DenseNet-121 Average runtime of TorchScript Frozen Model in cuda:0 : 2976.4624595999067

DenseNet-121 Size of Pytorch Model in cuda:0: 31.022117614746094
DenseNet-121 Size of TorchScript Model in cuda:0 : 31.195852279663086
DenseNet-121 Size of TorchScript Frozen Model in cuda:0 : 30.93644428253174

EfficientNet-B0 Python model top 5 results: [ 92 127 858 22 21]
EfficientNet-B0 TorchScript model top 5 results: [ 92 127 858 22 21]
EfficientNet-B0 TorchScript Frozen model top 5 results: [ 92 127 858 22 21]

EfficientNet-B0 Average runtime of Pytorch Model in cuda:0: 15.952289400001973
EfficientNet-B0 Average runtime of TorchScript Model in cuda:0 : 514.6069099000215
EfficientNet-B0 Average runtime of TorchScript Frozen Model in cuda:0 : 353.6449475999234

EfficientNet-B0 Size of Pytorch Model in cuda:0: 20.46376132965088
EfficientNet-B0 Size of TorchScript Model in cuda:0 : 20.659024238586426
EfficientNet-B0 Size of TorchScript Frozen Model in cuda:0 : 20.144643783569336

ConvNeXt-T Python model top 5 results: [6 3 5 4 2]
ConvNeXt-T TorchScript model top 5 results: [6 3 5 4 2]
ConvNeXt-T TorchScript Frozen model top 5 results: [6 3 5 4 2]

ConvNeXt-T Average runtime of Pytorch Model in cuda:0: 10.349689000031503
ConvNeXt-T Average runtime of TorchScript Model in cuda:0 : 1371.2755511000069
ConvNeXt-T Average runtime of TorchScript Frozen Model in cuda:0 : 2310.8484957000655

ConvNeXt-T Size of Pytorch Model in cuda:0: 109.1235933303833
ConvNeXt-T Size of TorchScript Model in cuda:0 : 109.1772108078003
ConvNeXt-T Size of TorchScript Frozen Model in cuda:0 : 109.11274719238281

# ONNX:

**********ResNet-34 **********

============= Diagnostic Run torch.onnx.export version 2.0.0+cu118 =============
verbose: False, log level: Level.ERROR
========== 0 NONE 0 NOTE 0 WARNING 0 ERROR ==========

ResNet-34 Average runtime of ONNX Model in GPU: 101.58939989996725
ResNet-34 Average runtime of ONNX Optimized Model in GPU: 99.03199939997194
quantized model saved to:ResNet-34_opt_quant.onnx
ResNet-34 ONNX full precision model size (MB): 83.13565731048584
ResNet-34 ONNX quantized model size (MB): 20.88004970550537
ResNet-34 Average runtime of ONNX Model in TPU: 147.2574743000223
ResNet-34 Average runtime of ONNX Quantized Model in TPU: 176.98937220000062

---

**********DenseNet-121 **********

============= Diagnostic Run torch.onnx.export version 2.0.0+cu118 =============
verbose: False, log level: Level.ERROR
========== 0 NONE 0 NOTE 0 WARNING 0 ERROR ==========

DenseNet-121 Average runtime of ONNX Model in GPU: 85.70039960000031
DenseNet-121 Average runtime of ONNX Optimized Model in GPU: 83.50030320002588
quantized model saved to:DenseNet-121_opt_quant.onnx
DenseNet-121 ONNX full precision model size (MB): 30.811330795288086
DenseNet-121 ONNX quantized model size (MB): 8.444440841674805
DenseNet-121 Average runtime of ONNX Model in TPU: 82.10658769994552
DenseNet-121 Average runtime of ONNX Quantized Model in TPU: 114.83463389999997

---

**********EfficientNet-B0 **********

============= Diagnostic Run torch.onnx.export version 2.0.0+cu118 =============
verbose: False, log level: Level.ERROR
========== 0 NONE 0 NOTE 0 WARNING 0 ERROR ==========

