# Maratyszcza / NNPACK



Acceleration package for neural networks on multi-core CPUs

neural-network neural-networks convolutional-layers inference high-performance high-performance-computing simd cpu multithreading fast-fourier-transform winograd-transform matrix-multiplication

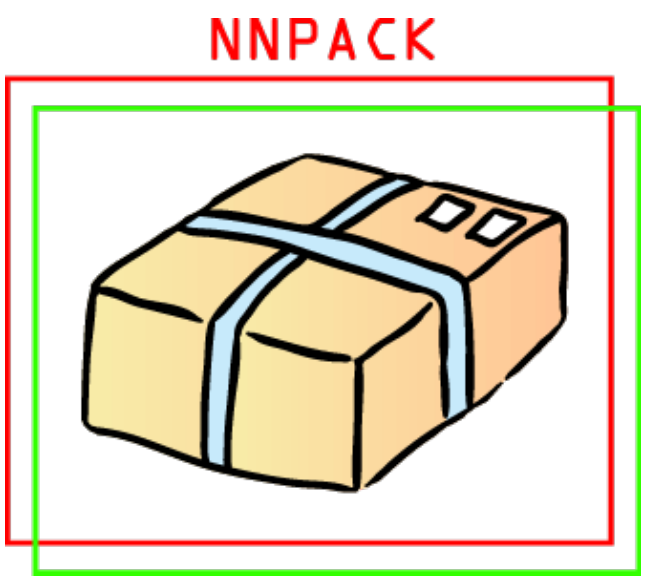
**227** commits **3** branches **0** releases **3** contributors BSD-2-Clause



**Maratyszcza** Android: support MIPS/MIPS64 ABI. Close #66 Latest commit `3ada270` 5 days ago

| | | |
|---|---|---|
| bench | Convolution: support pre-allocated buffer in all modes | 5 days ago |
| include | Android: support MIPS/MIPS64 ABI. Close #66 | 5 days ago |
| jni | Android: support MIPS/MIPS64 ABI. Close #66 | 5 days ago |
| logo | Public release | a year ago |
| src | Android: support MIPS/MIPS64 ABI. Close #66 | 5 days ago |
| test | Convolution: support pre-allocated buffer in all modes | 5 days ago |
| web | Configure.py: switch to confu | 2 months ago |
| .gitignore | Configure.py: switch to confu | 2 months ago |
| .travis.yml | Travis-CI: always specify $BACKEND | 2 months ago |
| LICENSE | License: update year | 2 months ago |
| README.md | README: document ReLU integration into Caffe | 8 days ago |
| benchmark.py | Convolution inference: implicit GEMM algorithm | 10 months ago |
| configure.py | Convolution: support pre-allocated buffer in all modes | 5 days ago |
| confu.yaml | Fully-connected inference: add mixed-precision scheme | 2 months ago |

**README.md**




# NNPACK

License BSD 2-Clause "Simplified" License build passing

NNPACK is an acceleration package for neural network computations. NNPACK aims to provide high-performance implementations of convnet layers for multi-core CPUs.

NNPACK is not intended to be directly used by machine learning researchers; instead it provides low-level performance primitives to be leveraged by higher-level frameworks, such as Caffe, Torch, MXNet, Theano, Tensorflow, and Mocha.jl.

# Requirements

## Host system

- Linux or OS X host system
- x86-64 processor with AVX2 instruction set
  - NNPACK is optimized for Intel Skylake, but can run on Haswell & Broadwell processors too
  - SSE2 instruction set can be targeted using `--backend=psimd` or `--backend=scalar` configuration options, but for performance reasons it is not recommended for production use
- ARMv7 processor with NEON instruction set
  - VFP instruction set (including ARMv6 systems with VFPv2) can be targeted using `--backend=scalar` configuration option, but for performance reasons it is not recommended for production use.

