CIFAR 10

**Preliminary installations**

1. !pip3 install torch torchvision

Requirement already satisfied: torch in /usr/local/lib/python3.6/dist-packages (1.0.0) Requirement already satisfied: torchvision in /usr/local/lib/python3.6/dist-packages (0.2.1) Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.6/dist-packages (from torchvision) (5.4.1) Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from torchvision) (1.14.6) Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from torchvision) (1.11.0)

**Importing Relevant Libraries**

1. import torch
2. import matplotlib.pyplot as plt
3. import numpy as np
4. import torch.nn.functional as F
5. from torch import nn
6. from torchvision import datasets, transforms

**Initializing GPU Usage**

1. device = torch.device("cuda:0" if torch.cuda.is\_available() else "cpu")

**Transformations and Data Augmentation**

1. transform\_train = transforms.Compose([transforms.Resize((32,32)),
2. transforms.RandomHorizontalFlip(),
3. transforms.RandomRotation(10),
4. transforms.RandomAffine(0, shear=10, scale=(0.8,1.2)),
5. transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2),
6. transforms.ToTensor(),
7. transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
8. ])

11. transform = transforms.Compose([transforms.Resize((32,32)),
12. transforms.ToTensor(),
13. transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
14. ])
15. training\_dataset = datasets.CIFAR10(root='./data', train=True, download=True, transform=transform\_train)
16. validation\_dataset = datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
18. training\_loader = torch.utils.data.DataLoader(training\_dataset, batch\_size=100, shuffle=True)
19. validation\_loader = torch.utils.data.DataLoader(validation\_dataset, batch\_size = 100, shuffle=False)

**NumPy Image Conversion Function**

1. def im\_convert(tensor):
2. image = tensor.clone().detach().numpy()
3. image = image.transpose(1, 2, 0)
4. image = image \* np.array((0.5, 0.5, 0.5)) + np.array((0.5, 0.5, 0.5))
5. image = image.clip(0, 1)
6. return image

**Defining Classes**

1. classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

**Plotting Image Data**

1. dataiter = iter(training\_loader)
2. images, labels = dataiter.next()
3. fig = plt.figure(figsize=(25, 4))
5. for idx in np.arange(20):
6. ax = fig.add\_subplot(2, 10, idx+1, xticks=[], yticks=[])
7. plt.imshow(im\_convert(images[idx]))
8. ax.set\_title(classes[labels[idx].item()])



**Model Class Constructor**

1. class LeNet(nn.Module):
2. def \_\_init\_\_(self):
3. super().\_\_init\_\_()
4. self.conv1 = nn.Conv2d(3, 16, 3, 1, padding=1)
5. self.conv2 = nn.Conv2d(16, 32, 3, 1, padding=1)
6. self.conv3 = nn.Conv2d(32, 64, 3, 1, padding=1)
7. self.fc1 = nn.Linear(4\*4\*64, 500)
8. self.dropout1 = nn.Dropout(0.5)
9. self.fc2 = nn.Linear(500, 10)
10. def forward(self, x):
11. x = F.relu(self.conv1(x))
12. x = F.max\_pool2d(x, 2, 2)
13. x = F.relu(self.conv2(x))
14. x = F.max\_pool2d(x, 2, 2)
15. x = F.relu(self.conv3(x))
16. x = F.max\_pool2d(x, 2, 2)
17. x = x.view(-1, 4\*4\*64)
18. x = F.relu(self.fc1(x))
19. x = self.dropout1(x)
20. x = self.fc2(x)
21. return x

**Declaring Classifier**

1. model = LeNet().to(device)
2. model

LeNet(   (conv1): Conv2d(3, 16, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))   (conv2): Conv2d(16, 32, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))   (conv3): Conv2d(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))   (fc1): Linear(in\_features=1024, out\_features=500, bias=True)   (dropout1): Dropout(p=0.5)   (fc2): Linear(in\_features=500, out\_features=10, bias=True) )

**Loss and Optimizer**

1. criterion = nn.CrossEntropyLoss()
2. optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)

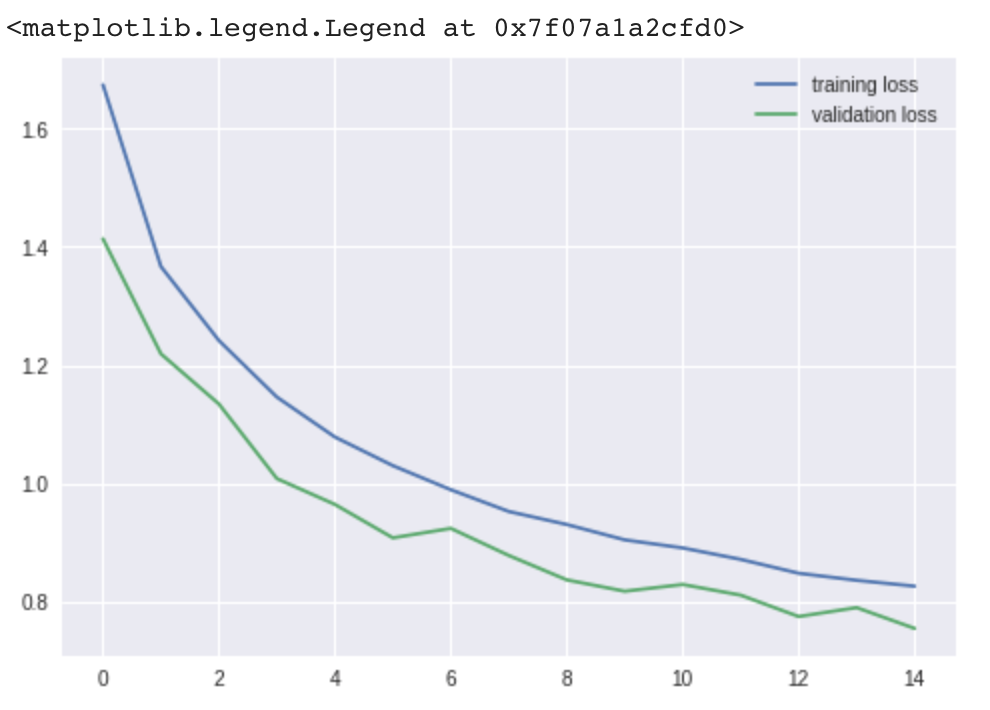
**Training Process**

1. epochs = 15
2. running\_loss\_history = []
3. running\_corrects\_history = []
4. val\_running\_loss\_history = []
5. val\_running\_corrects\_history = []
7. for e in range(epochs):
9. running\_loss = 0.0
10. running\_corrects = 0.0
11. val\_running\_loss = 0.0
12. val\_running\_corrects = 0.0
14. for inputs, labels in training\_loader:
15. inputs = inputs.to(device)
16. labels = labels.to(device)
17. outputs = model(inputs)
18. loss = criterion(outputs, labels)
20. optimizer.zero\_grad()
21. loss.backward()
22. optimizer.step()
24. \_, preds = torch.max(outputs, 1)
25. running\_loss += loss.item()
26. running\_corrects += torch.sum(preds == labels.data)
28. else:
29. with torch.no\_grad():
30. for val\_inputs, val\_labels in validation\_loader:
31. val\_inputs = val\_inputs.to(device)
32. val\_labels = val\_labels.to(device)
33. val\_outputs = model(val\_inputs)
34. val\_loss = criterion(val\_outputs, val\_labels)
36. \_, val\_preds = torch.max(val\_outputs, 1)
37. val\_running\_loss += val\_loss.item()
38. val\_running\_corrects += torch.sum(val\_preds == val\_labels.data)
40. epoch\_loss = running\_loss/len(training\_loader)
41. epoch\_acc = running\_corrects.float()/ len(training\_loader)
42. running\_loss\_history.append(epoch\_loss)
43. running\_corrects\_history.append(epoch\_acc)
45. val\_epoch\_loss = val\_running\_loss/len(validation\_loader)
46. val\_epoch\_acc = val\_running\_corrects.float()/ len(validation\_loader)
47. val\_running\_loss\_history.append(val\_epoch\_loss)
48. val\_running\_corrects\_history.append(val\_epoch\_acc)
49. print('epoch :', (e+1))
50. print('training loss: {:.4f}, acc {:.4f} '.format(epoch\_loss, epoch\_acc.item()))
51. print('validation loss: {:.4f}, validation acc {:.4f} '.format(val\_epoch\_loss, val\_epoch\_acc.item()))

Output:epoch : 1 training loss: 1.6735, acc 38.6500  validation loss: 1.4134, validation acc 49.6200  epoch : 2 training loss: 1.3667, acc 50.6620  validation loss: 1.2193, validation acc 56.1000  epoch : 3 training loss: 1.2418, acc 55.6380  validation loss: 1.1346, validation acc 59.2100  epoch : 4 training loss: 1.1463, acc 59.2120  validation loss: 1.0087, validation acc 64.0800  epoch : 5 training loss: 1.0789, acc 61.7960  validation loss: 0.9651, validation acc 65.8000  epoch : 6 training loss: 1.0305, acc 63.6680  validation loss: 0.9086, validation acc 68.1100  epoch : 7 training loss: 0.9899, acc 65.2080  validation loss: 0.9246, validation acc 67.4600  epoch : 8 training loss: 0.9531, acc 66.5540  validation loss: 0.8789, validation acc 69.8900  epoch : 9 training loss: 0.9312, acc 67.4380  validation loss: 0.8377, validation acc 70.9100  epoch : 10 training loss: 0.9052, acc 68.3520  validation loss: 0.8185, validation acc 71.5300  epoch : 11 training loss: 0.8916, acc 68.7220  validation loss: 0.8299, validation acc 71.4400  epoch : 12 training loss: 0.8723, acc 69.3520  validation loss: 0.8119, validation acc 71.8900  epoch : 13 training loss: 0.8489, acc 70.3820  validation loss: 0.7758, validation acc 73.1300  epoch : 14 training loss: 0.8369, acc 70.7560  validation loss: 0.7909, validation acc 72.6500  epoch : 15 training loss: 0.8270, acc 71.0360  validation loss: 0.7557, validation acc 73.5500

