CSCE 636: Deep Learning (Fall 2022)
Assignment #2

TEXAS A&M UNIVERSITY

20 Oct 2022

Submission By:
Ashutosh Chauhan
UIN: 232009024
Email: ashutosh@tamu.edu

a) **Download the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/ cifar.html) and complete "DataReader.py". For the dataset, you can download any version. But make sure you write corresponding code in "DataReader.py" to read it.**

Downloaded CIFAR-100 python version data set and kept the files inside project directory. Loaded data files in corresponding X_train, y_train, X_test and y_test basis the name of data set files.

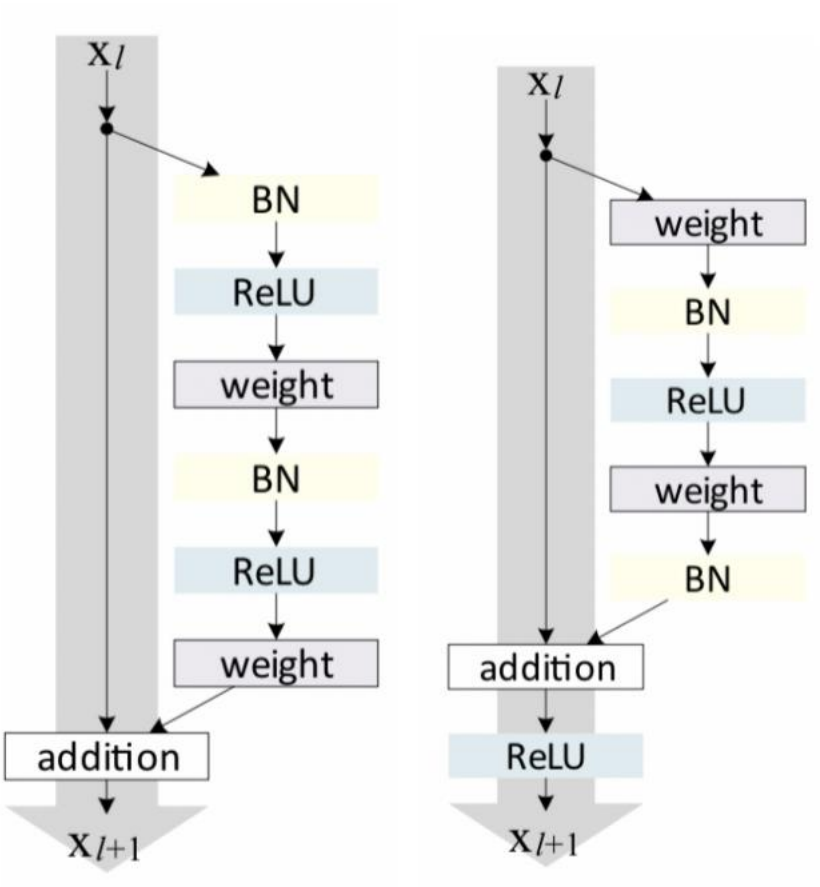
```
### YOUR CODE HERE
fnames = os.listdir(data_dir)
x_train = np.array([[]]).reshape(0,3072)
y_train = np.array([])
for fn in fnames:
    if fn.endswith(".html") or fn.endswith(".meta"):
        continue
    with open(os.path.join(data_dir,fn), 'rb') as fo:
        ds = pickle.load(fo, encoding='bytes')
        xtemp = np.array(ds[b'data'])
        ytemp = np.array(ds[b'labels'])

    if fn.startswith("test_batch"):
        x_test = xtemp
        y_test = ytemp

    if fn.startswith("data_batch"):
        x_train = np.concatenate((xtemp,x_train), axis=0)
        y_train = np.concatenate((ytemp,y_train), axis=0)

### YOUR CODE HERE
```

b) Implement data augmentation. To complete "ImageUtils.py", you will im- plement the augmentation process for a single image using numpy.

Reshaped the image from [3072,1] to [32,32,3] using reshape and transpose function. Padded the image with 4 pixels on each sides using numpy.pad() function. Randomly cropped and flipped image horizontally. Subtracted each pixel of image by mean of all pixels and divided by standard deviation

```
def preprocess_image(image, training):
    """ Preprocess a single image of shape [height, width, depth].

    Args:
        image: An array of shape [32, 32, 3].
        training: A boolean. Determine whether it is in training mode.

    Returns:
        image: An array of shape [32, 32, 3].
    """
    if training:
        ### YOUR CODE HERE
        # Resize the image to add four extra pixels on each side.

        ### YOUR CODE HERE
        layers = [np.pad(image[:,:,channel], ((4,4),(4,4)),'constant', constant_values=255) for channel in range(3)]
        image = np.stack(layers, axis=2)


        ### YOUR CODE HERE
        # Randomly crop a [32, 32] section of the image.
        # HINT: randomly generate the upper left point of the image

        ### YOUR CODE HERE
        topleft = np.random.randint(0,9, (2,1))
        image = image[int(topleft[0]):int(topleft[0])+32, int(topleft[1]):int(topleft[1])+32,:]


        ### YOUR CODE HERE
        # Randomly flip the image horizontally.
        image = image if np.random.randint(0,2) else np.fliplr(image)
        ### YOUR CODE HERE


    ### YOUR CODE HERE
    # Subtract off the mean and divide by the standard deviation of the pixels.

    image = (image - np.mean(image))/np.std(image)
    ### YOUR CODE HERE
    return image
```

c) Complete "NetWork.py". Read the required materials carefully before this step. You are asked to implement two versions of ResNet: version 1 uses original residual blocks (Figure4(a) in [2]) and version 2 uses full pre-activation residual blocks (Figure4(e) in [2]). In particular, for version 2, implement the bottleneck blocks instead of standard residual blocks. In this step, only basic PyTorch APIs in torch.nn and torch.optim are allowed to use.



Two blocks are used, standard block, bottleneck block.

