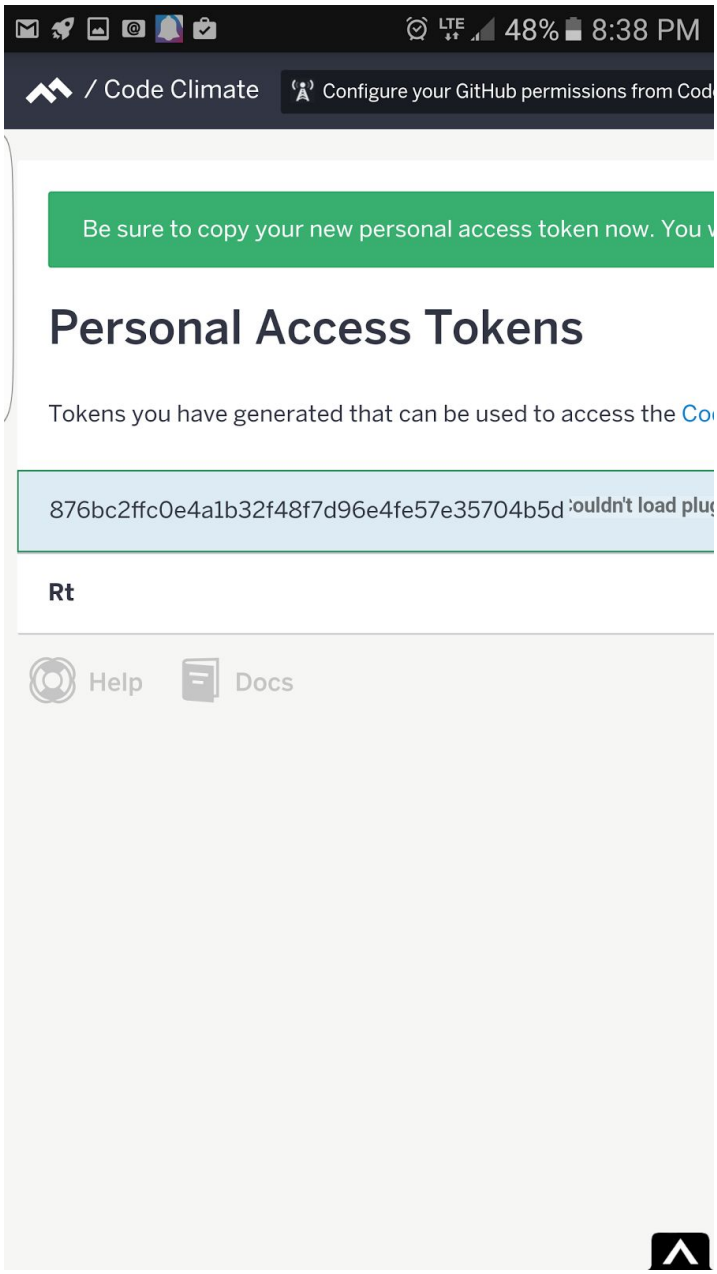
/ pytorch/pytorch

- Cneiman



- Pytorch/pytorch

- List
- Source
- All Issues17
- Complexity4
- Duplication13

-
- **Cyclomatic complexity is too high in method _checkOutputSize. (10)**
  Cyclomatic complexity is too high in method _checkOutputSize. (10)
  Cyclomatic Complexity
  Cyclomatic Complexity corresponds to the number of decisions a block of code contains plus 1. This number (also called McCabe number) is equal to the number of linearly independent paths through the code. This number can be used as a guide when testing conditional logic in blocks.
  Radon analyzes the AST tree of a Python program to compute Cyclomatic Complexity. Statements have the following effects on Cyclomatic Complexity:

Construct	Effect on CC	Reasoning
if	+1	An *if* statement is a single decision.
elif	+1	The *elif* statement adds another decision.
else	+0	The *else* statement does not cause a new decision. The decision is at the *if*.
for	+1	There is a decision at the start of the loop.
while	+1	There is a decision at the *while* statement.
except	+1	Each *except* branch adds a new conditional path of execution.
finally	+0	The finally block is unconditionally executed.
with	+1	The *with* statement roughly corresponds to a try/except block (see PEP 343 for details).

assert	+1	The *assert* statement internally roughly equals a conditional statement.
Comprehension	+1	A list/set/dict comprehension of generator expression is equivalent to a for loop.
Boolean Operator	+1	Every boolean operator (and, or) adds a decision point.

