# nupic.torch

***Release 0.0.1.dev0***

**Numenta**

**Apr 26, 2023**

# CONTENTS


**1 Installation** **3**
- 1.1 Test . . . 3

**2 Examples** **5**

**3 *Having problems?*** **7**

**4 Contents** **9**
- 4.1 nupic . . . 9
- 4.2 License . . . 17
- 4.3 TODO . . . 30

**5 Indices and tables** **31**

**Python Module Index** **33**

**Index** **35**

This library integrates selected neuroscience principles from Hierarchical Temporal Memory (HTM) into the pytorch deep learning platform. The current code aims to replicate how sparsity is enforced via Spatial Pooling, as defined in the paper *How Could We Be So Dense? The Benefits of Using Highly Sparse Representations*.

For detail on the neuroscience behind these theories, read Why Neurons Have Thousands of Synapses, A Theory of Sequence Memory in Neocortex. For a description of *Spatial Pooling* in isolation, read *Spatial Pooling (BAMI)*.

`nupic.torch` is named after the original HTM library, the Numenta Platform for Intelligent Computing (*NuPIC*).

Interested in *contributing*?

CHAPTER

ONE

# INSTALLATION

To install from local source code:

```
pip install -e .
```

Or using conda:

```
conda env create
```

## 1.1 Test

To run all tests:

```
pytest
```

CHAPTER

TWO

# EXAMPLES

We've created a few jupyter notebooks demonstrating how to use **nupic.torch** with standard datasets. You can find these notebooks in the examples/ directory or if you prefer you can open them in Google Colab and start experimenting.

# CHAPTER THREE

# HAVING PROBLEMS?

For any installation issues, please search our forums (post questions there). Report bugs here.

CHAPTER

FOUR

# CONTENTS

## 4.1 nupic

### 4.1.1 nupic namespace

#### nupic.torch package

#### nupic.torch.functions package

#### nupic.torch.functions.k_winners

**kwinners**(*x*, *duty_cycles*, *k: int*, *boost_strength: float*, *break_ties: bool = False*, *relu: bool = False*, *inplace: bool = False*)

A simple K-winner take all function for creating layers with sparse output.

Use the boost strength to compute a boost factor for each unit represented in x. These factors are used to increase the impact of each unit to improve their chances of being chosen. This encourages participation of more columns in the learning process.

The boosting function is a curve defined as:

$$boostFactors = \exp(-boostStrength \times (dutyCycles - targetDensity))$$

Intuitively this means that units that have been active (i.e. in the top-k) at the target activation level have a boost factor of 1, meaning their activity is not boosted. Columns whose duty cycle drops too much below that of their neighbors are boosted depending on how infrequently they have been active. Unit that has been active more than the target activation level have a boost factor below 1, meaning their activity is suppressed and they are less likely to be in the top-k.

Note that we do not transmit the boosted values. We only use boosting to determine the winning units.

The target activation density for each unit is k / number of units. The boostFactor depends on the duty_cycles via an exponential function:

```
boostFactor
    ^
    |
    |               |          1  _  |                |    _
    |     _ _
    |        _ _ _ _
    +--------------------> duty_cycles
```

(continues on next page)

(continued from previous page)

```
      |
target_density
```

**Parameters**

- **x** – Current activity of each unit, optionally batched along the 0th dimension.
- **duty_cycles** – The averaged duty cycle of each unit.
- **k** – The activity of the top k units will be allowed to remain, the rest are set to zero.
- **boost_strength** – A boost strength of 0.0 has no effect on x.
- **break_ties** – Whether to use a strict k-winners. Using break_ties=False is faster but may occasionally result in more than k active units.
- **relu** – Whether to simulate the effect of applying ReLU before KWinners
- **inplace** – Whether to modify x in place

**Returns**
A tensor representing the activity of x after k-winner take all.

**kwinners2d**(*x*, *duty_cycles*, *k: int*, *boost_strength: float*, *local: bool = True*, *break_ties: bool = False*, *relu: bool = False*, *inplace: bool = False*)

A K-winner take all function for creating Conv2d layers with sparse output.

If local=True, k-winners are chosen independently for each location. For Conv2d inputs (batch, channel, H, W), the top k channels are selected locally for each of the H X W locations. If there is a tie for the kth highest boosted value, there will be more than k winners.