Stack layer is used to build a stack of bottleneck and standard blocks of resnet size 18.

```python
class standard_block(nn.Module):

    def __init__(self, filters, projection_shortcut, strides, first_num_filters) -> None:
        super(standard_block, self).__init__()
        ### YOUR CODE HERE
        if projection_shortcut is not None:
            self.projection = nn.Sequential(
                nn.Conv2d(in_channels=first_num_filters,out_channels=filters,kernel_size=projection_shortcut,stride=strides,padding=0),
                nn.BatchNorm2d(num_features=filters,eps=1e-5,momentum=0.995)
            )
        else:
            self.projection = nn.Identity()
        ### YOUR CODE HERE
        self.layer1 = nn.Sequentia (parameter) first_num_filters: Any
            nn.Conv2d(in_channels=first_num_filters,out_channels=filters,kernel_size=(3,3),stride=strides,padding=1 if projection_shortcut is not None else 'same'),
            batch_norm_relu_layer(num_features=filters)
        )
        self.layer2 = nn.Sequential(
            nn.Conv2d(in_channels=filters,out_channels=filters,kernel_size=(3,3),stride=1,padding='same'),
            nn.BatchNorm2d(num_features=filters,eps=1e-5,momentum=0.995)
        )
        self.output_activation = nn.ReLU()
        ### YOUR CODE HERE

    def forward(self, inputs: Tensor) -> Tensor:
        ### YOUR CODE HERE
        projection = self.projection(inputs)
        layer1 = self.layer1(inputs)
        layer2 = self.layer2(layer1)
        output = torch.add(projection,layer2)
        return self.output_activation(output)
        ### YOUR CODE HERE
```

```python
class bottleneck_block(nn.Module):

    def __init__(self, filters, projection_shortcut, strides, first_num_filters) -> None:
        super(bottleneck_block, self).__init__()

        ### YOUR CODE HERE
        # Hint: Different from standard lib implementation, you need pay attention to
        # how to define in_channel of the first bn and conv of each block based on
        # Args given above.
        if projection_shortcut is not None:
            self.projection = nn.Sequential(
                batch_norm_relu_layer(num_features=first_num_filters),
                nn.Conv2d(in_channels=first_num_filters,out_channels=filters,kernel_size=projection_shortcut,stride=strides,padding=0)
            )
        else:
            self.projection = nn.Identity()

        self.layer1 = nn.Sequential(
            batch_norm_relu_layer(num_features=first_num_filters),
            nn.Conv2d(in_channels=first_num_filters,out_channels=filters//4,kernel_size=(1,1),stride=strides,padding=0 if strides>1 else 'same'
            )
        )
        self.layer2 = nn.Sequential(
            batch_norm_relu_layer(num_features=filters//4),
            nn.Conv2d(in_channels=filters//4,out_channels=filters//4,kernel_size=(3,3),stride=1,padding='same')
        )
        self.layer3 = nn.Sequential(
            batch_norm_relu_layer(num_features=filters//4),
            nn.Conv2d(in_channels=filters//4,out_channels=filters,kernel_size=(1,1),stride=1,padding='same')
        )
        ### YOUR CODE HERE

    def forward(self, inputs: Tensor) -> Tensor:
        ### YOUR CODE HERE
        # The projection shortcut should come after the first batch norm and ReLU
        # since it performs a 1x1 convolution.
        projection = self.projection(inputs)
        layer1 = self.layer1(inputs)
        layer2 = self.layer2(layer1)
        layer3 = self.layer3(layer2)
        output = torch.add(projection,layer3)
        return output
        ### YOUR CODE HERE
```

```python
class stack_layer(nn.Module):

    def __init__(self, filters, block_fn, strides, resnet_size, first_num_filters) -> None:
        super(stack_layer, self).__init__()
        filters_out = filters * 4 if block_fn is bottleneck_block else filters
        ### END CODE HERE
        # projection_shortcut =?
        # Only the first block per stack_layer uses projection_shortcut and strides

        self.stack = nn.ModuleList()
        in_channels=filters_out//2 if strides>1 else filters
        self.stack.append(block_fn(filters=filters_out,projection_shortcut=(1,1) if strides>1 or block_fn is bottleneck_block else None, strides=strides,first_num_fi

        for i in range(1,resnet_size):
            self.stack.append(block_fn(filters=filters_out,projection_shortcut=None, strides=1, first_num_filters=filters_out))

        ### END CODE HERE

    def forward(self, inputs: Tensor) -> Tensor:
        ### END CODE HERE
        x_in = inputs
        for i in range(len(self.stack)):
            x_in = self.stack[i](x_in)
        return x_in
        ### END CODE HERE
```

```python
class output_layer(nn.Module):

    def __init__(self, filters, resnet_version, num_classes) -> None:
        super(output_layer, self).__init__()
        # Only apply the BN and ReLU for model that does pre_activation in each
        # bottleneck block, e.g. resnet V2.
        if (resnet_version == 2):                    (parameter) filters: Any
            self.bn_relu = batch_norm_relu_layer(filters, eps=1e-5, momentum=0.997)

        ### END CODE HERE
        self.global_avg_pooling_layer = nn.Sequential(nn.AdaptiveAvgPool2d((1,1)),nn.Flatten())
        in_features = filters if resnet_version==2 else filters//4
        self.fc_layer = nn.Linear(in_features=in_features,out_features=num_classes)
        ### END CODE HERE

    def forward(self, inputs: Tensor) -> Tensor:
        ### END CODE HERE
        x_in = self.global_avg_pooling_layer(inputs)
        x_in = self.fc_layer(x_in)
        return x_in
        ### END CODE HERE
```

d) Complete "Model.py". Note: For this step and last step, pay attention to how to use batch normalization.

Cross Entropy loss was calculated using nn.CrossEntropyLoss() function. I used Stochastic gradient descent optimizer with learning rate 0.11 and momentum of 0.88. Then we defined training and testing functions. In training function batch size for each epoch was decided and then we trained complete model for 200 epoches using both network version 1 and network version 2. After every epochs that is multiple of 95 and 140, I am reducing the learning rate by a factor of 10.

```python
    self.loss_fun = nn.CrossEntropyLoss()
    self.optimizer = torch.optim.SGD(self.network.parameters(),lr=0.11,weight_decay=self.config.weight_decay,momentum=0.88)
```

