1. Install JIT compiler and load the CUDA / C++ extension into python:

```
from torch.utils.cpp_extension import load

ScCudaTorch = load(
  name="sc_cuda_torch",
  sources=["ScCudaTorch_2.cpp", "ScCudaTorch_UniPinConst_2.cu"],
  verbose=True
)
```

Use "ScCudaTorch_2.cpp", "ScCudaTorch_UniPinConst_2.cu" for Unipolar functions (Check files CIFAR_VGG9/ScCudaTorchExtension_VGG9_2.py or MNIST_LeNet5/ScCudaTorchExtension_Unipolars.py) or "ScCudaTorch.cpp", "ScCudaTorch.cu" for Bipolar functions. (Check files MNIST_LeNet5/ScCudaTorchExtension_Bipolars.py or CIFAR_VGG9/ScCudaTorchExtension_VGG9_Bipo_1.py)

2. Define the type of RNG from these possibilities:

```
enum RandomGeneratorType {
  MT19937,
  LFSR_16,
  PCG,
  XORSHIFT
};

randomNumberGenType = ScCudaTorch.RandomGeneratorType
rng_type = randomNumberGenType.MT19937
```

3. Use Sc_Conv2_py() and ScCudaTorch.ScCudaFcLayer() functions for the SC 2d convolution and SC fc layer respectively.
4. Inside Sc_Conv2_py() it may be necessary to modify the lines:

```
output_tensor = torch.zeros(output_channels, height - 2, width - 2)  # Output dimensions after 3x3 convolution
number_Accumulations = 3*3*input_channels
output = torch.zeros(height - 2, width - 2)  # Temporary output for each output channel
```

depending on the size of the convolutional kernel. In MNIST_LeNet5 a 5x5 kernel is used and in CIFAR_VGG9 a 3x3 kernel is used.

5. Build the SC NN inside a **class ScNet(nn.Module):** with an initializer: **def __init__()** followed by the SC forward propagation: **def forward(self, original_image).**
6. After that, define the normal NN: **class SCCNN9(nn.Module)** and create the model instance:

```
sccnn9 = SCCNN9(num_classes=10)
PATH = './cifar_sccnn9_67.pth'
sccnn9.load_state_dict(torch.load(PATH))
```

7. After that create the SC parameters and input them in a SC NN instance: **scNet = ScNet()**
8. CIFAR_VGG9/ScCudaTorchExtension_VGG9_2.py and CIFAR_VGG9/ScCudaTorchExtension_VGG9_Bipo_1.py have an example on how to create a SC VGG9 and infer 30 images from the class 'dog' of the CIFAR-10 dataset.
9. Important: When downloading the test images, it is important to change the batch_size to 1:

```
testloader = torch.utils.data.DataLoader(testset, batch_size=1,  shuffle=False, num_workers=1)
```

10. In the same way, both MNIST_LeNet5/customNN_MNIST.py and MNIST_LeNet5/customNN_MNIST_Bipolars.py have an example on how to create a SC LeNet5 and infer 100 images from the class 'Sneaker' of the fashionMNIST dataset.
11. MNIST_LeNet5/ScCudaTorchExtension_Bipolars.py and MNIST_LeNet5/ScCudaTorchExtension_Unipolars.py do the same but printing the similarities / errors of every layer (layerwise).
12. The CPU functions are in ScTorch.cpp and ScTorch.h.