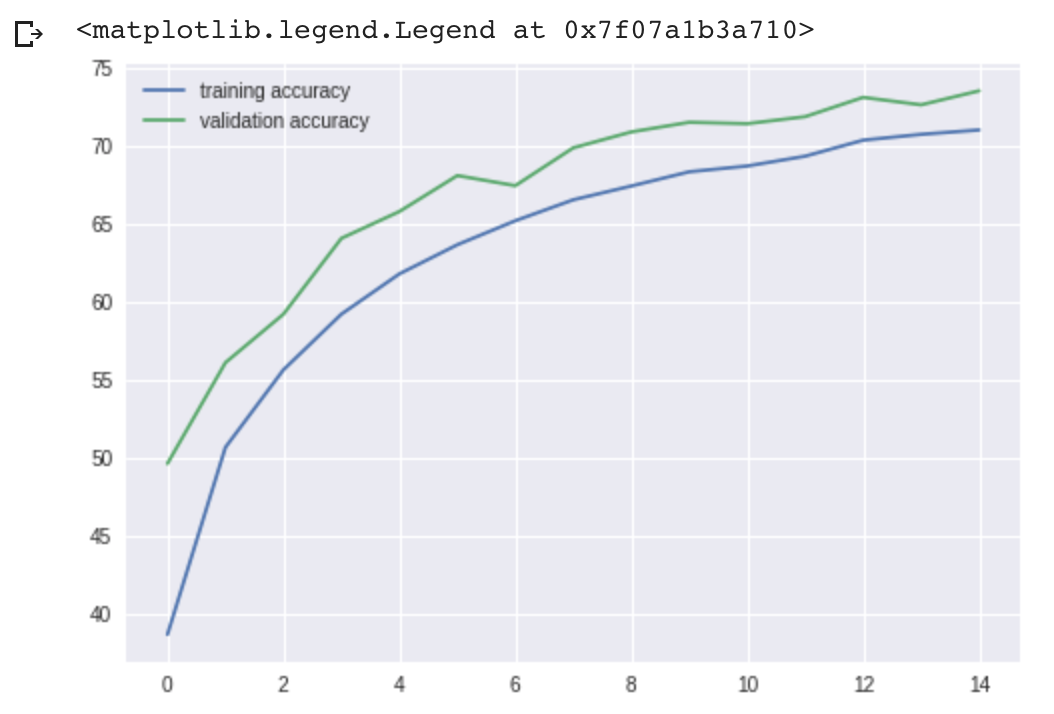
**Plotting Training and Validation Loss**

1. plt.plot(running\_loss\_history, label='training loss')
2. plt.plot(val\_running\_loss\_history, label='validation loss')
3. plt.legend()



**Plotting Accuracy**

1. plt.plot(running\_corrects\_history, label='training accuracy')
2. plt.plot(val\_running\_corrects\_history, label='validation accuracy')
3. plt.legend()



**Pillow Installation**

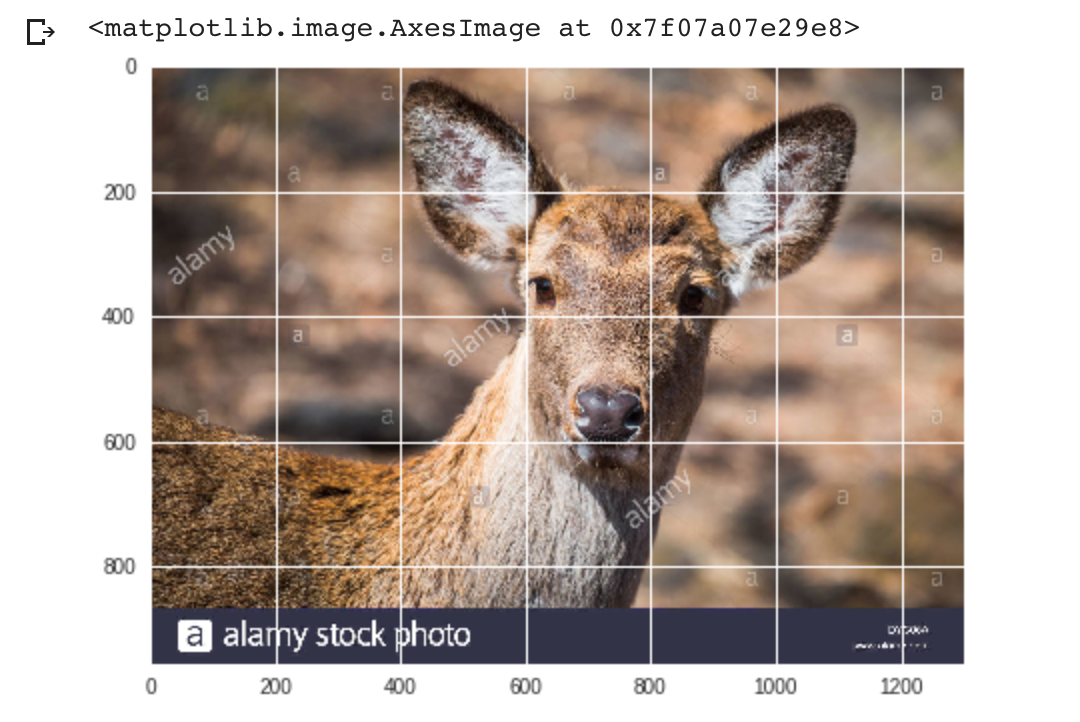
1. !pip3 install pillow==4.0.0

Collecting pillow==4.0.0   Using cached<https://files.pythonhosted.org/packages/37/e8/b3fbf87b0188d22246678f8cd61e23e31caa1769ebc06f1664e2e5fe8a17/Pillow-4.0.0-cp36-cp36m-manylinux1_x86_64.whl> Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-packages (from pillow==4.0.0) (0.46) torchvision 0.2.1 has requirement pillow>=4.1.1, but you'll have pillow 4.0.0 which is incompatible. Installing collected packages: pillow   Found existing installation: Pillow 5.4.1     Uninstalling Pillow-5.4.1:       Successfully uninstalled Pillow-5.4.1 Successfully installed pillow-4.0.0

1. import PIL.ImageOps

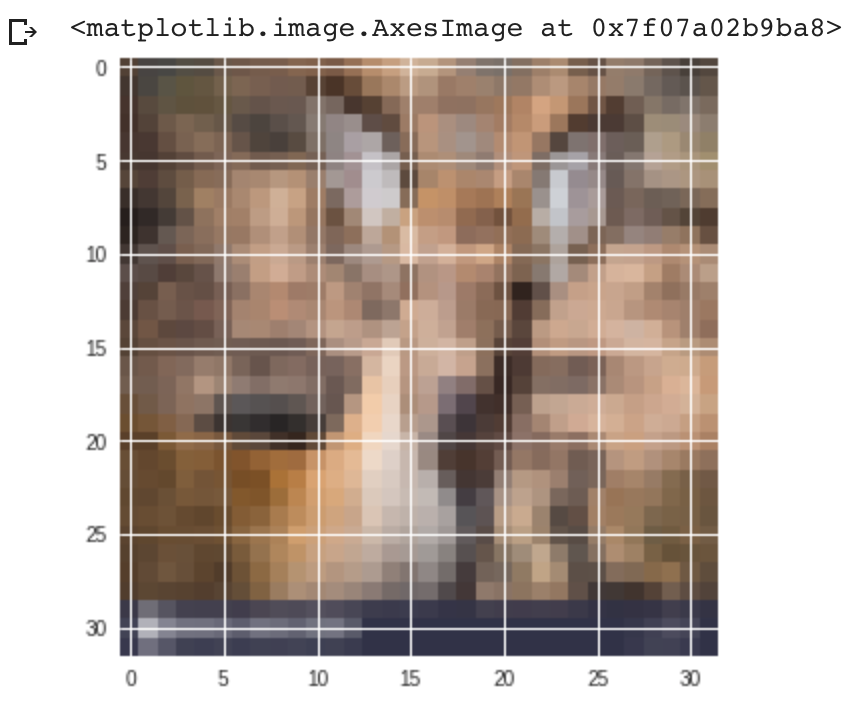
**Show Image**

1. import requests
2. from PIL import Image
4. url = 'https://c8.alamy.com/comp/DYC06A/hornless-reindeer-at-zoo-DYC06A.jpg'
5. response = requests.get(url, stream = True)
6. img = Image.open(response.raw)
7. plt.imshow(img)



**Pre-processing Image**

1. img = transform(img)
2. plt.imshow(im\_convert(img))



**Prediction**

1. image = img.to(device).unsqueeze(0)
2. output = model(image)
3. \_, pred = torch.max(output, 1)
4. print(classes[pred.item()])