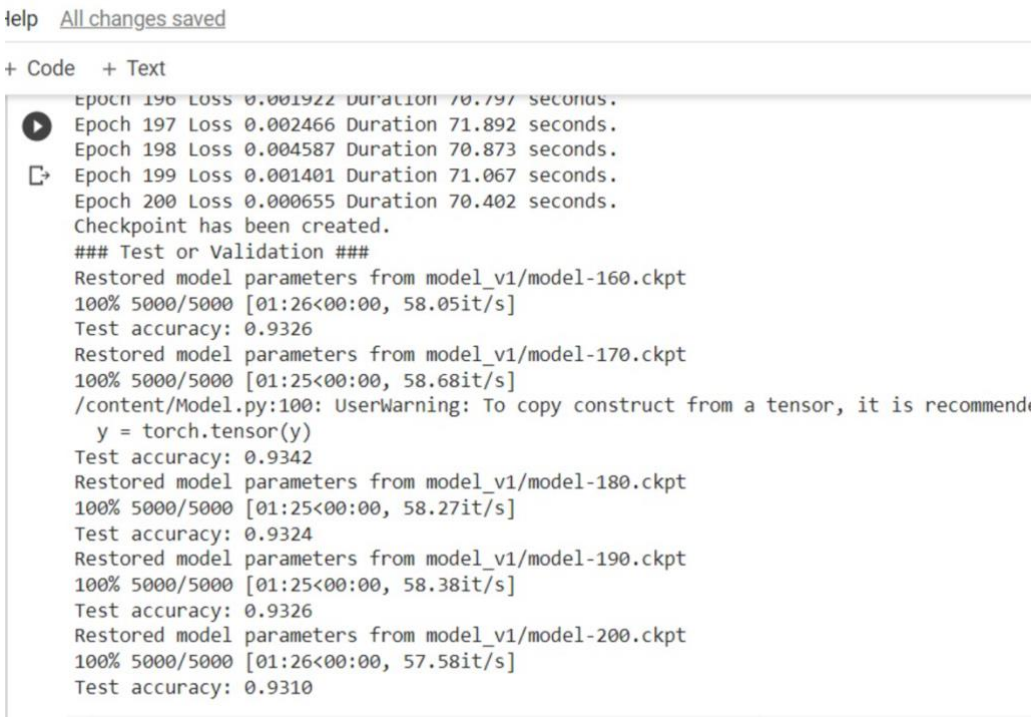
```
def train(self, x_train, y_train, max_epoch):
    self.network.train()
    # Determine how many batches in an epoch
    num_samples = x_train.shape[0]
    num_batches = num_samples // self.config.batch_size



    print('### Training... ###')
    for epoch in range(1, max_epoch+1):
        start_time = time.time()
        # Shuffle
        shuffle_index = np.random.permutation(num_samples)
        curr_x_train = x_train[shuffle_index]
        curr_y_train = y_train[shuffle_index]

        ### YOUR CODE HERE
        # Set the learning rate for this epoch
        # Usage example: divide the initial learning rate by 10 after several epochs

        if (epoch%95==0 or epoch%140==0):
            self.optimizer.param_groups[0]['lr'] = self.optimizer.param_groups[0]['lr']/10.
        ### YOUR CODE HERE

        for i in range(num_batches):
            ### YOUR CODE HERE
            # Construct the current batch.
            # Don't forget to use "parse_record" to perform data preprocessing.
            # Don't forget L2 weight decay

            X_batch = curr_x_train[i*self.config.batch_size:min((i+1)*self.config.batch_size,curr_x_train.shape[0])]
            y_batch = curr_y_train[i*self.config.batch_size:min((i+1)*self.config.batch_size,curr_y_train.shape[0])]
            #preprocessing data after constructing current batch
            X_batch = np.array(list(map(lambda x: parse_record(x,True),X_batch)))
            X_batch = torch.tensor(X_batch,device=self.device, dtype=torch.float)

            prediction = self.network.forward(X_batch)
            y_batch = torch.tensor(y_batch,device=self.device)
            y_batch = y_batch.to(torch.long)
            loss = self.loss_fun(prediction,y_batch)
            ### YOUR CODE HERE
            self.optimizer.zero_grad()
            loss.backward()
            self.optimizer.step()
```

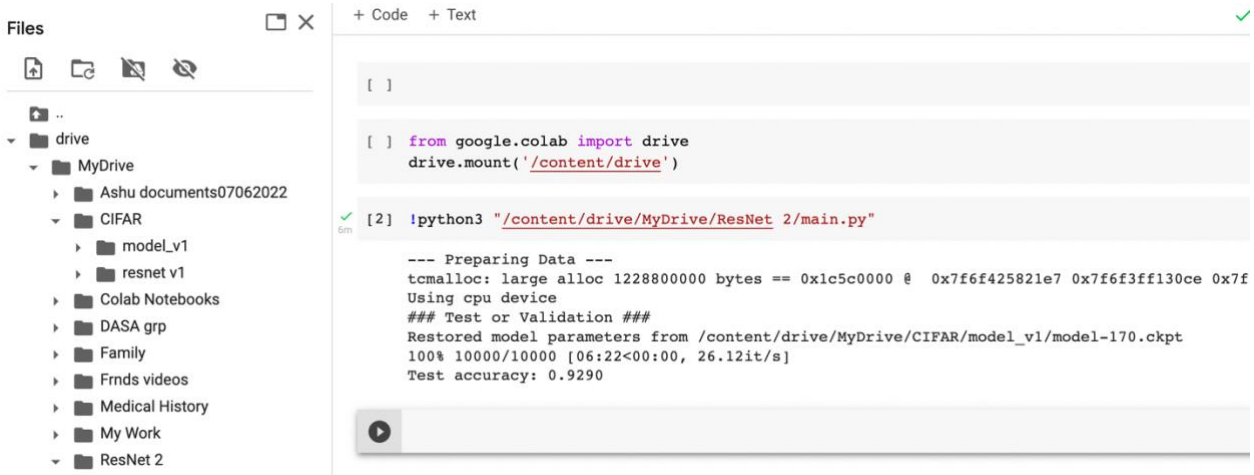
```python
def test_or_validate(self, x, y, checkpoint_num_list):
    self.network.eval()
    print('### Test or Validation ###')
    for checkpoint_num in checkpoint_num_list:
        checkpointfile = os.path.join(self.config.modeldir, 'model-%d.ckpt'%(checkpoint_num))
        self.load(checkpointfile)

        preds = []
        with torch.no_grad():
            for i in tqdm(range(x.shape[0])):
                ### YOUR CODE HERE
                x_processed = np.array(list(map(lambda x: parse_record(x,False),x[i:i+1])))
                x_processed = torch.tensor(x_processed,device=self.device,dtype=torch.float)
                preds.append(torch.argmax(self.network.forward(x_processed),axis=1))
                ### END CODE HERE

        y = torch.tensor(y)
        preds = torch.tensor(preds)
        print('Test accuracy: {:.4f}'.format(torch.sum(preds==y)/y.shape[0]))
```

e) Tune all the hyperparameters in "main.py" and report your final testing accuracy.