EfficientNet-B0 Average runtime of ONNX Model in GPU: 39.56867629997305

EfficientNet-B0 Average runtime of ONNX Optimized Model in GPU: 40.10078390003855
quantized model saved to:EfficientNet-B0_opt_quant.onnx
EfficientNet-B0 ONNX full precision model size (MB): 20.164688110351562
EfficientNet-B0 ONNX quantized model size (MB): 5.379528999328613
EfficientNet-B0 Average runtime of ONNX Model in TPU: 26.76917539997703
EfficientNet-B0 Average runtime of ONNX Quantized Model in TPU: 59.04552249996868

---

**********ConvNeXt-T **********

============= Diagnostic Run torch.onnx.export version 2.0.0+cu118 =============
verbose: False, log level: Level.ERROR
========== 0 NONE 0 NOTE 0 WARNING 0 ERROR ==========

ConvNeXt-T Average runtime of ONNX Model in GPU: 163.29668280000078
ConvNeXt-T Average runtime of ONNX Optimized Model in GPU: 167.82447880004838
quantized model saved to:ConvNeXt-T_opt_quant.onnx
ConvNeXt-T ONNX full precision model size (MB): 109.16751003265381
ConvNeXt-T ONNX quantized model size (MB): 27.682156562805176
ConvNeXt-T Average runtime of ONNX Model in TPU: 166.7791882000074
ConvNeXt-T Average runtime of ONNX Quantized Model in TPU: 221.07910109996283

---

## d. Analyze the models based on their architecture and inferencing techniques and write them in the report.

**ResNet-34:**

ResNet-34 is a deep neural network architecture that was introduced in 2015. It is a variant of the Residual Network (ResNet) architecture, which is designed to overcome the degradation problem in deep neural networks. The degradation problem occurs when the performance of a network saturates and then degrades as the depth of the network increases. ResNet-34 uses skip connections to allow the gradients to flow directly from one layer to another, which helps to mitigate the degradation problem.

ResNet-34 has 34 layers, including convolutional layers, pooling layers, and fully connected layers. The network consists of a series of residual blocks, where each block contains two or three convolutional layers. The input to each residual block is passed through a shortcut connection, which

adds the original input to the output of the block. This allows the network to learn residual functions, which are easier to optimize than the original functions.

In terms of inferencing techniques, ResNet-34 can be used for both classification and object detection tasks. It can be trained using supervised learning, and the weights can be fine-tuned using transfer learning.

**Architecture:**

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (2): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
  )
)
(layer2): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (2): BasicBlock(
    (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (3): BasicBlock(
    (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer3): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (2): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
```

```
  (3): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (4): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (5): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer4): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
```

```
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (2): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
 )
 (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
 (fc): Linear(in_features=512, out_features=1000, bias=True)
)
```

**DenseNet-121:**

DenseNet-121 is a deep neural network architecture that was introduced in 2016. It is a variant of the Dense Convolutional Network (DenseNet) architecture, which is designed to improve the flow of information through a network. DenseNet-121 uses dense connections, which connect every layer to every other layer in a feedforward fashion. This allows the network to reuse features learned by earlier layers, which helps to improve the efficiency and accuracy of the network.

DenseNet-121 has 121 layers, including convolutional layers, pooling layers, and fully connected layers. The network consists of a series of dense blocks, where each block contains several convolutional layers. The output of each dense block is concatenated with the input to the block, which allows the network to reuse features learned by earlier blocks.

In terms of inferencing techniques, DenseNet-121 can be used for both classification and object detection tasks. It can be trained using supervised learning, and the weights can be fine-tuned using transfer learning.

Architecture:

```
DenseNet(
  (features): Sequential(
    (conv0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (norm0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu0): ReLU(inplace=True)
    (pool0): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
    (denseblock1): _DenseBlock(
      (denselayer1): _DenseLayer(
        (norm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer2): _DenseLayer(
        (norm1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(96, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer3): _DenseLayer(
        (norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer4): _DenseLayer(

(norm1): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(160, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer5): _DenseLayer(