**Validation Prediction**

1. dataiter = iter(validation\_loader)
2. images, labels = dataiter.next()
3. images = images.to(device)
4. labels = labels.to(device)
5. output = model(images)
6. \_, preds = torch.max(output, 1)
8. fig = plt.figure(figsize=(25, 4))
10. for idx in np.arange(20):
11. ax = fig.add\_subplot(2, 10, idx+1, xticks=[], yticks=[])
12. plt.imshow(im\_convert(images[idx]))
13. ax.set\_title("{} ({})".format(str(classes[preds[idx].item()]), str(classes[labels[idx].item()])), color=("green" if preds[idx]==labels[idx] else "red"))



**Feel free to ignore the following code. It was mainly used for debugging purposes**

##   NO\_CAFFE2\_OPS#     disable Caffe2 operators build##   USE\_GLOO\_IBVERBS#     toggle features related to distributed support##   USE\_OPENCV#     enables use of OpenCV for additional operators##   USE\_FFMPEG#     enables use of ffmpeg for additional operators##   USE\_LEVELDB#     enables use of LevelDB for storage##   USE\_LMDB#     enables use of LMDB for storage##   BUILD\_BINARY#     enables the additional binaries/ build##   PYTORCH\_BUILD\_VERSION#   PYTORCH\_BUILD\_NUMBER#     specify the version of PyTorch, rather than the hard-coded version#     in this file; used when we're building binaries for distribution##   TORCH\_CUDA\_ARCH\_LIST#     specify which CUDA architectures to build for.#     ie `TORCH\_CUDA\_ARCH\_LIST="6.0;7.0"`#     These are not CUDA versions, instead, they specify what#     classes of NVIDIA hardware we should generate PTX for.##   ONNX\_NAMESPACE#     specify a namespace for ONNX built here rather than the hard-coded#     one in this file; needed to build with other frameworks that share ONNX.##   BLAS#     BLAS to be used by Caffe2. Can be MKL, Eigen, ATLAS, or OpenBLAS. If set#     then the build will fail if the requested BLAS is not found, otherwise#     the BLAS will be chosen based on what is found on your system.##   USE\_FBGEMM#     Enables use of FBGEMM##   USE\_REDIS#     Whether to use Redis for distributed workflows (Linux only)##   USE\_ZSTD#     Enables use of ZSTD, if the libraries are found## Environment variables we respect (these environment variables are# conventional and are often understood/set by other software.)##   CUDA\_HOME (Linux/OS X)#   CUDA\_PATH (Windows)#     specify where CUDA is installed; usually /usr/local/cuda or#     /usr/local/cuda-x.y#   CUDAHOSTCXX#     specify a different compiler than the system one to use as the CUDA#     host compiler for nvcc.##   CUDA\_NVCC\_EXECUTABLE#     Specify a NVCC to use. This is used in our CI to point to a cached nvcc##   CUDNN\_LIB\_DIR#   CUDNN\_INCLUDE\_DIR#   CUDNN\_LIBRARY#     specify where cuDNN is installed##   MIOPEN\_LIB\_DIR#   MIOPEN\_INCLUDE\_DIR#   MIOPEN\_LIBRARY#     specify where MIOpen is installed##   NCCL\_ROOT\_DIR#   NCCL\_LIB\_DIR#   NCCL\_INCLUDE\_DIR#     specify where nccl is installed##   NVTOOLSEXT\_PATH (Windows only)#     specify where nvtoolsext is installed##   LIBRARY\_PATH#   LD\_LIBRARY\_PATH#     we will search for libraries in these paths from \_\_future\_\_ import print\_functionfrom setuptools import setup, Extension, distutils, Command, find\_packagesimport setuptools.command.build\_extimport setuptools.command.installimport setuptools.command.developimport setuptools.command.build\_pyimport distutils.unixccompilerimport distutils.command.buildimport distutils.command.cleanimport distutils.sysconfigimport filecmpimport platformimport subprocessimport shutilimport multiprocessingimport sysimport osimport jsonimport globimport importlib # If you want to modify flags or environmental variables that is set when# building torch, you should do it in tools/setup\_helpers/configure.py.# Please don't add it here unless it's only used in PyTorch.from tools.setup\_helpers.configure import \*from tools.setup\_helpers.generate\_code import generate\_codefrom tools.setup\_helpers.ninja\_builder import NinjaBuilder, ninja\_build\_extimport tools.setup\_helpers.configure ################################################################################# Parameters parsed from environment################################################################################ VERBOSE\_SCRIPT = True# see if the user passed a quiet flag to setup.py arguments and respect# that in our parts of the buildfor arg in sys.argv:    if arg == "--":        break    if arg == '-q' or arg == '--quiet':        VERBOSE\_SCRIPT = False if VERBOSE\_SCRIPT:    def report(\*args):        print(\*args)else:    def report(\*args):        pass # Constant known variables used throughout this filecwd = os.path.dirname(os.path.abspath(\_\_file\_\_))lib\_path = os.path.join(cwd, "torch", "lib")third\_party\_path = os.path.join(cwd, "third\_party")tmp\_install\_path = lib\_path + "/tmp\_install"caffe2\_build\_dir = os.path.join(cwd, "build")# lib/pythonx.x/site-packagesrel\_site\_packages = distutils.sysconfig.get\_python\_lib(prefix='')# full absolute path to the dir abovefull\_site\_packages = distutils.sysconfig.get\_python\_lib()# CMAKE: full path to python libraryif IS\_WINDOWS:    cmake\_python\_library = "{}/libs/python{}.lib".format(        distutils.sysconfig.get\_config\_var("prefix"),        distutils.sysconfig.get\_config\_var("VERSION"))else:    cmake\_python\_library = "{}/{}".format(        distutils.sysconfig.get\_config\_var("LIBDIR"),        distutils.sysconfig.get\_config\_var("INSTSONAME"))cmake\_python\_include\_dir = distutils.sysconfig.get\_python\_inc()  class PytorchCommand(setuptools.Command):    """    Base Pytorch command to avoid implementing initialize/finalize\_options in    every subclass    """    user\_options = []     def initialize\_options(self):        pass     def finalize\_options(self):        pass  ################################################################################# Version, create\_version\_file, and package\_name################################################################################package\_name = os.getenv('TORCH\_PACKAGE\_NAME', 'torch')version = '1.1.0a0'if os.getenv('PYTORCH\_BUILD\_VERSION'):    assert os.getenv('PYTORCH\_BUILD\_NUMBER') is not None    build\_number = int(os.getenv('PYTORCH\_BUILD\_NUMBER'))    version = os.getenv('PYTORCH\_BUILD\_VERSION')    if build\_number > 1:        version += '.post' + str(build\_number)else:    try:        sha = subprocess.check\_output(['git', 'rev-parse', 'HEAD'], cwd=cwd).decode('ascii').strip()        version += '+' + sha[:7]    except Exception:        passreport("Building wheel {}-{}".format(package\_name, version))  class create\_version\_file(PytorchCommand):    def run(self):        global version, cwd        report('-- Building version ' + version)        version\_path = os.path.join(cwd, 'torch', 'version.py')        with open(version\_path, 'w') as f:            f.write("\_\_version\_\_ = '{}'\n".format(version))            # NB: This is not 100% accurate, because you could have built the            # library code with DEBUG, but csrc without DEBUG (in which case            # this would claim to be a release build when it's not.)            f.write("debug = {}\n".format(repr(DEBUG)))            f.write("cuda = {}\n".format(repr(CUDA\_VERSION)))  ################################################################################# Building dependent libraries################################################################################ # All libraries that torch could depend ondep\_libs = ['caffe2'] missing\_pydep = '''Missing build dependency: Unable to `import {importname}`.Please install it via `conda install {module}` or `pip install {module}`'''.strip()  def check\_pydep(importname, module):    try:        importlib.import\_module(importname)    except ImportError:        raise RuntimeError(missing\_pydep.format(importname=importname, module=module))  # Calls build\_pytorch\_libs.sh/bat with the correct env variablesdef build\_libs(libs):    for lib in libs:        assert lib in dep\_libs, 'invalid lib: {}'.format(lib)    if IS\_WINDOWS:        build\_libs\_cmd = ['tools\\build\_pytorch\_libs.bat']    else:        build\_libs\_cmd = ['bash', os.path.join('..', 'tools', 'build\_pytorch\_libs.sh')]     my\_env, extra\_flags = get\_pytorch\_env\_with\_flags()    build\_libs\_cmd.extend(extra\_flags)    my\_env["PYTORCH\_PYTHON\_LIBRARY"] = cmake\_python\_library    my\_env["PYTORCH\_PYTHON\_INCLUDE\_DIR"] = cmake\_python\_include\_dir    my\_env["PYTORCH\_BUILD\_VERSION"] = version     cmake\_prefix\_path = full\_site\_packages    if "CMAKE\_PREFIX\_PATH" in my\_env:        cmake\_prefix\_path = my\_env["CMAKE\_PREFIX\_PATH"] + ";" + cmake\_prefix\_path    my\_env["CMAKE\_PREFIX\_PATH"] = cmake\_prefix\_path     if VERBOSE\_SCRIPT:        my\_env['VERBOSE\_SCRIPT'] = '1'    try:        os.mkdir('build')    except OSError:        pass     kwargs = {'cwd': 'build'} if not IS\_WINDOWS else {}     if subprocess.call(build\_libs\_cmd + libs, env=my\_env, \*\*kwargs) != 0:        report("Failed to run '{}'".format(' '.join(build\_libs\_cmd + libs)))        sys.exit(1)  # Build all dependent librariesclass build\_deps(PytorchCommand):    def run(self):        report('setup.py::build\_deps::run()')        # Check if you remembered to check out submodules         def check\_file(f):            if not os.path.exists(f):                report("Could not find {}".format(f))                report("Did you run 'git submodule update --init --recursive'?")                sys.exit(1)         check\_file(os.path.join(third\_party\_path, "gloo", "CMakeLists.txt"))        check\_file(os.path.join(third\_party\_path, "pybind11", "CMakeLists.txt"))        check\_file(os.path.join(third\_party\_path, 'cpuinfo', 'CMakeLists.txt'))        check\_file(os.path.join(third\_party\_path, 'onnx', 'CMakeLists.txt'))        check\_file(os.path.join(third\_party\_path, 'QNNPACK', 'CMakeLists.txt'))        check\_file(os.path.join(third\_party\_path, 'fbgemm', 'CMakeLists.txt'))         check\_pydep('yaml', 'pyyaml')        check\_pydep('typing', 'typing')         libs = []        libs += ['caffe2']        build\_libs(libs)         # Use copies instead of symbolic files.        # Windows has very poor support for them.        sym\_files = ['tools/shared/cwrap\_common.py', 'tools/shared/\_utils\_internal.py']        orig\_files = ['aten/src/ATen/common\_with\_cwrap.py', 'torch/\_utils\_internal.py']        for sym\_file, orig\_file in zip(sym\_files, orig\_files):            same = False            if os.path.exists(sym\_file):                if filecmp.cmp(sym\_file, orig\_file):                    same = True                else:                    os.remove(sym\_file)            if not same:                shutil.copyfile(orig\_file, sym\_file)         self.copy\_tree('torch/lib/tmp\_install/share', 'torch/share')        self.copy\_tree('third\_party/pybind11/include/pybind11/',                       'torch/lib/include/pybind11')  build\_dep\_cmds = {}rebuild\_dep\_cmds = {} for lib in dep\_libs:    # wrap in function to capture lib    class build\_dep(build\_deps):        description = 'Build {} external library'.format(lib)         def run(self):            build\_libs([self.lib])    build\_dep.lib = lib    build\_dep\_cmds['build\_' + lib.lower()] = build\_dep     class rebuild\_dep(build\_deps):        description = 'Rebuild {} external library'.format(lib)         def run(self):            tools.setup\_helpers.configure.RERUN\_CMAKE = False            build\_libs([self.lib])    rebuild\_dep.lib = lib    rebuild\_dep\_cmds['rebuild\_' + lib.lower()] = rebuild\_dep  class build\_module(PytorchCommand):    def run(self):        report('setup.py::build\_module::run()')        self.run\_command('build\_py')        self.run\_command('build\_ext')  class build\_py(setuptools.command.build\_py.build\_py):     def run(self):        report('setup.py::build\_py::run()')        self.run\_command('create\_version\_file')        setuptools.command.build\_py.build\_py.run(self)  class develop(setuptools.command.develop.develop):     def run(self):        report('setup.py::develop::run()')        self.run\_command('create\_version\_file')        setuptools.command.develop.develop.run(self)        self.create\_compile\_commands()     def create\_compile\_commands(self):        def load(filename):            with open(filename) as f:                return json.load(f)        ninja\_files = glob.glob('build/\*compile\_commands.json')        cmake\_files = glob.glob('torch/lib/build/\*/compile\_commands.json')        all\_commands = [entry                        for f in ninja\_files + cmake\_files                        for entry in load(f)]         # cquery does not like c++ compiles that start with gcc.        # It forgets to include the c++ header directories.        # We can work around this by replacing the gcc calls that python        # setup.py generates with g++ calls instead        for command in all\_commands:            if command['command'].startswith("gcc "):                command['command'] = "g++ " + command['command'][4:]         new\_contents = json.dumps(all\_commands, indent=2)        contents = ''        if os.path.exists('compile\_commands.json'):            with open('compile\_commands.json', 'r') as f:                contents = f.read()        if contents != new\_contents:            with open('compile\_commands.json', 'w') as f:                f.write(new\_contents)         if not USE\_NINJA:            report("WARNING: 'develop' is not building C++ code incrementally")            report("because ninja is not installed. Run this to enable it:")            report(" > pip install ninja")  build\_ext\_parent = ninja\_build\_ext if USE\_NINJA \    else setuptools.command.build\_ext.build\_ext  class build\_ext(build\_ext\_parent):     def run(self):        # report build options        if USE\_NUMPY:            report('-- Building with NumPy bindings')        else:            report('-- NumPy not found')        if USE\_CUDNN:            report('-- Detected cuDNN at ' + CUDNN\_LIBRARY + ', ' + CUDNN\_INCLUDE\_DIR)        else:            report('-- Not using cuDNN')        if USE\_MIOPEN:            report('-- Detected MIOpen at ' + MIOPEN\_LIBRARY + ', ' + MIOPEN\_INCLUDE\_DIR)        else:            report('-- Not using MIOpen')        if USE\_CUDA:            report('-- Detected CUDA at ' + CUDA\_HOME)        else:            report('-- Not using CUDA')        if USE\_MKLDNN:            report('-- Using MKLDNN')        else:            report('-- Not using MKLDNN')        if USE\_NCCL and USE\_SYSTEM\_NCCL:            report('-- Using system provided NCCL library at ' + NCCL\_SYSTEM\_LIB + ', ' + NCCL\_INCLUDE\_DIR)        elif USE\_NCCL:            report('-- Building NCCL library')        else:            report('-- Not using NCCL')        if USE\_DISTRIBUTED:            report('-- Building with THD distributed package ')            if IS\_LINUX:                report('-- Building with c10d distributed package ')            else:                report('-- Building without c10d distributed package')        else:            report('-- Building without distributed package')         # It's an old-style class in Python 2.7...        setuptools.command.build\_ext.build\_ext.run(self)         # Copy the essential export library to compile C++ extensions.        if IS\_WINDOWS:            build\_temp = self.build\_temp             ext\_filename = self.get\_ext\_filename('\_C')            lib\_filename = '.'.join(ext\_filename.split('.')[:-1]) + '.lib'             export\_lib = os.path.join(                build\_temp, 'torch', 'csrc', lib\_filename).replace('\\', '/')             build\_lib = self.build\_lib             target\_lib = os.path.join(                build\_lib, 'torch', 'lib', '\_C.lib').replace('\\', '/')             self.copy\_file(export\_lib, target\_lib)     def build\_extensions(self):        # The caffe2 extensions are created in        # tmp\_install/lib/pythonM.m/site-packages/caffe2/python/        # and need to be copied to build/lib.linux.... , which will be a        # platform dependent build folder created by the "build" command of        # setuptools. Only the contents of this folder are installed in the        # "install" command by default.        # We only make this copy for Caffe2's pybind extensions        caffe2\_pybind\_exts = [            'caffe2.python.caffe2\_pybind11\_state',            'caffe2.python.caffe2\_pybind11\_state\_gpu',            'caffe2.python.caffe2\_pybind11\_state\_hip',        ]        i = 0        while i < len(self.extensions):            ext = self.extensions[i]            if ext.name not in caffe2\_pybind\_exts:                i += 1                continue            fullname = self.get\_ext\_fullname(ext.name)            filename = self.get\_ext\_filename(fullname)            report("\nCopying extension {}".format(ext.name))             src = os.path.join(tmp\_install\_path, rel\_site\_packages, filename)            if not os.path.exists(src):                report("{} does not exist".format(src))                del self.extensions[i]            else:                dst = os.path.join(os.path.realpath(self.build\_lib), filename)                report("Copying {} from {} to {}".format(ext.name, src, dst))                dst\_dir = os.path.dirname(dst)                if not os.path.exists(dst\_dir):                    os.makedirs(dst\_dir)                self.copy\_file(src, dst)                i += 1        distutils.command.build\_ext.build\_ext.build\_extensions(self)     def get\_outputs(self):        outputs = distutils.command.build\_ext.build\_ext.get\_outputs(self)        outputs.append(os.path.join(self.build\_lib, "caffe2"))        report("setup.py::get\_outputs returning {}".format(outputs))        return outputs  class build(distutils.command.build.build):    sub\_commands = [        ('build\_deps', lambda self: True),    ] + distutils.command.build.build.sub\_commands  class rebuild(distutils.command.build.build):    sub\_commands = [        ('build\_deps', lambda self: True),    ] + distutils.command.build.build.sub\_commands     def run(self):        tools.setup\_helpers.configure.RERUN\_CMAKE = False        distutils.command.build.build.run(self)  class install(setuptools.command.install.install):     def run(self):        report('setup.py::run()')        if not self.skip\_build:            self.run\_command('build\_deps')         setuptools.command.install.install.run(self)  class clean(distutils.command.clean.clean):    def run(self):        import glob        import re        with open('.gitignore', 'r') as f:            ignores = f.read()            pat = re.compile(r'^#( BEGIN NOT-CLEAN-FILES )?')            for wildcard in filter(None, ignores.split('\n')):                match = pat.match(wildcard)                if match:                    if match.group(1):                        # Marker is found and stop reading .gitignore.                        break                    # Ignore lines which begin with '#'.                else:                    for filename in glob.glob(wildcard):                        try:                            os.remove(filename)                        except OSError:                            shutil.rmtree(filename, ignore\_errors=True)         # It's an old-style class in Python 2.7...        distutils.command.clean.clean.run(self)  ################################################################################# Configure compile flags################################################################################ library\_dirs = [] if IS\_WINDOWS:    # /NODEFAULTLIB makes sure we only link to DLL runtime    # and matches the flags set for protobuf and ONNX    extra\_link\_args = ['/NODEFAULTLIB:LIBCMT.LIB']    # /MD links against DLL runtime    # and matches the flags set for protobuf and ONNX    # /Z7 turns on symbolic debugging information in .obj files    # /EHa is about native C++ catch support for asynchronous    # structured exception handling (SEH)    # /DNOMINMAX removes builtin min/max functions    # /wdXXXX disables warning no. XXXX    extra\_compile\_args = ['/MD', '/Z7',                          '/EHa', '/DNOMINMAX',                          '/wd4267', '/wd4251', '/wd4522', '/wd4522', '/wd4838',                          '/wd4305', '/wd4244', '/wd4190', '/wd4101', '/wd4996',                          '/wd4275']    if sys.version\_info[0] == 2:        if not check\_env\_flag('FORCE\_PY27\_BUILD'):            report('The support for PyTorch with Python 2.7 on Windows is very experimental.')            report('Please set the flag `FORCE\_PY27\_BUILD` to 1 to continue build.')            sys.exit(1)        # /bigobj increases number of sections in .obj file, which is needed to link        # against libaries in Python 2.7 under Windows        extra\_compile\_args.append('/bigobj')else:    extra\_link\_args = []    extra\_compile\_args = [        '-std=c++11',        '-Wall',        '-Wextra',        '-Wno-strict-overflow',        '-Wno-unused-parameter',        '-Wno-missing-field-initializers',        '-Wno-write-strings',        '-Wno-unknown-pragmas',        # This is required for Python 2 declarations that are deprecated in 3.        '-Wno-deprecated-declarations',        # Python 2.6 requires -fno-strict-aliasing, see        # http://legacy.python.org/dev/peps/pep-3123/        # We also depend on it in our code (even Python 3).        '-fno-strict-aliasing',        # Clang has an unfixed bug leading to spurious missing        # braces warnings, see        # https://bugs.llvm.org/show\_bug.cgi?id=21629        '-Wno-missing-braces',    ]    if check\_env\_flag('WERROR'):        extra\_compile\_args.append('-Werror') library\_dirs.append(lib\_path) # we specify exact lib names to avoid conflict with lua-torch installsCAFFE2\_LIBS = []if USE\_CUDA:    CAFFE2\_LIBS.extend(['-Wl,--no-as-needed', os.path.join(lib\_path, 'libcaffe2\_gpu.so'), '-Wl,--as-needed'])if USE\_ROCM:    CAFFE2\_LIBS.extend(['-Wl,--no-as-needed', os.path.join(lib\_path, 'libcaffe2\_hip.so'), '-Wl,--as-needed']) # static library onlyif IS\_DARWIN:    CAFFE2\_LIBS = []    if USE\_CUDA:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'libcaffe2\_gpu.dylib'))    if USE\_ROCM:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'libcaffe2\_hip.dylib')) if IS\_WINDOWS:    CAFFE2\_LIBS = []    if USE\_CUDA:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'caffe2\_gpu.lib'))    if USE\_ROCM:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'caffe2\_hip.lib')) main\_compile\_args = ['-D\_THP\_CORE', '-DONNX\_NAMESPACE=' + ONNX\_NAMESPACE]main\_libraries = ['shm', 'torch\_python']main\_link\_args = []main\_sources = ["torch/csrc/stub.cpp"] # Before the introduction of stub.cpp, \_C.so and libcaffe2.so defined# some of the same symbols, and it was important for \_C.so to be# loaded before libcaffe2.so so that the versions in \_C.so got# used. This happened automatically because we loaded \_C.so directly,# and libcaffe2.so was brought in as a dependency (though I suspect it# may have been possible to break by importing caffe2 first in the# same process).## Now, libtorch\_python.so and libcaffe2.so define some of the same# symbols. We directly load the \_C.so stub, which brings both of these# in as dependencies. We have to make sure that symbols continue to be# looked up in libtorch\_python.so first, by making sure it comes# before libcaffe2.so in the linker command.main\_link\_args.extend(CAFFE2\_LIBS) try:    import numpy as np    NUMPY\_INCLUDE\_DIR = np.get\_include()    USE\_NUMPY = Trueexcept ImportError:    USE\_NUMPY = False if USE\_CUDA:    if IS\_WINDOWS:        cuda\_lib\_path = CUDA\_HOME + '/lib/x64/'    else:        cuda\_lib\_dirs = ['lib64', 'lib']        for lib\_dir in cuda\_lib\_dirs:            cuda\_lib\_path = os.path.join(CUDA\_HOME, lib\_dir)            if os.path.exists(cuda\_lib\_path):                break    library\_dirs.append(cuda\_lib\_path) if DEBUG:    if IS\_WINDOWS:        extra\_link\_args.append('/DEBUG:FULL')    else:        extra\_compile\_args += ['-O0', '-g']        extra\_link\_args += ['-O0', '-g'] if REL\_WITH\_DEB\_INFO:    if IS\_WINDOWS:        extra\_link\_args.append('/DEBUG:FULL')    else:        extra\_compile\_args += ['-g']        extra\_link\_args += ['-g']  def make\_relative\_rpath(path):    if IS\_DARWIN:        return '-Wl,-rpath,@loader\_path/' + path    elif IS\_WINDOWS:        return ''    else:        return '-Wl,-rpath,$ORIGIN/' + path ################################################################################# Declare extensions and package################################################################################ extensions = []packages = find\_packages(exclude=('tools', 'tools.\*'))C = Extension("torch.\_C",              libraries=main\_libraries,              sources=main\_sources,              language='c++',              extra\_compile\_args=main\_compile\_args + extra\_compile\_args,              include\_dirs=[],              library\_dirs=library\_dirs,              extra\_link\_args=extra\_link\_args + main\_link\_args + [make\_relative\_rpath('lib')],              )extensions.append(C) if not IS\_WINDOWS:    DL = Extension("torch.\_dl",                   sources=["torch/csrc/dl.c"],                   language='c'                   )    extensions.append(DL)  if USE\_CUDA:    thnvrtc\_link\_flags = extra\_link\_args + [make\_relative\_rpath('lib')]    if IS\_LINUX:        thnvrtc\_link\_flags = thnvrtc\_link\_flags + ['-Wl,--no-as-needed']    # these have to be specified as -lcuda in link\_flags because they    # have to come right after the `no-as-needed` option    if IS\_WINDOWS:        thnvrtc\_link\_flags += ['cuda.lib', 'nvrtc.lib']    else:        thnvrtc\_link\_flags += ['-lcuda', '-lnvrtc']    cuda\_stub\_path = [cuda\_lib\_path + '/stubs']    if IS\_DARWIN:        # on macOS this is where the CUDA stub is installed according to the manual        cuda\_stub\_path = ["/usr/local/cuda/lib"]    THNVRTC = Extension("torch.\_nvrtc",                        sources=['torch/csrc/nvrtc.cpp'],                        language='c++',                        extra\_compile\_args=main\_compile\_args + extra\_compile\_args,                        include\_dirs=[cwd],                        library\_dirs=library\_dirs + cuda\_stub\_path,                        extra\_link\_args=thnvrtc\_link\_flags,                        )    extensions.append(THNVRTC) # These extensions are built by cmake and copied manually in build\_extensions()# inside the build\_ext implementaitonextensions.append(    Extension(        name=str('caffe2.python.caffe2\_pybind11\_state'),        sources=[]),)if USE\_CUDA:    extensions.append(        Extension(            name=str('caffe2.python.caffe2\_pybind11\_state\_gpu'),            sources=[]),    )if USE\_ROCM:    extensions.append(        Extension(            name=str('caffe2.python.caffe2\_pybind11\_state\_hip'),            sources=[]),    ) cmdclass = {    'create\_version\_file': create\_version\_file,    'build': build,    'build\_py': build\_py,    'build\_ext': build\_ext,    'build\_deps': build\_deps,    'build\_module': build\_module,    'rebuild': rebuild,    'develop': develop,    'install': install,    'clean': clean,}cmdclass.update(build\_dep\_cmds)cmdclass.update(rebuild\_dep\_cmds) entry\_points = {    'console\_scripts': [        'convert-caffe2-to-onnx = caffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'caffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inccaffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',