Version 1: I trained the model for 200 Epochs. Validation accuracy was checked with epoch hyperparameter of 160, 170, 180, 190 and 200.

Highest accuracy was found with 170 epoch. Model was trained again with hyperparameter of 170 epoch and model was feeded with testing data. Testing accuracy was found to be **92.90 %.** Screenshots are attached below for reference.





Version 2: I trained model with version 2 network and run it for 120 epoch. Validation accuracy was checked with epoch hyperparameter of 90, 100, 110 and 120.

Highest accuracy was found with 120 epoch. Model was trained again with hyperparameter of 120 epoch and model was feeded with testing data. Testing accuracy was found to be **92.96 %.** Screenshots are attached below for reference.

```
[3]
--- Preparing Data ---
tcmalloc: large alloc 1228800000 bytes == 0x1df76000 @  0x7fbb134e01e7 0x7fbb10e710ce 0x7fbb10ec7cf5 0x7fbb10f7086d 0x7fbb
Using cuda device
### Test or Validation ###
Restored model parameters from /content/drive/MyDrive/ResNet/model_v2/model-90.ckpt
100% 5000/5000 [02:02<00:00, 40.93it/s]
Test accuracy: 0.9146
Restored model parameters from /content/drive/MyDrive/ResNet/model_v2/model-100.ckpt
100% 5000/5000 [02:00<00:00, 41.40it/s]
/content/drive/MyDrive/ResNet/Model.py:101: UserWarning: To copy construct from a tensor, it is recommended to use sourceT
  y = torch.tensor(y)
Test accuracy: 0.9582
Restored model parameters from /content/drive/MyDrive/ResNet/model_v2/model-110.ckpt
100% 5000/5000 [01:58<00:00, 42.05it/s]
Test accuracy: 0.9616
Restored model parameters from /content/drive/MyDrive/ResNet/model_v2/model-120.ckpt
100% 5000/5000 [01:59<00:00, 41.73it/s]
Test accuracy: 0.9646
```

```
!python /content/drive/MyDrive/ResNet/main.py

--- Preparing Data ---
tcmalloc: large alloc 1228800000 bytes == 0x1e2c2000 @  0x7fa9470491e7 0x7fa9449da0ce 0x7fa944a30cf5 0x7fa944ad986d 0x7fa944ada17f 0x7f
Using cuda device
### Test or Validation ###
Restored model parameters from /content/drive/MyDrive/ResNet/model_v2/model-120.ckpt
100% 10000/10000 [04:02<00:00, 41.25it/s]
Test accuracy: 0.9296
```

APPENDIX:

Version 1

```
├-- Preparing Data ---
tcmalloc: large alloc 1228800000 bytes == 0xa0496000 @
0x7fc07cdfc1e7 0x7fc07a74d0ce 0x7fc07a7a3cf5 0x7fc07a84c86d
0x7fc07a84d17f 0x7fc07a84d2d0 0x4ba72b 0x7fc07a78e944 0x58f67f
0x50ff13 0x5b4ee6 0x58ff2e 0x50d482 0x58fd37 0x50c4fc 0x58fd37
0x50c4fc 0x5b4ee6 0x6005a3 0x607796 0x60785c 0x60a436 0x64db82
0x64dd2e 0x7fc07c9f9c87 0x5b636a
Using cuda device
### Training... ###
Epoch 1 Loss 1.933176 Duration 77.511 seconds.
Epoch 2 Loss 1.768240 Duration 71.773 seconds.
Epoch 3 Loss 1.473439 Duration 72.585 seconds.
Epoch 4 Loss 1.481856 Duration 72.728 seconds.
Epoch 5 Loss 1.071983 Duration 72.122 seconds.
Epoch 6 Loss 1.022431 Duration 72.020 seconds.
Epoch 7 Loss 0.863509 Duration 72.549 seconds.
Epoch 8 Loss 0.704582 Duration 71.947 seconds.
Epoch 9 Loss 0.639420 Duration 71.799 seconds.
Epoch 10 Loss 0.620936 Duration 72.294 seconds.
Checkpoint has been created.
Epoch 11 Loss 0.607533 Duration 71.927 seconds.
Epoch 12 Loss 0.564128 Duration 71.842 seconds.
Epoch 13 Loss 0.546423 Duration 72.285 seconds.
Epoch 14 Loss 0.564393 Duration 71.624 seconds.
Epoch 15 Loss 0.470974 Duration 71.774 seconds.
Epoch 16 Loss 0.537132 Duration 72.255 seconds.
Epoch 17 Loss 0.686993 Duration 71.845 seconds.
Epoch 18 Loss 0.545412 Duration 71.706 seconds.
Epoch 19 Loss 0.491481 Duration 72.303 seconds.
Epoch 20 Loss 0.344640 Duration 71.804 seconds.
Checkpoint has been created.
Epoch 21 Loss 0.466510 Duration 72.237 seconds.
Epoch 22 Loss 0.455249 Duration 72.623 seconds.
Epoch 23 Loss 0.504621 Duration 72.098 seconds.
Epoch 24 Loss 0.384757 Duration 71.854 seconds.
Epoch 25 Loss 0.398705 Duration 72.320 seconds.
Epoch 26 Loss 0.398576 Duration 71.746 seconds.
Epoch 27 Loss 0.376518 Duration 71.900 seconds.
Epoch 28 Loss 0.402342 Duration 72.474 seconds.
Epoch 29 Loss 0.359782 Duration 71.838 seconds.
Epoch 30 Loss 0.303382 Duration 71.907 seconds.
Checkpoint has been created.
Epoch 31 Loss 0.302946 Duration 72.079 seconds.
Epoch 32 Loss 0.434045 Duration 71.801 seconds.
Epoch 33 Loss 0.300648 Duration 72.248 seconds.
Epoch 34 Loss 0.230233 Duration 71.755 seconds.
Epoch 35 Loss 0.379239 Duration 71.785 seconds.
Epoch 36 Loss 0.318447 Duration 72.665 seconds.
Epoch 37 Loss 0.227609 Duration 72.311 seconds.
```

```
Epoch 38 Loss 0.268057 Duration 71.804 seconds.
Epoch 39 Loss 0.313987 Duration 72.246 seconds.
Epoch 40 Loss 0.357273 Duration 72.353 seconds.
Checkpoint has been created.
Epoch 41 Loss 0.341619 Duration 71.777 seconds.
Epoch 42 Loss 0.299034 Duration 72.305 seconds.
Epoch 43 Loss 0.238423 Duration 72.178 seconds.
Epoch 44 Loss 0.248159 Duration 71.853 seconds.
Epoch 45 Loss 0.284846 Duration 72.680 seconds.
Epoch 46 Loss 0.317930 Duration 71.600 seconds.
Epoch 47 Loss 0.262185 Duration 71.694 seconds.
Epoch 48 Loss 0.249243 Duration 72.914 seconds.
Epoch 49 Loss 0.203314 Duration 71.634 seconds.
Epoch 50 Loss 0.227368 Duration 72.009 seconds.
Checkpoint has been created.
Epoch 51 Loss 0.347420 Duration 72.240 seconds.
Epoch 52 Loss 0.243081 Duration 71.875 seconds.
Epoch 53 Loss 0.403184 Duration 72.313 seconds.
Epoch 54 Loss 0.269906 Duration 72.107 seconds.
Epoch 55 Loss 0.265178 Duration 71.877 seconds.
Epoch 56 Loss 0.230614 Duration 72.038 seconds.
Epoch 57 Loss 0.213946 Duration 71.913 seconds.
Epoch 58 Loss 0.298552 Duration 71.572 seconds.
Epoch 59 Loss 0.380853 Duration 72.013 seconds.
Epoch 60 Loss 0.321853 Duration 71.859 seconds.
Checkpoint has been created.
Epoch 61 Loss 0.210753 Duration 71.464 seconds.
Epoch 62 Loss 0.349826 Duration 72.036 seconds.
Epoch 63 Loss 0.225445 Duration 71.783 seconds.
Epoch 64 Loss 0.407722 Duration 71.566 seconds.
Epoch 65 Loss 0.248424 Duration 71.924 seconds.
Epoch 66 Loss 0.277507 Duration 71.647 seconds.
Epoch 67 Loss 0.285101 Duration 71.602 seconds.
Epoch 68 Loss 0.206673 Duration 71.913 seconds.
Epoch 69 Loss 0.321357 Duration 71.490 seconds.
Epoch 70 Loss 0.139293 Duration 71.927 seconds.
Checkpoint has been created.
Epoch 71 Loss 0.240887 Duration 71.704 seconds.
Epoch 72 Loss 0.240344 Duration 71.458 seconds.
Epoch 73 Loss 0.222445 Duration 71.918 seconds.
Epoch 74 Loss 0.194449 Duration 71.591 seconds.
Epoch 75 Loss 0.131751 Duration 71.677 seconds.
Epoch 76 Loss 0.237232 Duration 72.511 seconds.
Epoch 77 Loss 0.167395 Duration 71.590 seconds.
Epoch 78 Loss 0.306818 Duration 71.928 seconds.
Epoch 79 Loss 0.264529 Duration 71.360 seconds.
Epoch 80 Loss 0.294226 Duration 71.559 seconds.
Checkpoint has been created.
Epoch 81 Loss 0.198570 Duration 72.069 seconds.
```