(norm1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer6): _DenseLayer(

(norm1): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(224, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

```
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
  )
  (transition1): _Transition(
    (norm): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
  )
  (denseblock2): _DenseBlock(
    (denselayer1): _DenseLayer(
      (norm1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
    (denselayer2): _DenseLayer(
      (norm1): BatchNorm2d(160, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(160, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
    (denselayer3): _DenseLayer(
      (norm1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

(relu1): ReLU(inplace=True)

(conv1): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer4): _DenseLayer(

(norm1): BatchNorm2d(224, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(224, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer5): _DenseLayer(

(norm1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer6): _DenseLayer(

(norm1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(288, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer7): _DenseLayer(

(norm1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(320, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer8): _DenseLayer(

(norm1): BatchNorm2d(352, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(352, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer9): _DenseLayer(

(norm1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer10): _DenseLayer(

(norm1): BatchNorm2d(416, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(416, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer11): _DenseLayer(

(norm1): BatchNorm2d(448, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(448, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer12): _DenseLayer(

(norm1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(480, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

)

```
  (transition2): _Transition(
    (norm): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
  )
  (denseblock3): _DenseBlock(
    (denselayer1): _DenseLayer(
      (norm1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
    (denselayer2): _DenseLayer(
      (norm1): BatchNorm2d(288, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(288, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
    (denselayer3): _DenseLayer(
      (norm1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(320, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer4): _DenseLayer(

(norm1): BatchNorm2d(352, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(352, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer5): _DenseLayer(

(norm1): BatchNorm2d(384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer6): _DenseLayer(

(norm1): BatchNorm2d(416, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(416, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer7): _DenseLayer(

(norm1): BatchNorm2d(448, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(448, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer8): _DenseLayer(

(norm1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(480, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer9): _DenseLayer(

(norm1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer10): _DenseLayer(

(norm1): BatchNorm2d(544, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(544, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer11): _DenseLayer(

(norm1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(576, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer12): _DenseLayer(

(norm1): BatchNorm2d(608, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(608, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer13): _DenseLayer(

(norm1): BatchNorm2d(640, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(640, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer14): _DenseLayer(

(norm1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(672, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer15): _DenseLayer(

(norm1): BatchNorm2d(704, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(704, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer16): _DenseLayer(

(norm1): BatchNorm2d(736, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(736, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer17): _DenseLayer(

(norm1): BatchNorm2d(768, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer18): _DenseLayer(

(norm1): BatchNorm2d(800, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(800, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer19): _DenseLayer(

(norm1): BatchNorm2d(832, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(832, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer20): _DenseLayer(

(norm1): BatchNorm2d(864, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(864, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer21): _DenseLayer(

(norm1): BatchNorm2d(896, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(896, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer22): _DenseLayer(

(norm1): BatchNorm2d(928, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(928, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer23): _DenseLayer(

(norm1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

```
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(960, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
    (denselayer24): _DenseLayer(
      (norm1): BatchNorm2d(992, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(992, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
  )
  (transition3): _Transition(
    (norm): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
  )
  (denseblock4): _DenseBlock(
    (denselayer1): _DenseLayer(
      (norm1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
```

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer2): _DenseLayer(

(norm1): BatchNorm2d(544, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(544, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer3): _DenseLayer(

(norm1): BatchNorm2d(576, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(576, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer4): _DenseLayer(

(norm1): BatchNorm2d(608, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(608, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer5): _DenseLayer(

(norm1): BatchNorm2d(640, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(640, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer6): _DenseLayer(

(norm1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(672, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer7): _DenseLayer(

(norm1): BatchNorm2d(704, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(704, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer8): _DenseLayer(

(norm1): BatchNorm2d(736, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(736, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer9): _DenseLayer(

(norm1): BatchNorm2d(768, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(768, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer10): _DenseLayer(

(norm1): BatchNorm2d(800, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(800, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer11): _DenseLayer(

(norm1): BatchNorm2d(832, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(832, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer12): _DenseLayer(