- Source: http://radon.readthedocs.org/en/latest/intro.html

  - Disable radon
  - Disable this check
  - Ignore this issue

```
    def _checkOutputSize(self, input, output):
        if output.ndimension() != input.ndimension():
            raise RuntimeError('inconsistent dimension between
output and input.')

        if output.ndimension() == 3:
```

View moreFound by radon

-
- **Cyclomatic complexity is too high in method _checkInputSize. (9)**
  Cyclomatic complexity is too high in method _checkInputSize. (9)
  Cyclomatic Complexity
  Cyclomatic Complexity corresponds to the number of decisions a block of code contains plus 1. This number (also called McCabe number) is equal to the number of linearly independent paths through the code. This number can be used as a guide when testing conditional logic in blocks.
  Radon analyzes the AST tree of a Python program to compute Cyclomatic Complexity. Statements have the following effects on Cyclomatic Complexity:

Construct	Effect on CC	Reasoning

if	+1	An *if* statement is a single decision.
elif	+1	The *elif* statement adds another decision.
else	+0	The *else* statement does not cause a new decision. The decision is at the *if*.
for	+1	There is a decision at the start of the loop.
while	+1	There is a decision at the *while* statement.
except	+1	Each *except* branch adds a new conditional path of execution.
finally	+0	The finally block is unconditionally executed.
with	+1	The *with* statement roughly corresponds to a try/except block (see PEP 343 for details).
assert	+1	The *assert* statement internally roughly equals a conditional statement.
Comprehension	+1	A list/set/dict comprehension of generator expression is equivalent to a for loop.
Boolean Operator	+1	Every boolean operator (and, or) adds a decision point.

- Source: http://radon.readthedocs.org/en/latest/intro.html

  - Disable radon
  - Disable this check
  - Ignore this issue

```
    def _checkInputSize(self, input):
        if input.ndimension() == 3:
            if input.size(0) != self.nInputPlane or
input.size(1) != self.iH or input.size(1) != self.iW:
                raise RuntimeError(
                    'Given input size: ({}x{}x{}) inconsistent
with expected input size: ({}x{}x{}).'.format(
```

View moreFound by radon

-
- **Cyclomatic complexity is too high in method tostring. (7)**

Cyclomatic complexity is too high in method __tostring__. (7)

Cyclomatic Complexity

Cyclomatic Complexity corresponds to the number of decisions a block of code contains plus 1. This number (also called McCabe number) is equal to the number of linearly independent paths through the code. This number can be used as a guide when testing conditional logic in blocks.

-
-
-

Radon analyzes the AST tree of a Python program to compute Cyclomatic Complexity. Statements have the following effects on Cyclomatic Complexity:

Construct	Effect on CC	Reasoning
if	+1	An *if* statement is a single decision.
elif	+1	The *elif* statement adds another decision.
else	+0	The *else* statement does not cause a new decision. The decision is at the *if*.
for	+1	There is a decision at the start of the loop.
while	+1	There is a decision at the *while* statement.
except	+1	Each *except* branch adds a new conditional path of execution.
finally	+0	The finally block is unconditionally executed.
with	+1	The *with* statement roughly corresponds to a try/except block (see PEP 343 for details).
assert	+1	The *assert* statement internally roughly equals a

		conditional statement.
Comprehension	+1	A list/set/dict comprehension of generator expression is equivalent to a for loop.
Boolean Operator	+1	Every boolean operator (and, or) adds a decision point.