Checkpoint has been created.
Epoch 81 Loss 0.198570 Duration 72.069 seconds.
Epoch 82 Loss 0.323927 Duration 71.454 seconds.
Epoch 83 Loss 0.197942 Duration 71.461 seconds.
Epoch 84 Loss 0.225606 Duration 71.434 seconds.
Epoch 85 Loss 0.304044 Duration 71.412 seconds.
Epoch 86 Loss 0.285581 Duration 71.240 seconds.
Epoch 87 Loss 0.255387 Duration 71.932 seconds.
Epoch 88 Loss 0.169692 Duration 71.310 seconds.
Epoch 89 Loss 0.177714 Duration 71.703 seconds.
Epoch 90 Loss 0.186914 Duration 71.294 seconds.
Checkpoint has been created.
Epoch 91 Loss 0.279258 Duration 71.092 seconds.
Epoch 92 Loss 0.191666 Duration 71.459 seconds.
Epoch 93 Loss 0.286232 Duration 71.025 seconds.
Epoch 94 Loss 0.267893 Duration 71.051 seconds.
Epoch 95 Loss 0.090562 Duration 71.653 seconds.
Epoch 96 Loss 0.109067 Duration 70.927 seconds.
Epoch 97 Loss 0.101534 Duration 71.340 seconds.
Epoch 98 Loss 0.052109 Duration 71.132 seconds.
Epoch 99 Loss 0.030780 Duration 71.161 seconds.
Epoch 100 Loss 0.096555 Duration 71.736 seconds.
Checkpoint has been created.
Epoch 101 Loss 0.053900 Duration 71.352 seconds.
Epoch 102 Loss 0.029490 Duration 72.188 seconds.
Epoch 103 Loss 0.045053 Duration 71.762 seconds.
Epoch 104 Loss 0.028402 Duration 71.135 seconds.
Epoch 105 Loss 0.057767 Duration 71.690 seconds.
Epoch 106 Loss 0.023333 Duration 71.033 seconds.
Epoch 107 Loss 0.006764 Duration 71.232 seconds.
Epoch 108 Loss 0.006563 Duration 71.864 seconds.
Epoch 109 Loss 0.008366 Duration 71.197 seconds.
Epoch 110 Loss 0.038666 Duration 71.615 seconds.
Checkpoint has been created.
Epoch 111 Loss 0.033282 Duration 71.323 seconds.
Epoch 112 Loss 0.014241 Duration 71.073 seconds.
Epoch 113 Loss 0.004235 Duration 71.617 seconds.
Epoch 114 Loss 0.026214 Duration 71.044 seconds.
Epoch 115 Loss 0.006642 Duration 71.495 seconds.
Epoch 116 Loss 0.008620 Duration 71.070 seconds.
Epoch 117 Loss 0.023048 Duration 70.940 seconds.
Epoch 118 Loss 0.010254 Duration 71.353 seconds.
Epoch 119 Loss 0.005688 Duration 70.973 seconds.
Epoch 120 Loss 0.010795 Duration 71.016 seconds.
Checkpoint has been created.
Epoch 121 Loss 0.004849 Duration 71.507 seconds.
Epoch 122 Loss 0.013876 Duration 71.100 seconds.
Epoch 123 Loss 0.034672 Duration 71.634 seconds.
Epoch 124 Loss 0.004636 Duration 70.987 seconds.

```
Epoch 122 Loss 0.013876 Duration 71.100 seconds.
Epoch 123 Loss 0.034672 Duration 71.634 seconds.
Epoch 124 Loss 0.004636 Duration 70.987 seconds.
Epoch 125 Loss 0.003965 Duration 70.923 seconds.
Epoch 126 Loss 0.012413 Duration 71.343 seconds.
Epoch 127 Loss 0.031061 Duration 70.928 seconds.
Epoch 128 Loss 0.015199 Duration 71.547 seconds.
Epoch 129 Loss 0.014797 Duration 71.206 seconds.
Epoch 130 Loss 0.011159 Duration 71.127 seconds.
Checkpoint has been created.
Epoch 131 Loss 0.007991 Duration 71.713 seconds.
Epoch 132 Loss 0.027462 Duration 70.975 seconds.
Epoch 133 Loss 0.005527 Duration 71.523 seconds.
Epoch 134 Loss 0.008001 Duration 71.234 seconds.
Epoch 135 Loss 0.013405 Duration 71.065 seconds.
Epoch 136 Loss 0.012971 Duration 71.441 seconds.
Epoch 137 Loss 0.034632 Duration 70.894 seconds.
Epoch 138 Loss 0.002972 Duration 71.253 seconds.
Epoch 139 Loss 0.004988 Duration 71.071 seconds.
Epoch 140 Loss 0.003208 Duration 70.931 seconds.
Checkpoint has been created.