The boost strength is used to compute a boost factor for each unit represented in x. These factors are used to increase the impact of each unit to improve their chances of being chosen. This encourages participation of more columns in the learning process. See `kwinners()` for more details.

**Parameters**

- **x** – Current activity of each unit.
- **duty_cycles** – The averaged duty cycle of each unit.
- **k** – The activity of the top k units across the channels will be allowed to remain, the rest are set to zero.
- **boost_strength** – A boost strength of 0.0 has no effect on x.
- **local** – Whether or not to choose the k-winners locally (across the channels at each location) or globally (across the whole input and across all channels).
- **break_ties** – Whether to use a strict k-winners. Using break_ties=False is faster but may occasionally result in more than k active units.
- **relu** – Whether to simulate the effect of applying ReLU before KWinners.
- **inplace** – Whether to modify x in place

**Returns**
A tensor representing the activity of x after k-winner take all.

### nupic.torch.models package

#### nupic.torch.models.sparse_cnn

**class GSCSparseCNN**(**args: Any*, ***kwargs: Any*)

Bases: `Sequential`

Sparse CNN model used to classify *Google Speech Commands* dataset as described in How Can We Be So Dense? paper.

**Parameters**

- **`cnn_out_channels`** – output channels for each CNN layer
- **`cnn_percent_on`** – Percent of units allowed to remain on each convolution layer
- **`linear_units`** – Number of units in the linear layer
- **`linear_percent_on`** – Percent of units allowed to remain on the linear layer
- **`k_inference_factor`** – During inference (training=False) we increase *percent_on* in all sparse layers by this factor
- **`boost_strength`** – boost strength (0.0 implies no boosting)
- **`boost_strength_factor`** – Boost strength factor to use [0..1]
- **`duty_cycle_period`** – The period used to calculate duty cycles
- **`kwinner_local`** – Whether or not to choose the k-winners locally (across the channels at each location) or globally (across the whole input and across all channels)
- **`cnn_sparsity`** – Percent of weights that zero
- **`linear_sparsity`** – Percent of weights that are zero in the linear layer.

**class GSCSuperSparseCNN**(**args: Any*, ***kwargs: Any*)

Bases: *`GSCSparseCNN`*

Super Sparse CNN model used to classify *Google Speech Commands* dataset as described in How Can We Be So Dense? paper. This model provides a sparser version of *`GSCSparseCNN`*

**class MNISTSparseCNN**(**args: Any*, ***kwargs: Any*)

Bases: `Sequential`

Sparse CNN model used to classify *MNIST* dataset as described in How Can We Be So Dense? paper.

**Parameters**

- **`cnn_out_channels`** – output channels for each CNN layer
- **`cnn_percent_on`** – Percent of units allowed to remain on each convolution layer
- **`linear_units`** – Number of units in the linear layer
- **`linear_percent_on`** – Percent of units allowed to remain on the linear layer
- **`k_inference_factor`** – During inference (training=False) we increase *percent_on* in all sparse layers by this factor
- **`boost_strength`** – boost strength (0.0 implies no boosting)
- **`boost_strength_factor`** – Boost strength factor to use [0..1]
- **`duty_cycle_period`** – The period used to calculate duty cycles

- **kwinner_local** – Whether or not to choose the k-winners locally (across the channels at each location) or globally (across the whole input and across all channels)
- **cnn_sparsity** – Percent of weights that are zero
- **linear_sparsity** – Percent of weights that are zero.

**gsc_sparse_cnn**(*pretrained=False*, *progress=True*, ***kwargs*)

Sparse CNN model used to classify 'Google Speech Commands' dataset

**Parameters**

- **pretrained** – If True, returns a model pre-trained on Google Speech Commands
- **progress** – If True, displays a progress bar of the download to stderr
- **kwargs** – See *GSCSparseCNN*

**gsc_super_sparse_cnn**(*pretrained=False*, *progress=True*)

Super Sparse CNN model used to classify *Google Speech Commands* dataset as described in How Can We Be So Dense? paper. This model provides a sparser version of *GSCSparseCNN*

**Parameters**

- **pretrained** – If True, returns a model pre-trained on Google Speech Commands
- **progress** – If True, displays a progress bar of the download to stderr

## nupic.torch.modules package

### nupic.torch.modules.flatten

**class Flatten**(**args: Any*, ***kwargs: Any*)

Bases: Module

Flatten input retaining batch dimension.

