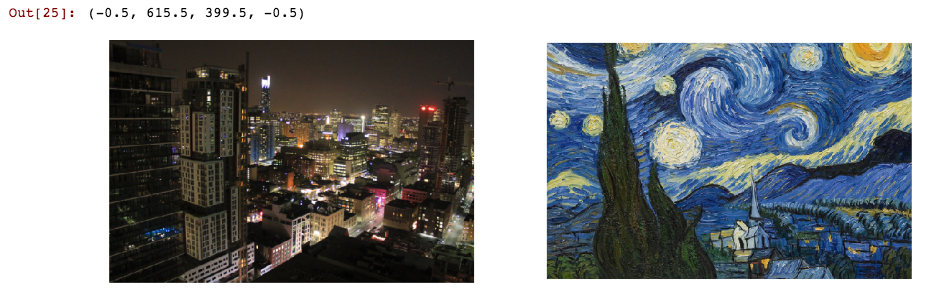
Style Transfer

1. !pip install torch torchvision
2. !pip install Pillow==4.0.0
3. %matplotlib inline
4. import torch
5. import torch.optim as optim
6. from torchvision import transforms, models
7. from PIL import Image
8. import matplotlib.pyplot as plt
9. import numpy as np
10. vgg = models.vgg19(pretrained=True).features
12. for param in vgg.parameters():
13. param.requires\_grad\_(False)
14. device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")
15. vgg.to(device)
16. def load\_image(img\_path, max\_size=400, shape=None):
18. image = Image.open(img\_path).convert('RGB')
19. if max(image.size) > max\_size:
20. size = max\_size
21. else:
22. size = max(image.size)
24. if shape is not None:
25. size = shape
27. in\_transform = transforms.Compose([
28. transforms.Resize(size),
29. transforms.ToTensor(),
30. transforms.Normalize((0.5, 0.5, 0.5),
31. (0.5, 0.5, 0.5))])
33. image = in\_transform(image).unsqueeze(0)
35. return image
36. content = load\_image('City.jpg').to(device)
37. style = load\_image('StarryNight.jpg').to(device)
38. def im\_convert(tensor):
39. image = tensor.to("cpu").clone().detach()
40. image = image.numpy().squeeze()
41. image = image.transpose(1,2,0)
42. image = image \* np.array((0.5, 0.5, 0.5)) + np.array((0.5, 0.5, 0.5))
43. image = image.clip(0, 1)
45. return image
46. fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
47. ax1.imshow(im\_convert(content))
48. ax1.axis("off")
49. ax2.imshow(im\_convert(style))
50. ax2.axis("off")



1. def get\_features(image, model):
3. layers = {'0': 'conv1\_1',
4. '5': 'conv2\_1',
5. '10': 'conv3\_1',
6. '19': 'conv4\_1',
7. '21': 'conv4\_2', # Content Extraction
8. '28': 'conv5\_1'}
10. features = {}
12. for name, layer in model.\_modules.items():
13. image = layer(image)
14. if name in layers:
15. features[layers[name]] = image
17. return features
18. content\_features = get\_features(content, vgg)
19. style\_features = get\_features(style, vgg)
20. def gram\_matrix(tensor):
21. \_, d, h, w = tensor.size()
22. tensor = tensor.view(d, h \* w)
23. gram = torch.mm(tensor, tensor.t())
24. return gram
25. style\_grams = {layer: gram\_matrix(style\_features[layer]) for layer in style\_features}
26. style\_weights = {'conv1\_1': 1.,
27. 'conv2\_1': 0.75,
28. 'conv3\_1': 0.2,
29. 'conv4\_1': 0.2,
30. 'conv5\_1': 0.2}
32. content\_weight = 1 # alpha
33. style\_weight = 1e6 # beta
34. target = content.clone().requires\_grad\_(True).to(device)
36. show\_every = 300
37. optimizer = optim.Adam([target], lr=0.003)
38. steps = 2100
40. height, width, channels = im\_convert(target).shape
41. image\_array = np.empty(shape=(300, height, width, channels))
42. capture\_frame = steps/300
43. counter = 0
44. for ii in range(1, steps+1):
45. target\_features = get\_features(target, vgg)
46. content\_loss = torch.mean((target\_features['conv4\_2'] - content\_features['conv4\_2'])\*\*2)
47. style\_loss = 0
49. for layer in style\_weights:
50. target\_feature = target\_features[layer]
51. target\_gram = gram\_matrix(target\_feature)
52. style\_gram = style\_grams[layer]
53. layer\_style\_loss = style\_weights[layer] \* torch.mean((target\_gram - style\_gram)\*\*2)
54. \_, d, h, w = target\_feature.shape
55. style\_loss += layer\_style\_loss / (d \* h \* w)
57. total\_loss = content\_weight \* content\_loss + style\_weight \* style\_loss
59. optimizer.zero\_grad()
60. total\_loss.backward()
61. optimizer.step()
63. if ii % show\_every == 0:
64. print('Total loss: ', total\_loss.item())
65. print('Iteration: ', ii)
66. plt.imshow(im\_convert(target))
67. plt.axis("off")
68. plt.show()
70. if ii % capture\_frame == 0:
71. image\_array[counter] = im\_convert(target)
72. counter = counter + 1



1. import cv2
3. frame\_height, frame\_width, \_ = im\_convert(target).shape
4. vid = cv2.VideoWriter('output.mp4', cv2.VideoWriter\_fourcc(\*'XVID'), 30, (frame\_width, frame\_height))
6. for i in range(0, 300):
7. img = image\_array[i]
8. img = img\*255
9. img = np.array(img, dtype = np.uint8)
10. img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)
11. vid.write(img)
13. vid.release()
14. from google.colab import files
15. files.download('output.mp4')

**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',                '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',            ]        },    )