- Source: http://radon.readthedocs.org/en/latest/intro.html

  - Disable radon
  - Disable this check
  - Ignore this issue

```
    def __tostring__(self, ):
        s = super(SpatialConvolution, self).__repr__()
        s += '({} -> {}, {}x{}, {}x{}'.format(self.nInputPlane,
self.nOutputPlane, self.iW, self.iH, self.kW, self.kH)
        if self.dW != 1 or self.dH != 1 or self.padW != 0 or
self.padH != 0:
            s += ', {}, {}'.format(self.dW, self.dH)
```

View moreFound by radon

-
- **Cyclomatic complexity is too high in method _makeContiguous. (6)**
  Cyclomatic complexity is too high in method _makeContiguous. (6)
  Cyclomatic Complexity
  Cyclomatic Complexity corresponds to the number of decisions a block of code contains plus 1. This number (also called McCabe number) is equal to the number of linearly independent paths through the code. This number can be used as a guide when testing conditional logic in blocks.
  Radon analyzes the AST tree of a Python program to compute Cyclomatic Complexity. Statements have the following effects on Cyclomatic Complexity:

Construct	Effect on CC	Reasoning
if	+1	An *if* statement is a single

		decision.
elif	+1	The *elif* statement adds another decision.
else	+0	The *else* statement does not cause a new decision. The decision is at the *if*.
for	+1	There is a decision at the start of the loop.
while	+1	There is a decision at the *while* statement.
except	+1	Each *except* branch adds a new conditional path of execution.
finally	+0	The finally block is unconditionally executed.
with	+1	The *with* statement roughly corresponds to a try/except block (see PEP 343 for details).
assert	+1	The *assert* statement internally roughly equals a conditional statement.
Comprehension	+1	A list/set/dict comprehension of generator expression is equivalent to a for loop.
Boolean Operator	+1	Every boolean operator (and, or) adds a decision point.

- Source: http://radon.readthedocs.org/en/latest/intro.html

  - Disable radon
  - Disable this check
  - Ignore this issue

```
def _makeContiguous(self, input, gradOutput=None):
    if not input.is_contiguous():
        if self._input is None:
            self._input = input.new()
        self._input.resize_as_(input).copy_(input)
```

View moreFound by radon

- # Similar code found in 1 other location (mass = 232)
  # Similar code found in 1 other location (mass = 232)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).

Tuning

This issue has a mass of 232.

We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.

The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.

If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.

See codeclimate-duplication's documentation

for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
        if input.ndimension() == 3:
            if input.size(0) != self.nInputPlane or
input.size(1) != self.iH or input.size(1) != self.iW:
                raise RuntimeError(
                    'Given input size: ({}x{}x{})
inconsistent with expected input size: ({}x{}x{}).'.format(
                        input.size(0), input.size(1),
input.size(2), self.nInputPlane, self.iH, self.iW))
```

View moreAlso found in 1 other location

- torch/legacy/nn/SpatialConvolutionLocal.py:92…104

Similar code found in 1 other location (mass = 232)

Similar code found in 1 other location (mass = 232)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a

single, unambiguous, authoritative representation within a system.
When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
Tuning
This issue has a mass of 232.
We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.
Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object

- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication
- Disable this check
- Ignore this issue

```
        if output.ndimension() == 3:
            if output.size(0) != self.nOutputPlane or
output.size(1) != self.oH or output.size(2) != self.oW:
                raise RuntimeError(
                    'Given output size: ({}x{}x{})
inconsistent with expected output size: ({}x{}x{}).'.format(
                        output.size(0), output.size(1),
output.size(2), self.nOutputPlane, self.oH, self.oW))
```

View moreAlso found in 1 other location

- torch/legacy/nn/SpatialConvolutionLocal.py:75…86

Identical code found in 4 other locations (mass = 147)

Identical code found in 4 other locations (mass = 147)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate

and also to diverge (leaving bugs as two similar implementations differ in subtle ways).

Tuning

This issue has a mass of 147.

We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.

The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.

If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.

See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking

- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76
- Disable duplication
- Disable this check
- Ignore this issue