**forward**(*x*)

### nupic.torch.modules.k_winners

**class KWinners**(**args: Any*, ***kwargs: Any*)

Bases: *KWinnersBase*

Applies K-Winner function to the input tensor.

See `htmresearch.frameworks.pytorch.functions.k_winners`

**Parameters**

- **n** (*int*) – Number of units
- **percent_on** (*float*) – The activity of the top k = percent_on * n will be allowed to remain, the rest are set to zero.
- **k_inference_factor** (*float*) – During inference (training=False) we increase percent_on by this factor. percent_on * k_inference_factor must be strictly less than 1.0, ideally much lower than 1.0
- **boost_strength** (*float*) – boost strength (0.0 implies no boosting).

- **boost_strength_factor** (*float*) – Boost strength factor to use [0..1]
- **duty_cycle_period** (*int*) – The period used to calculate duty cycles
- **break_ties** (*bool*) – Whether to use a strict k-winners. Using break_ties=False is faster but may occasionally result in more than k active units.
- **relu** (*bool*) – This will simulate the effect of having a ReLU before the KWinners.
- **inplace** (*bool*) – Modify the input in-place.

**extra_repr**()

**forward**(*x*)

**update_duty_cycle**(*x*)

Updates our duty cycle estimates with the new value. Duty cycles are updated according to the following formula:

$$dutyCycle = \frac{dutyCycle \times (period - batchSize) + newValue}{period}$$

**Parameters**

**x** – Current activity of each unit

**class KWinners2d**(**args: Any*, ***kwargs: Any*)

Bases: `KWinnersBase`

Applies K-Winner function to the input tensor.

See `htmresearch.frameworks.pytorch.functions.k_winners2d`

**Parameters**

- **channels** (*int*) – Number of channels (filters) in the convolutional layer.
- **percent_on** (*float*) – The activity of the top k = percent_on * number of input units will be allowed to remain, the rest are set to zero.
- **k_inference_factor** (*float*) – During inference (training=False) we increase percent_on by this factor. percent_on * k_inference_factor must be strictly less than 1.0, ideally much lower than 1.0
- **boost_strength** (*float*) – boost strength (0.0 implies no boosting).
- **boost_strength_factor** (*float*) – Boost strength factor to use [0..1]
- **duty_cycle_period** (*int*) – The period used to calculate duty cycles
- **local** (*bool*) – Whether or not to choose the k-winners locally (across the channels at each location) or globally (across the whole input and across all channels).
- **break_ties** (*bool*) – Whether to use a strict k-winners. Using break_ties=False is faster but may occasionally result in more than k active units.
- **relu** (*bool*) – This will simulate the effect of having a ReLU before the KWinners.
- **inplace** (*bool*) – Modify the input in-place.

**entropy**()

Returns the current total entropy of this layer.

**extra_repr**()

**forward**(*x*)

**update_duty_cycle**(*x*)

Updates our duty cycle estimates with the new value. Duty cycles are updated according to the following formula:

$$dutyCycle = \frac{dutyCycle \times (period - batchSize) + newValue}{period}$$

**Parameters**
**x** – Current activity of each unit

**class KWinnersBase**(**args: Any*, ***kwargs: Any*)

Bases: `Module`

Base KWinners class.

**Parameters**

- **percent_on** (*float*) – The activity of the top k = percent_on * number of input units will be allowed to remain, the rest are set to zero.
- **k_inference_factor** (*float*) – During inference (training=False) we increase percent_on by this factor. percent_on * k_inference_factor must be strictly less than 1.0, ideally much lower than 1.0
- **boost_strength** (*float*) – boost strength (0.0 implies no boosting). Must be >= 0.0
- **boost_strength_factor** (*float*) – Boost strength factor to use [0..1]
- **duty_cycle_period** (*int*) – The period used to calculate duty cycles

**entropy**()

Returns the current total entropy of this layer.

**extra_repr**()

**max_entropy**()

Returns the maximum total entropy we can expect from this layer.

**update_boost_strength**()

Update boost strength by multiplying by the boost strength factor. This is typically done during training at the beginning of each epoch.

**abstract update_duty_cycle**(*x*)

Updates our duty cycle estimates with the new value. Duty cycles are updated according to the following formula:

$$dutyCycle = \frac{dutyCycle \times (period - batchSize) + newValue}{period}$$

**Parameters**
**x** – Current activity of each unit

**update_boost_strength**(*m*)

Function used to update KWinner modules boost strength. This is typically done during training at the beginning of each epoch.