(norm1): BatchNorm2d(864, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(864, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer13): _DenseLayer(

(norm1): BatchNorm2d(896, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(896, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

(denselayer14): _DenseLayer(

(norm1): BatchNorm2d(928, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu1): ReLU(inplace=True)

(conv1): Conv2d(928, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)

(norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(relu2): ReLU(inplace=True)

(conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

)

```
    (denselayer15): _DenseLayer(
      (norm1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(960, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
    (denselayer16): _DenseLayer(
      (norm1): BatchNorm2d(992, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu1): ReLU(inplace=True)
      (conv1): Conv2d(992, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu2): ReLU(inplace=True)
      (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    )
  )
  (norm5): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
 )
 (classifier): Linear(in_features=1024, out_features=1000, bias=True)
)
```

**EfficientNet-B0:**

EfficientNet-B0 is a deep neural network architecture that was introduced in 2019. It is a family of models that are designed to be efficient in terms of both computation and memory. EfficientNet-B0 uses a combination of neural architecture search (NAS) and scaling techniques to achieve a high level of accuracy while minimizing computational resources.

EfficientNet-B0 has 0.5 million parameters and 5.3 million floating-point operations (FLOPS), which is much smaller than other state-of-the-art models. The network consists of a series of blocks, where each block contains several convolutional layers, including depthwise convolutions and pointwise convolutions. The network also uses a technique called "swish" activation, which is a smooth, non-monotonic function that is faster and more accurate than the traditional ReLU activation function.

In terms of inferencing techniques, EfficientNet-B0 can be used for both classification and object detection tasks. It can be trained using supervised learning, and the weights can be fine-tuned using transfer learning.