```
    def _makeContiguous(self, input, gradOutput=None):
        if not input.is_contiguous():
            if self._input is None:
                self._input = input.new()
            self._input.resize_as_(input).copy_(input)
```

View moreAlso found in 4 other locations

- torch/legacy/nn/SpatialConvolution.py:48…63
- torch/legacy/nn/SpatialFullConvolution.py:56…71
- torch/legacy/nn/VolumetricConvolution.py:42…57
- torch/legacy/nn/VolumetricFullConvolution.py:58…73

- **Similar code found in 1 other location (mass = 72)**
  Similar code found in 1 other location (mass = 72)
  Duplicated Code
  Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
  When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
  Tuning
  This issue has a mass of 72.
  We set useful threshold defaults for the

languages we support but you may want to adjust these settings based on your project guidelines.
The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
    def type(self, type=None, tensorCache=None):
        if self.finput is not None:
            self.finput = torch.Tensor()
```

```
            if self.fgradInput is not None:
                self.fgradInput = torch.Tensor()
            return super(SpatialConvolutionLocal,
self).type(type, tensorCache)
```

Also found in [1 other location]

- torch/legacy/nn/[SpatialFullConvolution.py:191…196]

- **Identical code found in 1 other location (mass = 63)**
  Identical code found in 1 other location (mass = 63)
  Duplicated Code
  Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
  When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
  Tuning
  This issue has a mass of 63.
  We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
  The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the

threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
        if stdv is not None:
            stdv = stdv * math.sqrt(3)
        else:
            stdv = 1. / math.sqrt(self.kW * self.kH *
self.nInputPlane)
```

Also found in 1 other location

- torch/legacy/nn/SpatialConvolution.py:39…42

Identical code found in 3 other locations (mass = 61)

Identical code found in 3 other locations (mass = 61)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).

Tuning

This issue has a mass of 61.

We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.

The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.

If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends

to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.
Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
        if self.dW != 1 or self.dH != 1 or self.padW != 0 or self.padH != 0:
            s += ', {}, {}'.format(self.dW, self.dH)
```

Also found in 3 other locations

- torch/legacy/nn/SpatialConvolution.py:152…153
- torch/legacy/nn/SpatialDilatedConvolution.py:77…78
- torch/legacy/nn/SpatialFullConvolution.py:201…202

Similar code found in 1 other location (mass = 53)

Similar code found in 1 other location (mass = 53)

Duplicated Code
Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle

states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
Tuning
This issue has a mass of 53.
We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.
Refactorings

- Extract Method
- Extract Class

- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
        self.oW = int(math.floor((self.padW * 2 + iW - self.kW) / self.dW)) + 1
```

Also found in 1 other location

- torch/legacy/nn/SpatialConvolutionLocal.py:24

Similar code found in 1 other location (mass = 53)

Similar code found in 1 other location (mass = 53)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).

Tuning

This issue has a mass of 53.
We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

- Disable duplication
- Disable this check

- Ignore this issue
- `self.oH = int(math.floor((self.padH * 2 + iH - self.kH) / self.dH)) + 1`
  Also found in 1 other location
  - torch/legacy/nn/SpatialConvolutionLocal.py:23
-

- **Similar code found in 1 other location (mass = 47)**
  Similar code found in 1 other location (mass = 47)
  Duplicated Code
  Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
  When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
  Tuning
  This issue has a mass of 47.
  We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
  The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the

comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.
Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
        self.weight = self.weight.view(self.oH * self.oW, self.nOutputPlane, self.nInputPlane * self.kH * self.kW)
```

Also found in 1 other location

- torch/legacy/nn/SpatialConvolutionLocal.py:65…66

Similar code found in 1 other location (mass = 47)

Similar code found in 1 other location (mass = 47)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
Tuning
This issue has a mass of 47.
We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass

threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
            self.gradWeight = self.gradWeight.view(
                self.oH * self.oW, self.nOutputPlane,
self.nInputPlane * self.kH * self.kW)
```

Also found in 1 other location

- torch/legacy/nn/SpatialConvolutionLocal.py:63

Identical code found in 2 other locations (mass = 43)

Identical code found in 2 other locations (mass = 43)

Duplicated Code

Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.

When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code

has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).

Tuning

This issue has a mass of 43.

We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.

The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.

If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.

See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki

- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76
- Disable duplication
- Disable this check
- Ignore this issue