Call using `torch.nn.Module.apply()` after each epoch if required For example: `m.apply(update_boost_strength)`

**Parameters**
**m** – KWinner module

### nupic.torch.modules.prunable_sparse_weights

**class PrunableSparseWeightBase**

Bases: `object`

Enable easy setting and getting of the off-mask that defines which weights are zero.

**property off_mask**

Gets the value of *zero_mask* in bool format. Thus one may call ` self.weight[~self.off_mask] # returns weights that are currently on `

**class PrunableSparseWeights**(**args: Any*, ***kwargs: Any*)

Bases: *SparseWeights*, *PrunableSparseWeightBase*

Enforce weight sparsity on linear module. The off-weights may be changed dynamically through the *off_mask* property.

**class PrunableSparseWeights2d**(**args: Any*, ***kwargs: Any*)

Bases: *SparseWeights2d*, *PrunableSparseWeightBase*

Enforce weight sparsity on CNN modules. The off-weights may be changed dynamically through the *off_mask* property.

### nupic.torch.modules.sparse_weights

**class HasRezeroWeights**

Bases: `object`

**abstract rezero_weights()**

Set the previously selected weights to zero.

**class SparseWeights**(**args: Any*, ***kwargs: Any*)

Bases: *SparseWeightsBase*

Enforce weight sparsity on linear module during training.

Sample usage:

model = nn.Linear(784, 10) model = SparseWeights(model, sparsity=0.4)

**Parameters**

- **module** – The module to sparsify the weights
- **weight_sparsity** – Pct of weights that are NON-ZERO in the layer. Also equal to 1-sparsity **Please note this is the first positional parameter for backwards compatibility**
- **sparsity** – Pct of weights that are ZERO in the layer Accepts either sparsity or weight_sparsity, but not both at a time
- **allow_extremes** – Allow values sparsity=0 and sparsity=1. These values are often a sign that there is a bug in the configuration, because they lead to Identity and Zero layers, respectively, but they can make sense in scenarios where the mask is dynamic.

**rezero_weights()**

Set the previously selected weights to zero.

**class SparseWeights2d**(**args: Any*, ***kwargs: Any*)

Bases: *SparseWeightsBase*

Enforce weight sparsity on CNN modules Sample usage:

model = nn.Conv2d(in_channels, out_channels, kernel_size, ...) model = SparseWeights2d(model, sparsity=0.4)

**Parameters**

- **module** – The module to sparsify the weights
- **weight_sparsity** – Pct of weights that are NON-ZERO in the layer. Also equal to 1-sparsity **Please note this is the first positional parameter for backwards compatibility**
- **sparsity** – Pct of weights that are ZERO in the layer Accepts either sparsity or weight_sparsity, but not both at a time
- **allow_extremes** – Allow values sparsity=0 and sparsity=1. These values are often a sign that there is a bug in the configuration, because they lead to Identity and Zero layers, respectively, but they can make sense in scenarios where the mask is dynamic.

**rezero_weights**()

Set the previously selected weights to zero.

**class SparseWeightsBase**(**args: Any*, ***kwargs: Any*)

Bases: `Module`, *HasRezeroWeights*

Base class for the all Sparse Weights modules.

**Parameters**

- **module** – The module to sparsify the weights
- **weight_sparsity** – Pct of weights that are NON-ZERO in the layer. Also equal to 1-sparsity **Please note this is the first positional parameter for backwards compatibility**
- **sparsity** – Pct of weights that are ZERO in the layer Accepts either sparsity or weight_sparsity, but not both at a time

**extra_repr**()

**forward**(*x*)

**property bias**

**property weight**

**property weight_sparsity**

**normalize_sparse_weights**(*m*)

Initialize the weights using kaiming_uniform initialization normalized to the number of non-zeros in the layer instead of the whole input size.

Similar to torch.nn.Linear.reset_parameters() but applying weight sparsity to the input size

**rezero_weights**(*m*)

Function used to update the weights after each epoch.

Call using `torch.nn.Module.apply()` after each epoch if required For example: `m.apply(rezero_weights)`

**Parameters**
**m** – HasRezeroWeights module

### nupic.torch.compatibility

**upgrade_to_masked_sparseweights**(*state_dict*)
Returns a new state dict with any "zero_weights" tensors converted to "zero_mask" tensors. (The "zero_weights" was a list of indices of zeroes in the weight tensor.)