Architecture:

```
EfficientNet(
  (features): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Sequential(
      (0): MBConv(
        (block): Sequential(
          (0): Conv2dNormActivation(
            (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
            (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
            (2): SiLU(inplace=True)
          )
          (1): SqueezeExcitation(
            (avgpool): AdaptiveAvgPool2d(output_size=1)
            (fc1): Conv2d(32, 8, kernel_size=(1, 1), stride=(1, 1))
            (fc2): Conv2d(8, 32, kernel_size=(1, 1), stride=(1, 1))
            (activation): SiLU(inplace=True)
            (scale_activation): Sigmoid()
          )
          (2): Conv2dNormActivation(
            (0): Conv2d(32, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
            (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          )
        )
        (stochastic_depth): StochasticDepth(p=0.0, mode=row)
```

```
  )
)
(2): Sequential(
  (0): MBConv(
    (block): Sequential(
      (0): Conv2dNormActivation(
        (0): Conv2d(16, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): SiLU(inplace=True)
      )
      (1): Conv2dNormActivation(
        (0): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=96, bias=False)
        (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): SiLU(inplace=True)
      )
      (2): SqueezeExcitation(
        (avgpool): AdaptiveAvgPool2d(output_size=1)
        (fc1): Conv2d(96, 4, kernel_size=(1, 1), stride=(1, 1))
        (fc2): Conv2d(4, 96, kernel_size=(1, 1), stride=(1, 1))
        (activation): SiLU(inplace=True)
        (scale_activation): Sigmoid()
      )
      (3): Conv2dNormActivation(
        (0): Conv2d(96, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (stochastic_depth): StochasticDepth(p=0.0125, mode=row)
  )
  (1): MBConv(
    (block): Sequential(
```

```
      (0): Conv2dNormActivation(
        (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): SiLU(inplace=True)
      )
      (1): Conv2dNormActivation(
        (0): Conv2d(144, 144, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=144, bias=False)
        (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): SiLU(inplace=True)
      )
      (2): SqueezeExcitation(
        (avgpool): AdaptiveAvgPool2d(output_size=1)
        (fc1): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
        (fc2): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
        (activation): SiLU(inplace=True)
        (scale_activation): Sigmoid()
      )
      (3): Conv2dNormActivation(
        (0): Conv2d(144, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (stochastic_depth): StochasticDepth(p=0.025, mode=row)
  )
)
(3): Sequential(
  (0): MBConv(
    (block): Sequential(
      (0): Conv2dNormActivation(
        (0): Conv2d(24, 144, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(144, 144, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), groups=144, bias=False)
      (1): BatchNorm2d(144, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(144, 6, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(6, 144, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(144, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(40, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.037500000000000006, mode=row)
)
(1): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(240, 240, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=240, bias=False)
      (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(240, 40, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(40, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.05, mode=row)
 )
)
(4): Sequential(
 (0): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(40, 240, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(240, 240, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=240, bias=False)
      (1): BatchNorm2d(240, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
```

```
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(240, 10, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(10, 240, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(240, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.0625, mode=row)
)
(1): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=480, bias=False)
      (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
```

```
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.07500000000000001, mode=row)
)
(2): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(480, 480, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=480, bias=False)
      (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(480, 80, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(80, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
```

```
    )
    (stochastic_depth): StochasticDepth(p=0.08750000000000001, mode=row)
  )
)
(5): Sequential(
  (0): MBConv(
    (block): Sequential(
      (0): Conv2dNormActivation(
        (0): Conv2d(80, 480, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): SiLU(inplace=True)
      )
      (1): Conv2dNormActivation(
        (0): Conv2d(480, 480, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=480, bias=False)