```
        if self.padW != 0 or self.padH != 0:
            s += ', {}, {}'.format(self.padW, self.padH)
```

Also found in 2 other locations

- torch/legacy/nn/SpatialConvolution.py:155…156
- torch/legacy/nn/SpatialDilatedConvolution.py:80…81

- **Similar code found in 1 other location (mass = 32)**
  Similar code found in 1 other location (mass = 32)
  Duplicated Code
  Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
  When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
  Tuning
  This issue has a mass of 32.
  We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project

guidelines.
The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

Disable duplication

- Disable this check
- Ignore this issue

```
        self.weight = self.weight.view(self.oH, self.oW, self.nOutputPlane, self.nInputPlane, self.kH, self.kW)
```

Also found in 1 other location

- torch/legacy/nn/SpatialConvolutionLocal.py:71…72

- **Similar code found in 1 other location (mass = 32)**
  Similar code found in 1 other location (mass = 32)
  Duplicated Code
  Duplicated code can lead to software that is hard to understand and difficult to change. The Don't Repeat Yourself (DRY) principle states:Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.
  When you violate DRY, bugs and maintenance problems are sure to follow. Duplicated code has a tendency to both continue to replicate and also to diverge (leaving bugs as two similar implementations differ in subtle ways).
  Tuning
  This issue has a mass of 32.
  We set useful threshold defaults for the languages we support but you may want to adjust these settings based on your project guidelines.
  The threshold configuration represents the minimum mass a code block must have to be analyzed for duplication. The lower the threshold, the more fine-grained the comparison.
  If the engine is too easily reporting duplication, try raising the threshold. If you suspect that the

engine isn't catching enough duplication, try lowering the threshold. The best setting tends to differ from language to language.
See codeclimate-duplication's documentation for more information about tuning the mass threshold in your .codeclimate.yml.

Refactorings

- Extract Method
- Extract Class
- Form Template Method
- Introduce Null Object
- Pull Up Method
- Pull Up Field
- Substitute Algorithm

Further Reading

- Don't Repeat Yourself on the C2 Wiki
- Duplicated Code on SourceMaking
- Refactoring: Improving the Design of Existing Code by Martin Fowler. *Duplicated Code*, p76

- Disable duplication
- Disable this check
- Ignore this issue

- `self.gradWeight = self.gradWeight.view( self.oH, self.oW, self.nOutputPlane, self.nInputPlane, self.kH, self.kW)`
  Also found in 1 other location
- torch/legacy/nn/SpatialConvolutionLocal.py:69

torch/legacy/nn/SpatialConvolutionLocal.py

```
import math
import torch
from .Module import Module
from .utils import clear


class SpatialConvolutionLocal(Module):

    def __init__(self, nInputPlane, nOutputPlane, iW, iH, kW, kH,
dW=1, dH=1, padW=0, padH=None):
```

```
        super(SpatialConvolutionLocal, self).__init__()

        self.nInputPlane = nInputPlane
        self.nOutputPlane = nOutputPlane
        self.kW = kW
        self.kH = kH
        self.iW = iW
        self.iH = iH

        self.dW = dW
        self.dH = dH
        self.padW = padW
        self.padH = padH if padH is not None else padW

        self.oW = int(math.floor((self.padW * 2 + iW - self.kW) /
self.dW)) + 1
        self.oH = int(math.floor((self.padH * 2 + iH - self.kH) /
self.dH)) + 1

        assert 1 <= self.oW and 1 <= self.oH

        self.weight = torch.Tensor(self.oH, self.oW, nOutputPlane,
nInputPlane, kH, kW)
        self.bias = torch.Tensor(nOutputPlane, self.oH, self.oW)
        self.gradWeight = torch.Tensor().resize_as_(self.weight)
        self.gradBias = torch.Tensor().resize_as_(self.bias)