'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inccaffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inclib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/include/pybind11/\*.h',                'lib/include/pybind11/detail/\*.h',                'lib/include/TH/\*.h\*',                'lib/include/TH/generic/\*.h\*',                'lib/include/THC/\*.cuh',                'lib/include/THC/\*.h\*',                'lib/include/THC/generic/\*.h',                'lib/include/THCUNN/\*.cuh',                'lib/include/THNN/\*.h',                'share/cmake/ATen/\*.cmake',                'share/cmake/Caffe2/\*.cmake',                'share/cmake/Caffe2/public/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/FindCUDA/\*.cmake',                'share/cmake/Gloo/\*.cmake',                'share/cmake/Torch/\*.cmake',            ],            'caffe2': [                'cpp\_test/\*',                'python/serialized\_test/data/operator\_test/\*.zip',            ]        },    )

'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inclib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/include/pybind11/\*.h',                'lib/include/pybind11/detail/\*.h',                'lib/include/TH/\*.h\*',                'lib/include/TH/generic/\*.h\*',                'lib/include/THC/\*.cuh',                'lib/include/THC/\*.h\*',                'lib/include/THC/generic/\*.h',                'lib/include/THCUNN/\*.cuh',                'lib/include/THNN/\*.h',                'share/cmake/ATen/\*.cmake',                'share/cmake/Caffe2/\*.cmake',                'share/cmake/Caffe2/public/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/FindCUDA/\*.cmake',                'share/cmake/Gloo/\*.cmake',                'share/cmake/Torch/\*.cmake',            ],            'caffe2': [                'cpp\_test/\*',                'python/serialized\_test/data/operator\_test/\*.zip',            ]        },    )