        (1): BatchNorm2d(480, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): SiLU(inplace=True)
      )
      (2): SqueezeExcitation(
        (avgpool): AdaptiveAvgPool2d(output_size=1)
        (fc1): Conv2d(480, 20, kernel_size=(1, 1), stride=(1, 1))
        (fc2): Conv2d(20, 480, kernel_size=(1, 1), stride=(1, 1))
        (activation): SiLU(inplace=True)
        (scale_activation): Sigmoid()
      )
      (3): Conv2dNormActivation(
        (0): Conv2d(480, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (stochastic_depth): StochasticDepth(p=0.1, mode=row)
  )
```

```
(1): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=672, bias=False)
      (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.1125, mode=row)
)
(2): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=672, bias=False)
      (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(672, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(112, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.125, mode=row)
 )
)
(6): Sequential(
 (0): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(112, 672, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
```

```
          (0): Conv2d(672, 672, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), groups=672, bias=False)
          (1): BatchNorm2d(672, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
        (2): SqueezeExcitation(
          (avgpool): AdaptiveAvgPool2d(output_size=1)
          (fc1): Conv2d(672, 28, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(28, 672, kernel_size=(1, 1), stride=(1, 1))
          (activation): SiLU(inplace=True)
          (scale_activation): Sigmoid()
        )
        (3): Conv2dNormActivation(
          (0): Conv2d(672, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (stochastic_depth): StochasticDepth(p=0.1375, mode=row)
    )
    (1): MBConv(
      (block): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=1152,
bias=False)
          (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
```

```
        (2): SqueezeExcitation(
          (avgpool): AdaptiveAvgPool2d(output_size=1)
          (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
          (activation): SiLU(inplace=True)
          (scale_activation): Sigmoid()
        )
        (3): Conv2dNormActivation(
          (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (stochastic_depth): StochasticDepth(p=0.15000000000000002, mode=row)
    )
    (2): MBConv(
      (block): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=1152,
bias=False)
          (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
        (2): SqueezeExcitation(
          (avgpool): AdaptiveAvgPool2d(output_size=1)
          (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
```

```
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
      (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (stochastic_depth): StochasticDepth(p=0.1625, mode=row)
)
(3): MBConv(
  (block): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (1): Conv2dNormActivation(
      (0): Conv2d(1152, 1152, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2), groups=1152,
bias=False)
      (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): SiLU(inplace=True)
    )
    (2): SqueezeExcitation(
      (avgpool): AdaptiveAvgPool2d(output_size=1)
      (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
      (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
      (activation): SiLU(inplace=True)
      (scale_activation): Sigmoid()
    )
    (3): Conv2dNormActivation(
```

```
          (0): Conv2d(1152, 192, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (stochastic_depth): StochasticDepth(p=0.17500000000000002, mode=row)
    )
  )
  (7): Sequential(
    (0): MBConv(
      (block): Sequential(
        (0): Conv2dNormActivation(
          (0): Conv2d(192, 1152, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
        (1): Conv2dNormActivation(
          (0): Conv2d(1152, 1152, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=1152,
bias=False)
          (1): BatchNorm2d(1152, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): SiLU(inplace=True)
        )
        (2): SqueezeExcitation(
          (avgpool): AdaptiveAvgPool2d(output_size=1)
          (fc1): Conv2d(1152, 48, kernel_size=(1, 1), stride=(1, 1))
          (fc2): Conv2d(48, 1152, kernel_size=(1, 1), stride=(1, 1))
          (activation): SiLU(inplace=True)
          (scale_activation): Sigmoid()
        )
        (3): Conv2dNormActivation(
          (0): Conv2d(1152, 320, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(320, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
        )
       )
       (stochastic_depth): StochasticDepth(p=0.1875, mode=row)
     )
    )
    (8): Conv2dNormActivation(
     (0): Conv2d(320, 1280, kernel_size=(1, 1), stride=(1, 1), bias=False)
     (1): BatchNorm2d(1280, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (2): SiLU(inplace=True)
    )
   )
   (avgpool): AdaptiveAvgPool2d(output_size=1)
   (classifier): Sequential(
    (0): Dropout(p=0.2, inplace=True)
    (1): Linear(in_features=1280, out_features=1000, bias=True)
   )
)
```