## Cross-compilation options:

- Android with x86/x86-64 (SSE2), ARMv7 with NEON, or ARM64 architecture
- WebAssembly for next-generation Web browsers
- Emscripten/Asm.js to run inside any modern Web browser
- Portable Native Client to run inside Google Chrome (no packaging required)
- Native Client (x86-64) to run as a packaged Google Chrome App

# Features

- Fast convolution algorithms based on Fourier transform and Winograd transform.
  - Forward propagation performance on Intel Core i7 6700K vs BVLC Caffe master branch as of March 24, 2016 (protobufs from convnet-benchmarks, integration via caffe-nnpack):

| Library | Caffe | NNPACK | NNPACK | NNPACK |
|---|---|---|---|---|
| **Algorithm** | **im2col + sgemm** | **FFT-8x8** | **FFT-16x16** | **Winograd F(6x6, 3x3)** |
| AlexNet:conv2 | 315 ms | 129 ms | **86** ms | N/A |
| AlexNet:conv3 | 182 ms | 87 ms | **44** ms | 70 ms |
| AlexNet:conv4 | 264 ms | 109 ms | **56** ms | 89 ms |
| AlexNet:conv5 | 177 ms | 77 ms | **40** ms | 64 ms |
| VGG-A:conv1 | **255** ms | 303 ms | 260 ms | 404 ms |
| VGG-A:conv2 | 902 ms | 369 ms | **267** ms | 372 ms |
| VGG-A:conv3.1 | 566 ms | 308 ms | **185** ms | 279 ms |
| VGG-A:conv3.2 | 1091 ms | 517 ms | **309** ms | 463 ms |
| VGG-A:conv4.1 | 432 ms | 228 ms | **149** ms | 188 ms |
| VGG-A:conv4.2 | 842 ms | 402 ms | **264** ms | 329 ms |
| VGG-A:conv5 | 292 ms | 141 ms | **83** ms | 114 ms |
| OverFeat:conv2 | 424 ms | 158 ms | **73** ms | N/A |
| OverFeat:conv3 | 250 ms | 69 ms | 74 ms | **54** ms |
| OverFeat:conv4 | 927 ms | 256 ms | 272 ms | **173** ms |
| OverFeat:conv5 | 1832 ms | 466 ms | 524 ms | **315** ms |

- Built-in expert-tuned kernels with very high performance:
  - Fast Fourier transform
  - Winograd transform

    - Matrix-matrix multiplication (GEMM)
    - Matrix-vector multiplication (GEMV)
    - Max-pooling.
- Multi-threaded SIMD-aware implementations of neural network layers.
- Implemented in C99 and Python without external dependencies.
- Extensive unit tests using C++ and Google Test.
- Supports Native Client target and outperforms native Caffe/CPU when running inside Chrome.

## Layers

- Convolutional layer
    - Training-optimized forward propagation ( `nnp_convolution_output` )
    - Training-optimized backward input gradient update ( `nnp_convolution_input_gradient` )
    - Training-optimized backward kernel gradient update ( `nnp_convolution_kernel_gradient` )
    - Inference-optimized forward propagation ( `nnp_convolution_inference` )
- Fully-connected layer
    - Training-optimized forward propagation ( `nnp_fully_connected_output` )
    - Inference-optimized forward propagation ( `nnp_fully_connected_inference` )
- Max pooling layer
    - Forward propagation, both for training and inference, ( `nnp_max_pooling_output` )
- ReLU layer (with parametrized negative slope)
    - Forward propagation, both for training and inference, optionally in-place, ( `nnp_relu_output` )
    - Backward input gradient update ( `nnp_relu_input_gradient` )
- Softmax layer
    - Forward propagation, both for training and inference, optionally in-place ( `nnp_softmax_output` )

## Building

NNPACK can be build on OS X and Linux.