VERBOSE\_SCRIPT:    def report(\*args):        print(\*args)else:    def report(\*args):        pass # Constant known variables used throughout this filecwd = os.path.dirname(os.path.abspath(\_\_file\_\_))lib\_path = os.path.join(cwd, "torch", "lib")third\_party\_path = os.path.join(cwd, "third\_party")tmp\_install\_path = lib\_path + "/tmp\_install"caffe2\_build\_dir = os.path.join(cwd, "build")# lib/pythonx.x/site-packagesrel\_site\_packages = distutils.sysconfig.get\_python\_lib(prefix='')# full absolute path to the dir abovefull\_site\_packages = distutils.sysconfig.get\_python\_lib()# CMAKE: full path to python libraryif IS\_WINDOWS:    cmake\_python\_library = "{}/libs/python{}.lib".format(        distutils.sysconfig.get\_config\_var("prefix"),        distutils.sysconfig.get\_config\_var("VERSION"))else:    cmake\_python\_library = "{}/{}".format(        distutils.sysconfig.get\_config\_var("LIBDIR"),        distutils.sysconfig.get\_config\_var("INSTSONAME"))cmake\_python\_include\_dir = distutils.sysconfig.get\_python\_inc()  class PytorchCommand(setuptools.Command):    """    Base Pytorch command to avoid implementing initialize/finalize\_options in    every subclass    """    user\_options = []     def initialize\_options(self):        pass     def finalize\_options(self):        pass  ################################################################################# Version, create\_version\_file, and package\_name################################################################################package\_name = os.getenv('TORCH\_PACKAGE\_NAME', 'torch')version = '1.1.0a0'if os.getenv('PYTORCH\_BUILD\_VERSION'):    assert os.getenv('PYTORCH\_BUILD\_NUMBER') is not None    build\_number = int(os.getenv('PYTORCH\_BUILD\_NUMBER'))    version = os.getenv('PYTORCH\_BUILD\_VERSION')    if build\_number > 1:        version += '.post' + str(build\_number)else:    try:        sha = subprocess.check\_output(['git', 'rev-parse', 'HEAD'], cwd=cwd).decode('ascii').strip()        version += '+' + sha[:7]    except Exception:        passreport("Building wheel {}-{}".format(package\_name, version))  class create\_version\_file(PytorchCommand):    def run(self):        global version, cwd        report('-- Building version ' + version)        version\_path = os.path.join(cwd, 'torch', 'version.py')        with open(version\_path, 'w') as f:            f.write("\_\_version\_\_ = '{}'\n".format(version))            # NB: This is not 100% accurate, because you could have built the            # library code with DEBUG, but csrc without DEBUG (in which case            # this would claim to be a release build when it's not.)            f.write("debug = {}\n".format(repr(DEBUG)))            f.write("cuda = {}\n".format(repr(CUDA\_VERSION)))  ################################################################################# Building dependent libraries################################################################################ # All libraries that torch could depend ondep\_libs = ['caffe2'] missing\_pydep = '''Missing build dependency: Unable to `import {importname}`.Please install it via `conda install {module}` or `pip install {module}`'''.strip()  def check\_pydep(importname, module):    try:        importlib.import\_module(importname)    except ImportError:        raise RuntimeError(missing\_pydep.format(importname=importname, module=module))  # Calls build\_pytorch\_libs.sh/bat with the correct env variablesdef build\_libs(libs):    for lib in libs:        assert lib in dep\_libs, 'invalid lib: {}'.format(lib)    if IS\_WINDOWS:        build\_libs\_cmd = ['tools\\build\_pytorch\_libs.bat']    else:        build\_libs\_cmd = ['bash', os.path.join('..', 'tools', 'build\_pytorch\_libs.sh')]     my\_env, extra\_flags = get\_pytorch\_env\_with\_flags()    build\_libs\_cmd.extend(extra\_flags)    my\_env["PYTORCH\_PYTHON\_LIBRARY"] = cmake\_python\_library    my\_env["PYTORCH\_PYTHON\_INCLUDE\_DIR"] = cmake\_python\_include\_dir    my\_env["PYTORCH\_BUILD\_VERSION"] = version     cmake\_prefix\_path = full\_site\_packages    if "CMAKE\_PREFIX\_PATH" in my\_env:        cmake\_prefix\_path = my\_env["CMAKE\_PREFIX\_PATH"] + ";" + cmake\_prefix\_path    my\_env["CMAKE\_PREFIX\_PATH"] = cmake\_prefix\_path     if VERBOSE\_SCRIPT:        my\_env['VERBOSE\_SCRIPT'] = '1'    try:        os.mkdir('build')    except OSError:        pass     kwargs = {'cwd': 'build'} if not IS\_WINDOWS else {}     if subprocess.call(build\_libs\_cmd + libs, env=my\_env, \*\*kwargs) != 0:        report("Failed to run '{}'".format(' '.join(build\_libs\_cmd + libs)))        sys.exit(1)  # Build all dependent librariesclass build\_deps(PytorchCommand):    def run(self):        report('setup.py::build\_deps::run()')        # Check if you remembered to check out submodules         def check\_file(f):            if not os.path.exists(f):                report("Could not find {}".format(f))                report("Did you run 'git submodule update --init --recursive'?")                sys.exit(1)         check\_file(os.path.join(third\_party\_path, "gloo", "CMakeLists.txt"))        check\_file(os.path.join(third\_party\_path, "pybind11", "CMakeLists.txt"))        check\_file(os.path.join(third\_party\_path, 'cpuinfo', 'CMakeLists.txt'))        check\_file(os.path.join(third\_party\_path, 'onnx', 'CMakeLists.txt'))        check\_file(os.path.join(third\_party\_path, 'QNNPACK', 'CMakeLists.txt'))        check\_file(os.path.join(third\_party\_path, 'fbgemm', 'CMakeLists.txt'))         check\_pydep('yaml', 'pyyaml')        check\_pydep('typing', 'typing')         libs = []        libs += ['caffe2']        build\_libs(libs)         # Use copies instead of symbolic files.        # Windows has very poor support for them.        sym\_files = ['tools/shared/cwrap\_common.py', 'tools/shared/\_utils\_internal.py']        orig\_files = ['aten/src/ATen/common\_with\_cwrap.py', 'torch/\_utils\_internal.py']        for sym\_file, orig\_file in zip(sym\_files, orig\_files):            same = False            if os.path.exists(sym\_file):                if filecmp.cmp(sym\_file, orig\_file):                    same = True                else:                    os.remove(sym\_file)            if not same:                shutil.copyfile(orig\_file, sym\_file)         self.copy\_tree('torch/lib/tmp\_install/share', 'torch/share')        self.copy\_tree('third\_party/pybind11/include/pybind11/',                       'torch/lib/include/pybind11')  build\_dep\_cmds = {}rebuild\_dep\_cmds = {} for lib in dep\_libs:    # wrap in function to capture lib    class build\_dep(build\_deps):        description = 'Build {} external library'.format(lib)         def run(self):            build\_libs([self.lib])    build\_dep.lib = lib    build\_dep\_cmds['build\_' + lib.lower()] = build\_dep     class rebuild\_dep(build\_deps):        description = 'Rebuild {} external library'.format(lib)         def run(self):            tools.setup\_helpers.configure.RERUN\_CMAKE = False            build\_libs([self.lib])    rebuild\_dep.lib = lib    rebuild\_dep\_cmds['rebuild\_' + lib.lower()] = rebuild\_dep  class build\_module(PytorchCommand):    def run(self):        report('setup.py::build\_module::run()')        self.run\_command('build\_py')        self.run\_command('build\_ext')  class build\_py(setuptools.command.build\_py.build\_py):     def run(self):        report('setup.py::build\_py::run()')        self.run\_command('create\_version\_file')        setuptools.command.build\_py.build\_py.run(self)  class develop(setuptools.command.develop.develop):     def run(self):        report('setup.py::develop::run()')        self.run\_command('create\_version\_file')        setuptools.command.develop.develop.run(self)        self.create\_compile\_commands()     def create\_compile\_commands(self):        def load(filename):            with open(filename) as f:                return json.load(f)        ninja\_files = glob.glob('build/\*compile\_commands.json')        cmake\_files = glob.glob('torch/lib/build/\*/compile\_commands.json')        all\_commands = [entry                        for f in ninja\_files + cmake\_files                        for entry in load(f)]         # cquery does not like c++ compiles that start with gcc.        # It forgets to include the c++ header directories.        # We can work around this by replacing the gcc calls that python        # setup.py generates with g++ calls instead        for command in all\_commands:            if command['command'].startswith("gcc "):                command['command'] = "g++ " + command['command'][4:]         new\_contents = json.dumps(all\_commands, indent=2)        contents = ''        if os.path.exists('compile\_commands.json'):            with open('compile\_commands.json', 'r') as f:                contents = f.read()        if contents != new\_contents:            with open('compile\_commands.json', 'w') as f:                f.write(new\_contents)         if not USE\_NINJA:            report("WARNING: 'develop' is not building C++ code incrementally")            report("because ninja is not installed. Run this to enable it:")            report(" > pip install ninja")  build\_ext\_parent = ninja\_build\_ext if USE\_NINJA \    else setuptools.command.build\_ext.build\_ext  class build\_ext(build\_ext\_parent):     def run(self):        # report build options        if USE\_NUMPY:            report('-- Building with NumPy bindings')        else:            report('-- NumPy not found')        if USE\_CUDNN:            report('-- Detected cuDNN at ' + CUDNN\_LIBRARY + ', ' + CUDNN\_INCLUDE\_DIR)        else:            report('-- Not using cuDNN')        if USE\_MIOPEN:            report('-- Detected MIOpen at ' + MIOPEN\_LIBRARY + ', ' + MIOPEN\_INCLUDE\_DIR)        else:            report('-- Not using MIOpen')        if USE\_CUDA:            report('-- Detected CUDA at ' + CUDA\_HOME)        else:            report('-- Not using CUDA')        if USE\_MKLDNN:            report('-- Using MKLDNN')        else:            report('-- Not using MKLDNN')        if USE\_NCCL and USE\_SYSTEM\_NCCL:            report('-- Using system provided NCCL library at ' + NCCL\_SYSTEM\_LIB + ', ' + NCCL\_INCLUDE\_DIR)        elif USE\_NCCL:            report('-- Building NCCL library')        else:            report('-- Not using NCCL')        if USE\_DISTRIBUTED:            report('-- Building with THD distributed package ')            if IS\_LINUX:                report('-- Building with c10d distributed package ')            else:                report('-- Building without c10d distributed package')        else:            report('-- Building without distributed package')         # It's an old-style class in Python 2.7...        setuptools.command.build\_ext.build\_ext.run(self)         # Copy the essential export library to compile C++ extensions.        if IS\_WINDOWS:            build\_temp = self.build\_temp             ext\_filename = self.get\_ext\_filename('\_C')            lib\_filename = '.'.join(ext\_filename.split('.')[:-1]) + '.lib'             export\_lib = os.path.join(                build\_temp, 'torch', 'csrc', lib\_filename).replace('\\', '/')             build\_lib = self.build\_lib             target\_lib = os.path.join(                build\_lib, 'torch', 'lib', '\_C.lib').replace('\\', '/')             self.copy\_file(export\_lib, target\_lib)     def build\_extensions(self):        # The caffe2 extensions are created in        # tmp\_install/lib/pythonM.m/site-packages/caffe2/python/        # and need to be copied to build/lib.linux.... , which will be a        # platform dependent build folder created by the "build" command of        # setuptools. Only the contents of this folder are installed in the        # "install" command by default.        # We only make this copy for Caffe2's pybind extensions        caffe2\_pybind\_exts = [            'caffe2.python.caffe2\_pybind11\_state',            'caffe2.python.caffe2\_pybind11\_state\_gpu',            'caffe2.python.caffe2\_pybind11\_state\_hip',        ]        i = 0        while i < len(self.extensions):            ext = self.extensions[i]            if ext.name not in caffe2\_pybind\_exts:                i += 1                continue            fullname = self.get\_ext\_fullname(ext.name)            filename = self.get\_ext\_filename(fullname)            report("\nCopying extension {}".format(ext.name))             src = os.path.join(tmp\_install\_path, rel\_site\_packages, filename)            if not os.path.exists(src):                report("{} does not exist".format(src))                del self.extensions[i]            else:                dst = os.path.join(os.path.realpath(self.build\_lib), filename)                report("Copying {} from {} to {}".format(ext.name, src, dst))                dst\_dir = os.path.dirname(dst)                if not os.path.exists(dst\_dir):                    os.makedirs(dst\_dir)                self.copy\_file(src, dst)                i += 1        distutils.command.build\_ext.build\_ext.build\_extensions(self)     def get\_outputs(self):        outputs = distutils.command.build\_ext.build\_ext.get\_outputs(self)        outputs.append(os.path.join(self.build\_lib, "caffe2"))        report("setup.py::get\_outputs returning {}".format(outputs))        return outputs  class build(distutils.command.build.build):    sub\_commands = [        ('build\_deps', lambda self: True),    ] + distutils.command.build.build.sub\_commands  class rebuild(distutils.command.build.build):    sub\_commands = [        ('build\_deps', lambda self: True),    ] + distutils.command.build.build.sub\_commands     def run(self):        tools.setup\_helpers.configure.RERUN\_CMAKE = False        distutils.command.build.build.run(self)  class install(setuptools.command.install.install):     def run(self):        report('setup.py::run()')        if not self.skip\_build:            self.run\_command('build\_deps')         setuptools.command.install.install.run(self)  class clean(distutils.command.clean.clean):    def run(self):        import glob        import re        with open('.gitignore', 'r') as f:            ignores = f.read()            pat = re.compile(r'^#( BEGIN NOT-CLEAN-FILES )?')            for wildcard in filter(None, ignores.split('\n')):                match = pat.match(wildcard)                if match:                    if match.group(1):                        # Marker is found and stop reading .gitignore.                        break                    # Ignore lines which begin with '#'.                else:                    for filename in glob.glob(wildcard):                        try:                            os.remove(filename)                        except OSError:                            shutil.rmtree(filename, ignore\_errors=True)         # It's an old-style class in Python 2.7...        distutils.command.clean.clean.run(self)  ################################################################################# Configure compile flags################################################################################ library\_dirs = [] if IS\_WINDOWS:    # /NODEFAULTLIB makes sure we only link to DLL runtime    # and matches the flags set for protobuf and ONNX    extra\_link\_args = ['/NODEFAULTLIB:LIBCMT.LIB']    # /MD links against DLL runtime    # and matches the flags set for protobuf and ONNX    # /Z7 turns on symbolic debugging information in .obj files    # /EHa is about native C++ catch support for asynchronous    # structured exception handling (SEH)    # /DNOMINMAX removes builtin min/max functions    # /wdXXXX disables warning no. XXXX    extra\_compile\_args = ['/MD', '/Z7',                          '/EHa', '/DNOMINMAX',                          '/wd4267', '/wd4251', '/wd4522', '/wd4522', '/wd4838',                          '/wd4305', '/wd4244', '/wd4190', '/wd4101', '/wd4996',                          '/wd4275']    if sys.version\_info[0] == 2:        if not check\_env\_flag('FORCE\_PY27\_BUILD'):            report('The support for PyTorch with Python 2.7 on Windows is very experimental.')            report('Please set the flag `FORCE\_PY27\_BUILD` to 1 to continue build.')            sys.exit(1)        # /bigobj increases number of sections in .obj file, which is needed to link        # against libaries in Python 2.7 under Windows        extra\_compile\_args.append('/bigobj')else:    extra\_link\_args = []    extra\_compile\_args = [        '-std=c++11',        '-Wall',        '-Wextra',        '-Wno-strict-overflow',        '-Wno-unused-parameter',        '-Wno-missing-field-initializers',        '-Wno-write-strings',        '-Wno-unknown-pragmas',        # This is required for Python 2 declarations that are deprecated in 3.        '-Wno-deprecated-declarations',        # Python 2.6 requires -fno-strict-aliasing, see        # http://legacy.python.org/dev/peps/pep-3123/        # We also depend on it in our code (even Python 3).        '-fno-strict-aliasing',        # Clang has an unfixed bug leading to spurious missing        # braces warnings, see        # https://bugs.llvm.org/show\_bug.cgi?id=21629        '-Wno-missing-braces',    ]    if check\_env\_flag('WERROR'):        extra\_compile\_args.append('-Werror') library\_dirs.append(lib\_path) # we specify exact lib names to avoid conflict with lua-torch installsCAFFE2\_LIBS = []if USE\_CUDA:    CAFFE2\_LIBS.extend(['-Wl,--no-as-needed', os.path.join(lib\_path, 'libcaffe2\_gpu.so'), '-Wl,--as-needed'])if USE\_ROCM:    CAFFE2\_LIBS.extend(['-Wl,--no-as-needed', os.path.join(lib\_path, 'libcaffe2\_hip.so'), '-Wl,--as-needed']) # static library onlyif IS\_DARWIN:    CAFFE2\_LIBS = []    if USE\_CUDA:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'libcaffe2\_gpu.dylib'))    if USE\_ROCM:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'libcaffe2\_hip.dylib')) if IS\_WINDOWS:    CAFFE2\_LIBS = []    if USE\_CUDA:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'caffe2\_gpu.lib'))    if USE\_ROCM:        CAFFE2\_LIBS.append(os.path.join(lib\_path, 'caffe2\_hip.lib')) main\_compile\_args = ['-D\_THP\_CORE', '-DONNX\_NAMESPACE=' + ONNX\_NAMESPACE]main\_libraries = ['shm', 'torch\_python']main\_link\_args = []main\_sources = ["torch/csrc/stub.cpp"] # Before the introduction of stub.cpp, \_C.so and libcaffe2.so defined# some of the same symbols, and it was important for \_C.so to be# loaded before libcaffe2.so so that the versions in \_C.so got# used. This happened automatically because we loaded \_C.so directly,# and libcaffe2.so was brought in as a dependency (though I suspect it# may have been possible to break by importing caffe2 first in the# same process).## Now, libtorch\_python.so and libcaffe2.so define some of the same# symbols. We directly load the \_C.so stub, which brings both of these# in as dependencies. We have to make sure that symbols continue to be# looked up in libtorch\_python.so first, by making sure it comes# before libcaffe2.so in the linker command.main\_link\_args.extend(CAFFE2\_LIBS) try:    import numpy as np    NUMPY\_INCLUDE\_DIR = np.get\_include()    USE\_NUMPY = Trueexcept ImportError:    USE\_NUMPY = False if USE\_CUDA:    if IS\_WINDOWS:        cuda\_lib\_path = CUDA\_HOME + '/lib/x64/'    else:        cuda\_lib\_dirs = ['lib64', 'lib']        for lib\_dir in cuda\_lib\_dirs:            cuda\_lib\_path = os.path.join(CUDA\_HOME, lib\_dir)            if os.path.exists(cuda\_lib\_path):                break    library\_dirs.append(cuda\_lib\_path) if DEBUG:    if IS\_WINDOWS:        extra\_link\_args.append('/DEBUG:FULL')    else:        extra\_compile\_args += ['-O0', '-g']        extra\_link\_args += ['-O0', '-g'] if REL\_WITH\_DEB\_INFO:    if IS\_WINDOWS:        extra\_link\_args.append('/DEBUG:FULL')    else:        extra\_compile\_args += ['-g']        extra\_link\_args += ['-g']  def make\_relative\_rpath(path):    if IS\_DARWIN:        return '-Wl,-rpath,@loader\_path/' + path    elif IS\_WINDOWS:        return ''    else:        return '-Wl,-rpath,$ORIGIN/' + path ################################################################################# Declare extensions and package################################################################################ extensions = []packages = find\_packages(exclude=('tools', 'tools.\*'))C = Extension("torch.\_C",              libraries=main\_libraries,              sources=main\_sources,              language='c++',              extra\_compile\_args=main\_compile\_args + extra\_compile\_args,              include\_dirs=[],              library\_dirs=library\_dirs,              extra\_link\_args=extra\_link\_args + main\_link\_args + [make\_relative\_rpath('lib')],              )extensions.append(C) if not IS\_WINDOWS:    DL = Extension("torch.\_dl",                   sources=["torch/csrc/dl.c"],                   language='c'                   )    extensions.append(DL)  if USE\_CUDA:    thnvrtc\_link\_flags = extra\_link\_args + [make\_relative\_rpath('lib')]    if IS\_LINUX:        thnvrtc\_link\_flags = thnvrtc\_link\_flags + ['-Wl,--no-as-needed']    # these have to be specified as -lcuda in link\_flags because they    # have to come right after the `no-as-needed` option    if IS\_WINDOWS:        thnvrtc\_link\_flags += ['cuda.lib', 'nvrtc.lib']    else:        thnvrtc\_link\_flags += ['-lcuda', '-lnvrtc']    cuda\_stub\_path = [cuda\_lib\_path + '/stubs']    if IS\_DARWIN:        # on macOS this is where the CUDA stub is installed according to the manual        cuda\_stub\_path = ["/usr/local/cuda/lib"]    THNVRTC = Extension("torch.\_nvrtc",                        sources=['torch/csrc/nvrtc.cpp'],                        language='c++',                        extra\_compile\_args=main\_compile\_args + extra\_compile\_args,                        include\_dirs=[cwd],                        library\_dirs=library\_dirs + cuda\_stub\_path,                        extra\_link\_args=thnvrtc\_link\_flags,                        )    extensions.append(THNVRTC) # These extensions are built by cmake and copied manually in build\_extensions()# inside the build\_ext implementaitonextensions.append(    Extension(        name=str('caffe2.python.caffe2\_pybind11\_state'),        sources=[]),)if USE\_CUDA:    extensions.append(        Extension(            name=str('caffe2.python.caffe2\_pybind11\_state\_gpu'),            sources=[]),    )if USE\_ROCM:    extensions.append(        Extension(            name=str('caffe2.python.caffe2\_pybind11\_state\_hip'),            sources=[]),    ) cmdclass = {    'create\_version\_file': create\_version\_file,    'build': build,    'build\_py': build\_py,    'build\_ext': build\_ext,    'build\_deps': build\_deps,    'build\_module': build\_module,    'rebuild': rebuild,    'develop': develop,    'install': install,    'clean': clean,}cmdclass.update(build\_dep\_cmds)cmdclass.update(rebuild\_dep\_cmds) entry\_points = {    'console\_scripts': [        'convert-caffe2-to-onnx = caffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'caffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inccaffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',