Epoch 141 Loss 0.001671 Duration 71.380 seconds.
Epoch 142 Loss 0.010502 Duration 71.046 seconds.
Epoch 143 Loss 0.005331 Duration 71.470 seconds.
Epoch 144 Loss 0.001008 Duration 70.866 seconds.
Epoch 145 Loss 0.004124 Duration 71.310 seconds.
Epoch 146 Loss 0.001383 Duration 70.908 seconds.
Epoch 147 Loss 0.010496 Duration 70.988 seconds.
Epoch 148 Loss 0.000715 Duration 71.262 seconds.
Epoch 149 Loss 0.006057 Duration 71.184 seconds.
Epoch 150 Loss 0.000855 Duration 71.286 seconds.
Checkpoint has been created.
Epoch 151 Loss 0.005178 Duration 70.986 seconds.
Epoch 152 Loss 0.046018 Duration 70.901 seconds.
Epoch 153 Loss 0.003578 Duration 71.302 seconds.
Epoch 154 Loss 0.002329 Duration 70.854 seconds.
Epoch 155 Loss 0.004800 Duration 71.259 seconds.
Epoch 156 Loss 0.001151 Duration 70.856 seconds.
Epoch 157 Loss 0.002733 Duration 71.363 seconds.
Epoch 158 Loss 0.004223 Duration 70.878 seconds.
Epoch 159 Loss 0.001864 Duration 70.870 seconds.
Epoch 160 Loss 0.001403 Duration 71.488 seconds.
Checkpoint has been created.
Epoch 161 Loss 0.002852 Duration 70.914 seconds.
Epoch 162 Loss 0.002064 Duration 71.295 seconds.
Epoch 163 Loss 0.003659 Duration 70.926 seconds.
Epoch 164 Loss 0.001053 Duration 70.913 seconds.
Epoch 165 Loss 0.001872 Duration 71.244 seconds.
Epoch 166 Loss 0.003110 Duration 70.955 seconds.
```

```
Epoch 164 Loss 0.001053 Duration 70.913 seconds.
Epoch 165 Loss 0.001872 Duration 71.244 seconds.
Epoch 166 Loss 0.003110 Duration 70.955 seconds.
Epoch 167 Loss 0.000852 Duration 71.390 seconds.
Epoch 168 Loss 0.001663 Duration 70.984 seconds.
Epoch 169 Loss 0.007535 Duration 71.295 seconds.
Epoch 170 Loss 0.003598 Duration 70.908 seconds.
Checkpoint has been created.
Epoch 171 Loss 0.001184 Duration 70.903 seconds.
Epoch 172 Loss 0.002763 Duration 71.223 seconds.
Epoch 173 Loss 0.003899 Duration 70.909 seconds.
Epoch 174 Loss 0.000840 Duration 71.246 seconds.
Epoch 175 Loss 0.001393 Duration 70.918 seconds.
Epoch 176 Loss 0.001730 Duration 71.266 seconds.
Epoch 177 Loss 0.001115 Duration 70.872 seconds.
Epoch 178 Loss 0.006647 Duration 70.854 seconds.
Epoch 179 Loss 0.001647 Duration 71.304 seconds.
Epoch 180 Loss 0.001840 Duration 70.833 seconds.
Checkpoint has been created.
Epoch 181 Loss 0.000887 Duration 71.352 seconds.
Epoch 182 Loss 0.000438 Duration 70.688 seconds.
Epoch 183 Loss 0.005247 Duration 70.952 seconds.
Epoch 184 Loss 0.007510 Duration 70.645 seconds.
Epoch 185 Loss 0.002836 Duration 70.884 seconds.
Epoch 186 Loss 0.002090 Duration 70.693 seconds.
Epoch 187 Loss 0.001660 Duration 70.519 seconds.
Epoch 188 Loss 0.007522 Duration 71.087 seconds.
Epoch 189 Loss 0.000884 Duration 70.788 seconds.
Epoch 190 Loss 0.000456 Duration 70.919 seconds.
Checkpoint has been created.
Epoch 191 Loss 0.001388 Duration 70.664 seconds.
Epoch 192 Loss 0.001209 Duration 71.267 seconds.
Epoch 193 Loss 0.002226 Duration 70.893 seconds.
Epoch 194 Loss 0.002910 Duration 71.241 seconds.
Epoch 195 Loss 0.001566 Duration 71.126 seconds.
Epoch 196 Loss 0.001922 Duration 70.797 seconds.
Epoch 197 Loss 0.002466 Duration 71.892 seconds.
Epoch 198 Loss 0.004587 Duration 70.873 seconds.
Epoch 199 Loss 0.001401 Duration 71.067 seconds.
Epoch 200 Loss 0.000655 Duration 70.402 seconds.
Checkpoint has been created.
```