**ConvNeXt-T:**

ConvNeXt-T is a deep neural network architecture that was introduced in 2017. It is a variant of the Convolutional Neural Network with NeXt blocks (ConvNeXt) architecture, which is designed to improve the accuracy and efficiency of deep neural networks. ConvNeXt-T uses a combination of parallel pathways and channel gating to improve the flow of information through the network.

ConvNeXt-T has 34 layers, including convolutional layers

Architecture:

```
ConvNeXt(
  (features): Sequential(
    (0): Conv2dNormActivation(
      (0): Conv2d(3, 96, kernel_size=(4, 4), stride=(4, 4))
      (1): LayerNorm2d((96,), eps=1e-06, elementwise_affine=True)
    )
    (1): Sequential(
      (0): CNBlock(
        (block): Sequential(
```

```
      (0): Conv2d(96, 96, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=96)
      (1): Permute()
      (2): LayerNorm((96,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=96, out_features=384, bias=True)
      (4): GELU(approximate='none')
      (5): Linear(in_features=384, out_features=96, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.0, mode=row)
  )
  (1): CNBlock(
    (block): Sequential(
      (0): Conv2d(96, 96, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=96)
      (1): Permute()
      (2): LayerNorm((96,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=96, out_features=384, bias=True)
      (4): GELU(approximate='none')
      (5): Linear(in_features=384, out_features=96, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.0058823529411764705, mode=row)
  )
  (2): CNBlock(
    (block): Sequential(
      (0): Conv2d(96, 96, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=96)
      (1): Permute()
      (2): LayerNorm((96,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=96, out_features=384, bias=True)
      (4): GELU(approximate='none')
      (5): Linear(in_features=384, out_features=96, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.011764705882352941, mode=row)
  )
)
(2): Sequential(
  (0): LayerNorm2d((96,), eps=1e-06, elementwise_affine=True)
  (1): Conv2d(96, 192, kernel_size=(2, 2), stride=(2, 2))
)
(3): Sequential(
  (0): CNBlock(
    (block): Sequential(
      (0): Conv2d(192, 192, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=192)
      (1): Permute()
      (2): LayerNorm((192,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=192, out_features=768, bias=True)
      (4): GELU(approximate='none')
```

```
      (5): Linear(in_features=768, out_features=192, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.017647058823529415, mode=row)
  )
  (1): CNBlock(
    (block): Sequential(
      (0): Conv2d(192, 192, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=192)
      (1): Permute()
      (2): LayerNorm((192,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=192, out_features=768, bias=True)
      (4): GELU(approximate='none')
      (5): Linear(in_features=768, out_features=192, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.023529411764705882, mode=row)
  )
  (2): CNBlock(
    (block): Sequential(
      (0): Conv2d(192, 192, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=192)
      (1): Permute()
      (2): LayerNorm((192,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=192, out_features=768, bias=True)
      (4): GELU(approximate='none')
      (5): Linear(in_features=768, out_features=192, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.029411764705882353, mode=row)
  )
)
(4): Sequential(
  (0): LayerNorm2d((192,), eps=1e-06, elementwise_affine=True)
  (1): Conv2d(192, 384, kernel_size=(2, 2), stride=(2, 2))
)
(5): Sequential(
  (0): CNBlock(
    (block): Sequential(
      (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
      (1): Permute()
      (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
      (3): Linear(in_features=384, out_features=1536, bias=True)
      (4): GELU(approximate='none')
      (5): Linear(in_features=1536, out_features=384, bias=True)
      (6): Permute()
    )
    (stochastic_depth): StochasticDepth(p=0.03529411764705883, mode=row)
  )
```

```
(1): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.0411764705882353, mode=row)
)
(2): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.047058823529411764, mode=row)
)
(3): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.052941176470588235, mode=row)
)
(4): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.058823529411764705, mode=row)
```

```
)
(5): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.06470588235294118, mode=row)
)
(6): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.07058823529411766, mode=row)
)
(7): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
  (stochastic_depth): StochasticDepth(p=0.07647058823529412, mode=row)
)
(8): CNBlock(
  (block): Sequential(
    (0): Conv2d(384, 384, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=384)
    (1): Permute()
    (2): LayerNorm((384,), eps=1e-06, elementwise_affine=True)
    (3): Linear(in_features=384, out_features=1536, bias=True)
    (4): GELU(approximate='none')
    (5): Linear(in_features=1536, out_features=384, bias=True)
    (6): Permute()
  )
```

```
      (stochastic_depth): StochasticDepth(p=0.0823529411764706, mode=row)
    )
  )
  (6): Sequential(
    (0): LayerNorm2d((384,), eps=1e-06, elementwise_affine=True)
    (1): Conv2d(384, 768, kernel_size=(2, 2), stride=(2, 2))
  )
  (7): Sequential(
    (0): CNBlock(
      (block): Sequential(
        (0): Conv2d(768, 768, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=768)
        (1): Permute()
        (2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
        (3): Linear(in_features=768, out_features=3072, bias=True)
        (4): GELU(approximate='none')
        (5): Linear(in_features=3072, out_features=768, bias=True)
        (6): Permute()
      )
      (stochastic_depth): StochasticDepth(p=0.08823529411764706, mode=row)
    )
    (1): CNBlock(
      (block): Sequential(
        (0): Conv2d(768, 768, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=768)
        (1): Permute()
        (2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
        (3): Linear(in_features=768, out_features=3072, bias=True)
        (4): GELU(approximate='none')
        (5): Linear(in_features=3072, out_features=768, bias=True)
        (6): Permute()
      )
      (stochastic_depth): StochasticDepth(p=0.09411764705882353, mode=row)
    )
    (2): CNBlock(
      (block): Sequential(
        (0): Conv2d(768, 768, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3), groups=768)
        (1): Permute()
        (2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
        (3): Linear(in_features=768, out_features=3072, bias=True)
        (4): GELU(approximate='none')
        (5): Linear(in_features=3072, out_features=768, bias=True)
        (6): Permute()
      )
      (stochastic_depth): StochasticDepth(p=0.1, mode=row)
    )
  )
)
(avgpool): AdaptiveAvgPool2d(output_size=1)
```

```
  (classifier): Sequential(
    (0): LayerNorm2d((768,), eps=1e-06, elementwise_affine=True)
    (1): Flatten(start_dim=1, end_dim=-1)
    (2): Linear(in_features=768, out_features=1000, bias=True)
  )
)
```