'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inccaffe2.python.onnx.bin.conversion:caffe2\_to\_onnx',        'convert-onnx-to-caffe2 = caffe2.python.onnx.bin.conversion:onnx\_to\_caffe2',    ]} if \_\_name\_\_ == '\_\_main\_\_':    setup(        name=package\_name,        version=version,        description=("Tensors and Dynamic neural networks in "                     "Python with strong GPU acceleration"),        ext\_modules=extensions,        cmdclass=cmdclass,        packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inclib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/include/pybind11/\*.h',                'lib/include/pybind11/detail/\*.h',                'lib/include/TH/\*.h\*',                'lib/include/TH/generic/\*.h\*',                'lib/include/THC/\*.cuh',                'lib/include/THC/\*.h\*',                'lib/include/THC/generic/\*.h',                'lib/include/THCUNN/\*.cuh',                'lib/include/THNN/\*.h',                'share/cmake/ATen/\*.cmake',                'share/cmake/Caffe2/\*.cmake',                'share/cmake/Caffe2/public/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/FindCUDA/\*.cmake',                'share/cmake/Gloo/\*.cmake',                'share/cmake/Torch/\*.cmake',            ],            'caffe2': [                'cpp\_test/\*',                'python/serialized\_test/data/operator\_test/\*.zip',            ]        },    )

'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'li   'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h', packages=packages,        entry\_points=entry\_points,        package\_data={            'torch': [                'lib/\*.so\*',                'lib/\*.dylib\*',                'lib/\*.dll',                'lib/\*.lib',                'lib/\*.pdb',                'lib/torch\_shm\_manager',                'lib/\*.h',                'lib/include/ATen/\*.h',                'lib/include/ATen/cpu/\*.h',                'lib/include/ATen/core/\*.h',                'lib/include/ATen/cuda/\*.cuh',                'lib/include/ATen/cuda/\*.h',                'lib/include/ATen/cuda/detail/\*.cuh',                'lib/include/ATen/cuda/detail/\*.h',                'lib/include/ATen/cudnn/\*.h',                'lib/include/ATen/detail/\*.h',                'lib/include/caffe2/utils/\*.h',                'lib/include/c10/\*.h',                'lib/include/c10/macros/\*.h',                'lib/include/c10/core/\*.h',                'lib/include/ATen/core/dispatch/\*.h',                'lib/include/c10/core/impl/\*.h',                'lib/include/ATen/core/opschema/\*.h',                'lib/include/c10/util/\*.h',                'lib/include/c10/cuda/\*.h',                'lib/include/c10/cuda/impl/\*.h',                'lib/include/c10/hip/\*.h',                'lib/include/c10/hip/impl/\*.h',                'lib/include/caffe2/\*\*/\*.h',                'lib/include/torch/\*.h',                'lib/include/torch/csrc/\*.h',                'lib/include/torch/csrc/api/include/torch/\*.h',                'lib/include/torch/csrc/api/include/torch/data/\*.h',                'lib/include/torch/csrc/api/include/torch/data/dataloader/\*.h',                'lib/include/torch/csrc/api/include/torch/data/datasets/\*.h',                'lib/include/torch/csrc/api/include/torch/data/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/data/samplers/\*.h',                'lib/include/torch/csrc/api/include/torch/data/transforms/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/\*.h',                'lib/include/torch/csrc/api/include/torch/detail/ordered\_dict.h',                'lib/include/torch/csrc/api/include/torch/nn/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/inclib/include/torch/csrc/api/include/torch/nn/modules/\*.h',                'lib/include/torch/csrc/api/include/torch/nn/parallel/\*.h',                'lib/include/torch/csrc/api/include/torch/optim/\*.h',                'lib/include/torch/csrc/api/include/torch/serialize/\*.h',                'lib/include/torch/csrc/autograd/\*.h',                'lib/include/torch/csrc/autograd/generated/\*.h',                'lib/include/torch/csrc/cuda/\*.h',                'lib/include/torch/csrc/jit/\*.h',                'lib/include/torch/csrc/jit/generated/\*.h',                'lib/include/torch/csrc/jit/passes/\*.h',                'lib/include/torch/csrc/jit/script/\*.h',                'lib/include/torch/csrc/utils/\*.h',                'lib/include/pybind11/\*.h',                'lib/include/pybind11/detail/\*.h',                'lib/include/TH/\*.h\*',                'lib/include/TH/generic/\*.h\*',                'lib/include/THC/\*.cuh',                'lib/include/THC/\*.h\*',                'lib/include/THC/generic/\*.h',                'lib/include/THCUNN/\*.cuh',                'lib/include/THNN/\*.h',                'share/cmake/ATen/\*.cmake',                'share/cmake/Caffe2/\*.cmake',                'share/cmake/Caffe2/public/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/\*.cmake',                'share/cmake/Caffe2/Modules\_CUDA\_fix/upstream/FindCUDA/\*.cmake',                'share/cmake/Gloo/\*.cmake',                'share/cmake/Torch/\*.cmake',            ],            'caffe2': [                'cpp\_test/\*',                'python/serialized\_test/data/operator\_test/\*.zip',            ]        },    )