Version 2:

```
--- Preparing Data ---
tcmalloc: large alloc 1228800000 bytes == 0x1c072000 @
0x7f14939901e7 0x7f14912e10ce 0x7f1491337cf5 0x7f14913e086d
0x7f14913e117f 0x7f14913e12d0 0x4ba72b 0x7f1491322944 0x58f67f
0x50ff13 0x5b4ee6 0x58ff2e 0x50d482 0x58fd37 0x50c4fc 0x58fd37
0x50c4fc 0x5b4ee6 0x6005a3 0x607796 0x60785c 0x60a436 0x64db82
0x64dd2e 0x7f149358dc87 0x5b636a
Using cuda device
### Training... ###
Epoch 1 Loss 1.675681 Duration 122.146 seconds.
Epoch 2 Loss 1.358612 Duration 118.026 seconds.
Epoch 3 Loss 0.907389 Duration 118.401 seconds.
Epoch 4 Loss 0.820122 Duration 117.416 seconds.
Epoch 5 Loss 0.978654 Duration 118.228 seconds.
Epoch 6 Loss 0.768838 Duration 118.142 seconds.
Epoch 7 Loss 0.618724 Duration 118.196 seconds.
Epoch 8 Loss 0.447273 Duration 118.449 seconds.
Epoch 9 Loss 0.731959 Duration 118.343 seconds.
Epoch 10 Loss 0.628646 Duration 118.160 seconds.
Checkpoint has been created.
Epoch 11 Loss 0.508103 Duration 118.434 seconds.
Epoch 12 Loss 0.427843 Duration 118.400 seconds.
Epoch 13 Loss 0.486423 Duration 118.172 seconds.
Epoch 14 Loss 0.283727 Duration 118.134 seconds.
Epoch 15 Loss 0.614309 Duration 118.377 seconds.
Epoch 16 Loss 0.495796 Duration 118.100 seconds.
Epoch 17 Loss 0.360788 Duration 118.282 seconds.
Epoch 18 Loss 0.422807 Duration 118.248 seconds.
Epoch 19 Loss 0.402745 Duration 118.481 seconds.
Epoch 20 Loss 0.400654 Duration 117.758 seconds.
Checkpoint has been created.
Epoch 21 Loss 0.423809 Duration 119.264 seconds.
Epoch 22 Loss 0.468240 Duration 119.530 seconds.
Epoch 23 Loss 0.441965 Duration 118.399 seconds.
Epoch 24 Loss 0.303660 Duration 117.878 seconds.
Epoch 25 Loss 0.295388 Duration 118.397 seconds.
Epoch 26 Loss 0.348446 Duration 117.967 seconds.
Epoch 27 Loss 0.314136 Duration 117.973 seconds.
Epoch 28 Loss 0.272812 Duration 117.957 seconds.
Epoch 29 Loss 0.339939 Duration 117.861 seconds.
Epoch 30 Loss 0.255720 Duration 117.600 seconds.
Checkpoint has been created.
Epoch 31 Loss 0.339771 Duration 118.403 seconds.
Epoch 32 Loss 0.319767 Duration 117.445 seconds.
Epoch 33 Loss 0.308960 Duration 117.494 seconds.
Epoch 34 Loss 0.293013 Duration 117.757 seconds.
Epoch 35 Loss 0.254420 Duration 117.714 seconds.
Epoch 36 Loss 0.250413 Duration 117.739 seconds.
Epoch 37 Loss 0.204139 Duration 117.711 seconds.
Epoch 38 Loss 0.254563 Duration 117.612 seconds.
Epoch 39 Loss 0.266242 Duration 117.674 seconds.
Epoch 40 Loss 0.421805 Duration 117.347 seconds.
Checkpoint has been created.
Epoch 41 Loss 0.294483 Duration 117.495 seconds.
```

```
Checkpoint has been created.
Epoch 41 Loss 0.294483 Duration 117.495 seconds.
Epoch 42 Loss 0.315100 Duration 117.934 seconds.
Epoch 43 Loss 0.441399 Duration 117.716 seconds.
Epoch 44 Loss 0.350575 Duration 117.833 seconds.
Epoch 45 Loss 0.346020 Duration 117.565 seconds.
Epoch 46 Loss 0.535848 Duration 117.549 seconds.
Epoch 47 Loss 0.276930 Duration 117.329 seconds.
Epoch 48 Loss 0.397458 Duration 117.389 seconds.
Epoch 49 Loss 0.313786 Duration 117.683 seconds.
Epoch 50 Loss 0.310321 Duration 117.895 seconds.
Checkpoint has been created.
Epoch 51 Loss 0.248109 Duration 118.261 seconds.
Epoch 52 Loss 0.149022 Duration 117.975 seconds.
Epoch 53 Loss 0.238293 Duration 118.342 seconds.
Epoch 54 Loss 0.382980 Duration 117.873 seconds.
Epoch 55 Loss 0.183293 Duration 118.589 seconds.
Epoch 56 Loss 0.255850 Duration 118.376 seconds.
Epoch 57 Loss 0.344008 Duration 118.297 seconds.
Epoch 58 Loss 0.294776 Duration 118.277 seconds.
Epoch 59 Loss 0.209932 Duration 118.224 seconds.
Epoch 60 Loss 0.359829 Duration 118.092 seconds.
Checkpoint has been created.
Epoch 61 Loss 0.315518 Duration 117.621 seconds.
Epoch 62 Loss 0.198196 Duration 117.499 seconds.
Epoch 63 Loss 0.229297 Duration 117.364 seconds.
Epoch 64 Loss 0.288755 Duration 117.875 seconds.
Epoch 65 Loss 0.382836 Duration 117.815 seconds.
Epoch 66 Loss 0.163099 Duration 117.630 seconds.
Epoch 67 Loss 0.249302 Duration 117.501 seconds.
Epoch 68 Loss 0.320657 Duration 117.458 seconds.
Epoch 69 Loss 0.244948 Duration 116.903 seconds.
Epoch 70 Loss 0.246495 Duration 117.379 seconds.
Checkpoint has been created.
Epoch 71 Loss 0.283090 Duration 117.500 seconds.
Epoch 72 Loss 0.241911 Duration 117.911 seconds.
Epoch 73 Loss 0.217461 Duration 117.581 seconds.
Epoch 74 Loss 0.297773 Duration 118.191 seconds.
Epoch 75 Loss 0.356912 Duration 117.445 seconds.
Epoch 76 Loss 0.260260 Duration 117.736 seconds.
Epoch 77 Loss 0.251044 Duration 117.295 seconds.
Epoch 78 Loss 0.268691 Duration 117.832 seconds.