### nupic.torch.duty_cycle_metrics

**binary_entropy**(*x*)
Calculate entropy for a list of binary random variables.

**Parameters**
**x** – (torch tensor) the probability of the variable to be 1.

**Returns**
entropy: (torch tensor) entropy, sum(entropy)

**max_entropy**(*n*, *k*)
The maximum entropy we could get with n units and k winners.

## 4.2 License

```
                    GNU AFFERO GENERAL PUBLIC LICENSE
                       Version 3, 19 November 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU Affero General Public License is a free, copyleft license for
software and other kinds of works, specifically designed to ensure
cooperation with the community in the case of network server software.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
our General Public Licenses are intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.
```

(continues on next page)

(continued from previous page)

```
  Developers that use our General Public Licenses protect your rights
with two steps: (1) assert copyright on the software, and (2) offer
you this License which gives you legal permission to copy, distribute
and/or modify the software.

  A secondary benefit of defending all users' freedom is that
improvements made in alternate versions of the program, if they
receive widespread use, become available for other developers to
incorporate.  Many developers of free software are heartened and
encouraged by the resulting cooperation.  However, in the case of
software used on network servers, this result may fail to come about.
The GNU General Public License permits making a modified version and
letting the public access it on a server without ever releasing its
source code to the public.

  The GNU Affero General Public License is designed specifically to
ensure that, in such cases, the modified source code becomes available
to the community.  It requires the operator of a network server to
provide the source code of the modified version running there to the
users of that server.  Therefore, public use of a modified version, on
a publicly accessible server, gives the public access to the source
code of the modified version.

  An older license, called the Affero General Public License and
published by Affero, was designed to accomplish similar goals.  This is
a different license, not a version of the Affero GPL, but Affero has
released a new version of the Affero GPL which permits relicensing under
this license.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU Affero General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
```

(continues on next page)

(continued from previous page)

```
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
```

(continues on next page)

(continued from previous page)

```
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.
```

(continues on next page)

(continued from previous page)

```
  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.
```

(continues on next page)

(continued from previous page)

```
  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
```

(continues on next page)

(continued from previous page)

```
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
```

(continues on next page)

(continued from previous page)

```
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.
```

(continues on next page)

(continued from previous page)

```
  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
```

(continues on next page)

(continued from previous page)

```
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
```

(continues on next page)

(continued from previous page)

```
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Remote Network Interaction; Use with the GNU General Public License.

  Notwithstanding any other provision of this License, if you modify the
Program, your modified version must prominently offer all users
interacting with it remotely through a computer network (if your version
supports such interaction) an opportunity to receive the Corresponding
```

(continues on next page)

(continued from previous page)

```
Source of your version by providing access to the Corresponding Source
from a network server at no charge, through some standard or customary
means of facilitating copying of software.  This Corresponding Source
shall include the Corresponding Source for any work covered by version 3
of the GNU General Public License that is incorporated pursuant to the
following paragraph.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the work with which it is combined will remain governed by version
3 of the GNU General Public License.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU Affero General Public License from time to time.  Such new versions
will be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU Affero General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU Affero General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU Affero General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
```

(continues on next page)

(continued from previous page)

```
  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as published
    by the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.
```

(continues on next page)

(continued from previous page)

```
  If your software can interact with users remotely through a computer
network, you should also make sure that it provides a way for users to
get its source.  For example, if your program is a web application, its
interface could display a "Source" link that leads users to an archive
of the code.  There are many ways you could offer source, and different
solutions will be better for different programs; see section 13 for the
specific requirements.

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU AGPL, see
<https://www.gnu.org/licenses/>.
```

## 4.3 TODO

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## n

nupic, 9
nupic.torch, 9
nupic.torch.compatibility, 17
nupic.torch.duty_cycle_metrics, 17
nupic.torch.functions, 9
nupic.torch.functions.k_winners, 9
nupic.torch.models, 11
nupic.torch.models.sparse_cnn, 11
nupic.torch.modules, 12
nupic.torch.modules.flatten, 12
nupic.torch.modules.k_winners, 12
nupic.torch.modules.prunable_sparse_weights, 15
nupic.torch.modules.sparse_weights, 15

# INDEX


## B

bias (*SparseWeightsBase property*), 16
binary_entropy() (*in module nupic.torch.duty_cycle_metrics*), 17