Install ninja build system

```
sudo apt-get install ninja-build || brew install ninja
```

Install PeachPy assembler and confu configuration system

```
[sudo] pip install --upgrade git+https://github.com/Maratyszcza/PeachPy
[sudo] pip install --upgrade git+https://github.com/Maratyszcza/confu
```

Then clone NNPACK, install dependencies, configure, and build

```
git clone https://github.com/Maratyszcza/NNPACK.git
cd NNPACK
confu setup
python ./configure.py
ninja
```

### Cross-compilation for Android

- Download and setup Android NDK
- Add `ndk-build` to `PATH` variable
- Navigate to NNPACK directory and setup dependencies ( `confu setup` )
- Build NNPACK with `ndk-build` build system.

### Cross-compilation for Emscripten/WebAssembly

- Download and setup upstream version of Emscripten SDK

- Using `emsdk`, download, build and activate `incoming` version of Emscripten and Binaryen, and setup environment variables. `$EMSCRIPTEN` should specify the path to activated Emscripten environment.
- Configure NNPACK with `--target=wasm` option.

### Cross-compilation for Emscripten/Asm.js

- Download and setup Emscripten SDK
- Using `emsdk`, download, build and activate one of the environments, and setup environment variables. `$EMSCRIPTEN` should specify the path to activated Emscripten environment.
- Configure NNPACK with `--target=asmjs` option.

### Cross-compilation for Portable Native Client

- Download and setup Native Client SDK
- Set `NACL_SDK_ROOT` variable to a versioned SDK directory (e.g. `/opt/nacl_sdk/pepper_49`).
- Configure NNPACK with `--target=pnacl` option.

### Cross-compilation for Native Client

- Download and setup Native Client SDK
- Set `NACL_SDK_ROOT` variable to a versioned SDK directory (e.g. `/opt/nacl_sdk/pepper_49`).
- Configure NNPACK with `--target=x86_64-nacl-newlib` (recommended) or `--target=x86_64-nacl-gnu` option.

## Testing

NNPACK contains extensive test suite for transformation and neural network layers.

After configuration type `ninja smoketest` to run a set of quick tests, or `ninja test` to additionally NNPACK layers with parameters from AlexNet, VGG-A, and Overfeat-Fast networks (this will take a while).

## Packaging

Binary packages need to distribute two files: `include/nnpack.h` and `lib/libnnpack.a` (also `lib/libnnpack.so` or `lib/libnnpack.dylib` if NNPACK was configured with shared library support).

## Bindings

### Deep Learning Frameworks

- Caffe2 natively supports NNPACK
- MXNet - supports NNPACK for inference in convolutional layers, fully-connected, and max-pooling layers. See MXNet wiki for configuration instructions and performance benchmarks).
- szagoruyko/nnpack.torch - integration of NNPACK into Lua Torch via ffi
- tiny-dnn - header-only deep learning framework in C++11, which natively supports NNPACK.
- darknet-nnpack - fork of Darknet framework with NNPACK support.
- Maratyszcza/caffe - up-to-date integration of NNPACK (convolutional, fully-connected, max-pooling, and ReLU layers) into Caffe based on `nnpack-pr` branch in ajtulloch/caffe.
- Maratyszcza/caffe-nnpack - older and unmaintained integration of NNPACK (convolutional layers only) into Caffe.
- See also discussion in Issue #1

### Languages and Environments

- node-nnpack - Node.js bindings
- peterhj/libnnpack - Rust bindings

### Users

- Facebook uses NNPACK in production.
- Prisma uses NNPACK in the mobile app.

## Acknowledgements


The library is developed by Marat Dukhan of Georgia Tech with extensive advice from Nicolas Vasilache and Soumith Chintala of Facebook Artificial Intelligence Research. Andrew Tulloch of Facebook Artificial Intelligence Research contributed Caffe integration. We thank Andrew Lavin for fruitful discussions on Winograd transform-based implementations. NNPACK is a research project at Richard Vuduc's HPC Garage lab in the Georgia Institute of Technology, College of Computing, School of Computational Science and Engineering.

This material is based upon work supported by the U.S. National Science Foundation (NSF) Award Number 1339745. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of NSF.