Epoch 79 Loss 0.386179 Duration 117.485 seconds.
Epoch 80 Loss 0.326096 Duration 118.205 seconds.
Checkpoint has been created.
Epoch 81 Loss 0.337248 Duration 117.353 seconds.
Epoch 82 Loss 0.320138 Duration 117.656 seconds.
Epoch 83 Loss 0.247501 Duration 117.632 seconds.
Epoch 84 Loss 0.408144 Duration 117.378 seconds.
Epoch 85 Loss 0.274050 Duration 117.717 seconds.
Epoch 86 Loss 0.248042 Duration 117.225 seconds.
Epoch 87 Loss 0.213529 Duration 117.398 seconds.
Epoch 88 Loss 0.304530 Duration 117.378 seconds.
Epoch 89 Loss 0.204641 Duration 117.262 seconds.
```

```
Epoch 68 Loss 0.320657 Duration 117.458 seconds.
Epoch 69 Loss 0.244948 Duration 116.903 seconds.
Epoch 70 Loss 0.246495 Duration 117.379 seconds.
Checkpoint has been created.
Epoch 71 Loss 0.283090 Duration 117.500 seconds.
Epoch 72 Loss 0.241911 Duration 117.911 seconds.
Epoch 73 Loss 0.217461 Duration 117.581 seconds.
Epoch 74 Loss 0.297773 Duration 118.191 seconds.
Epoch 75 Loss 0.356912 Duration 117.445 seconds.
Epoch 76 Loss 0.260260 Duration 117.736 seconds.
Epoch 77 Loss 0.251044 Duration 117.295 seconds.
Epoch 78 Loss 0.268691 Duration 117.832 seconds.
Epoch 79 Loss 0.386179 Duration 117.485 seconds.
Epoch 80 Loss 0.326096 Duration 118.205 seconds.
Checkpoint has been created.
Epoch 81 Loss 0.337248 Duration 117.353 seconds.
Epoch 82 Loss 0.320138 Duration 117.656 seconds.
Epoch 83 Loss 0.247501 Duration 117.632 seconds.
Epoch 84 Loss 0.408144 Duration 117.378 seconds.
Epoch 85 Loss 0.274050 Duration 117.717 seconds.
Epoch 86 Loss 0.248042 Duration 117.225 seconds.
Epoch 87 Loss 0.213529 Duration 117.398 seconds.
Epoch 88 Loss 0.304530 Duration 117.378 seconds.
Epoch 89 Loss 0.204641 Duration 117.262 seconds.
Epoch 90 Loss 0.318828 Duration 117.458 seconds.
Checkpoint has been created.
Epoch 91 Loss 0.119399 Duration 117.091 seconds.
Epoch 92 Loss 0.287731 Duration 117.806 seconds.
Epoch 93 Loss 0.332155 Duration 117.467 seconds.
Epoch 94 Loss 0.252301 Duration 117.691 seconds.
Epoch 95 Loss 0.070960 Duration 117.221 seconds.
Epoch 96 Loss 0.132312 Duration 117.502 seconds.
Epoch 97 Loss 0.033834 Duration 117.333 seconds.
Epoch 98 Loss 0.042253 Duration 117.959 seconds.
Epoch 99 Loss 0.028994 Duration 117.176 seconds.
Epoch 100 Loss 0.081678 Duration 117.660 seconds.
Checkpoint has been created.
Epoch 101 Loss 0.094440 Duration 117.072 seconds.
Epoch 102 Loss 0.050345 Duration 118.005 seconds.
Epoch 103 Loss 0.069376 Duration 117.131 seconds.
Epoch 104 Loss 0.028610 Duration 116.699 seconds.
Epoch 105 Loss 0.048618 Duration 116.855 seconds.
Epoch 106 Loss 0.036088 Duration 117.383 seconds.
Epoch 107 Loss 0.013446 Duration 117.255 seconds.
Epoch 108 Loss 0.017841 Duration 117.322 seconds.
Epoch 109 Loss 0.010052 Duration 117.215 seconds.
Epoch 110 Loss 0.040885 Duration 117.432 seconds.
Checkpoint has been created.
Epoch 111 Loss 0.016503 Duration 117.224 seconds.
Epoch 112 Loss 0.024681 Duration 117.998 seconds.
Epoch 113 Loss 0.012640 Duration 117.457 seconds.
Epoch 114 Loss 0.025675 Duration 117.609 seconds.
Epoch 115 Loss 0.020395 Duration 117.358 seconds.
```

```
Epoch 97 Loss 0.070009 Duration 120.322 seconds.
Epoch 98 Loss 0.039141 Duration 120.188 seconds.
Epoch 99 Loss 0.067419 Duration 119.936 seconds.
Epoch 100 Loss 0.089066 Duration 120.245 seconds.
Checkpoint has been created.
Epoch 101 Loss 0.055996 Duration 119.978 seconds.
Epoch 102 Loss 0.045447 Duration 119.814 seconds.
Epoch 103 Loss 0.020437 Duration 119.892 seconds.
Epoch 104 Loss 0.020245 Duration 119.827 seconds.
Epoch 105 Loss 0.007162 Duration 119.441 seconds.
Epoch 106 Loss 0.078248 Duration 119.837 seconds.
Epoch 107 Loss 0.061933 Duration 119.860 seconds.
Epoch 108 Loss 0.044369 Duration 119.592 seconds.
Epoch 109 Loss 0.031180 Duration 119.750 seconds.
Epoch 110 Loss 0.005670 Duration 120.093 seconds.
Checkpoint has been created.
Epoch 111 Loss 0.012814 Duration 119.858 seconds.
Epoch 112 Loss 0.033699 Duration 119.692 seconds.
Epoch 113 Loss 0.020911 Duration 119.969 seconds.
Epoch 114 Loss 0.009984 Duration 119.964 seconds.
Epoch 115 Loss 0.011757 Duration 119.600 seconds.
Epoch 116 Loss 0.013819 Duration 119.743 seconds.
Epoch 117 Loss 0.004775 Duration 119.707 seconds.
Epoch 118 Loss 0.010550 Duration 119.879 seconds.
Epoch 119 Loss 0.015565 Duration 119.692 seconds.
Epoch 120 Loss 0.032362 Duration 119.791 seconds.
Checkpoint has been created.
Epoch 121 Loss 0.019400 Duration 119.971 seconds.

Epoch 122 Loss 0.008926 Duration 119.817 seconds.
```