## E

entropy() (*KWinners2d method*), 13
entropy() (*KWinnersBase method*), 14
extra_repr() (*KWinners method*), 13
extra_repr() (*KWinners2d method*), 13
extra_repr() (*KWinnersBase method*), 14
extra_repr() (*SparseWeightsBase method*), 16

## F

Flatten (*class in nupic.torch.modules.flatten*), 12
forward() (*Flatten method*), 12
forward() (*KWinners method*), 13
forward() (*KWinners2d method*), 13
forward() (*SparseWeightsBase method*), 16

## G

gsc_sparse_cnn() (*in module nupic.torch.models.sparse_cnn*), 12
gsc_super_sparse_cnn() (*in module nupic.torch.models.sparse_cnn*), 12
GSCSparseCNN (*class in nupic.torch.models.sparse_cnn*), 11
GSCSuperSparseCNN (*class in nupic.torch.models.sparse_cnn*), 11

## H

HasRezeroWeights (*class in nupic.torch.modules.sparse_weights*), 15

## K

KWinners (*class in nupic.torch.modules.k_winners*), 12
kwinners() (*in module nupic.torch.functions.k_winners*), 9
KWinners2d (*class in nupic.torch.modules.k_winners*), 13
kwinners2d() (*in module nupic.torch.functions.k_winners*), 10
KWinnersBase (*class in nupic.torch.modules.k_winners*), 14

## M

max_entropy() (*in module nupic.torch.duty_cycle_metrics*), 17
max_entropy() (*KWinnersBase method*), 14
MNISTSparseCNN (*class in nupic.torch.models.sparse_cnn*), 11
module
- nupic, 9
- nupic.torch, 9
- nupic.torch.compatibility, 17
- nupic.torch.duty_cycle_metrics, 17
- nupic.torch.functions, 9
- nupic.torch.functions.k_winners, 9
- nupic.torch.models, 11
- nupic.torch.models.sparse_cnn, 11
- nupic.torch.modules, 12
- nupic.torch.modules.flatten, 12
- nupic.torch.modules.k_winners, 12
- nupic.torch.modules.prunable_sparse_weights, 15
- nupic.torch.modules.sparse_weights, 15

## N

normalize_sparse_weights() (*in module nupic.torch.modules.sparse_weights*), 16
nupic
- module, 9

nupic.torch
- module, 9

nupic.torch.compatibility
- module, 17

nupic.torch.duty_cycle_metrics
- module, 17

nupic.torch.functions
- module, 9

nupic.torch.functions.k_winners
- module, 9

nupic.torch.models
    module, 11
nupic.torch.models.sparse_cnn
    module, 11
nupic.torch.modules
    module, 12
nupic.torch.modules.flatten
    module, 12
nupic.torch.modules.k_winners
    module, 12
nupic.torch.modules.prunable_sparse_weights
    module, 15
nupic.torch.modules.sparse_weights
    module, 15

## O

off_mask (*PrunableSparseWeightBase property*), 15

## P

PrunableSparseWeightBase (*class in nupic.torch.modules.prunable_sparse_weights*), 15
PrunableSparseWeights (*class in nupic.torch.modules.prunable_sparse_weights*), 15
PrunableSparseWeights2d (*class in nupic.torch.modules.prunable_sparse_weights*), 15

## R

rezero_weights() (*HasRezeroWeights method*), 15
rezero_weights() (*in module nupic.torch.modules.sparse_weights*), 16
rezero_weights() (*SparseWeights method*), 15
rezero_weights() (*SparseWeights2d method*), 16

## S

SparseWeights (*class in nupic.torch.modules.sparse_weights*), 15
SparseWeights2d (*class in nupic.torch.modules.sparse_weights*), 15
SparseWeightsBase (*class in nupic.torch.modules.sparse_weights*), 16

## U

update_boost_strength() (*in module nupic.torch.modules.k_winners*), 14
update_boost_strength() (*KWinnersBase method*), 14
update_duty_cycle() (*KWinners method*), 13
update_duty_cycle() (*KWinners2d method*), 14
update_duty_cycle() (*KWinnersBase method*), 14
upgrade_to_masked_sparseweights() (*in module nupic.torch.compatibility*), 17

## W

weight (*SparseWeightsBase property*), 16
weight_sparsity (*SparseWeightsBase property*), 16