        self.reset()
        self.finput = None
        self.fgradInput = None

    def reset(self, stdv=None):

        if stdv is not None:
            stdv = stdv * math.sqrt(3)
        else:
            stdv = 1. / math.sqrt(self.kW * self.kH *
self.nInputPlane)

        self.weight.uniform_(-stdv, stdv)
        self.bias.uniform_(-stdv, stdv)


    def _makeContiguous(self, input, gradOutput=None):
        if not input.is_contiguous():
            if self._input is None:
                self._input = input.new()
            self._input.resize_as_(input).copy_(input)
```

```
            input = self._input

        if gradOutput is not None:
            if not gradOutput.is_contiguous():
                if self._gradOutput is None:
                    self._gradOutput = gradOutput.new()

self._gradOutput.resize_as_(gradOutput).copy_(gradOutput)
                gradOutput = self._gradOutput
            return input, gradOutput

        return input

    def _viewWeight(self):

        self.weight = self.weight.view(self.oH * self.oW,
self.nOutputPlane, self.nInputPlane * self.kH * self.kW)

        if self.gradWeight is not None and self.gradWeight.dim() > 0:

            self.gradWeight = self.gradWeight.view(
                self.oH * self.oW, self.nOutputPlane,
self.nInputPlane * self.kH * self.kW)

    def _unviewWeight(self):

        self.weight = self.weight.view(self.oH, self.oW,
self.nOutputPlane, self.nInputPlane, self.kH, self.kW)

        if self.gradWeight is not None and self.gradWeight.dim() > 0:

            self.gradWeight = self.gradWeight.view(
                self.oH, self.oW, self.nOutputPlane,
self.nInputPlane, self.kH, self.kW)


    def _checkInputSize(self, input):

        if input.ndimension() == 3:
            if input.size(0) != self.nInputPlane or input.size(1) !=
self.iH or input.size(1) != self.iW:
                raise RuntimeError(
                    'Given input size: ({}x{}x{}) inconsistent with
expected input size: ({}x{}x{}).'.format(
                        input.size(0), input.size(1), input.size(2),
self.nInputPlane, self.iH, self.iW))
        elif input.ndimension() == 4:
            if input.size(1) != self.nInputPlane or input.size(2) !=
```

```
self.iH or input.size(3) != self.iW:
                raise RuntimeError(
                    'Given input size: ({}x{}x{}x{}) inconsistent
with expected input size: (*x{}x{}x{}).'.format(
                        input.size(0), input.size(1), input.size(2),
input.size(3), self.nInputPlane, self.iH, self.iW))
        else:
            raise RuntimeError('3D or 4D (batch mode) tensor
expected')


    def _checkOutputSize(self, input, output):
        if output.ndimension() != input.ndimension():
            raise RuntimeError('inconsistent dimension between output
and input.')


        if output.ndimension() == 3:
            if output.size(0) != self.nOutputPlane or output.size(1)
!= self.oH or output.size(2) != self.oW:
                raise RuntimeError(
                    'Given output size: ({}x{}x{}) inconsistent with
expected output size: ({}x{}x{}).'.format(
                        output.size(0), output.size(1),
output.size(2), self.nOutputPlane, self.oH, self.oW))
        elif output.ndimension() == 4:
            if output.size(1) != self.nOutputPlane or output.size(2)
!= self.oH or output.size(3) != self.oW:
                raise RuntimeError('Given output size: ({}x{}x{}x{})
inconsistent with expected output size: '
                                   '(batchsize x{}x{}x{}).'.format(
                                       output.size(0),
output.size(1), output.size(2),
                                       output.size(3),
self.nOutputPlane, self.oH, self.oW))
        else:
            raise RuntimeError('3D or 4D(batch mode) tensor
expected')

    def updateOutput(self, input):
        if self.finput is None:
            self.finput = input.new()
        if self.fgradInput is None:
            self.fgradInput = input.new()
        self._checkInputSize(input)
        self._viewWeight()
        input = self._makeContiguous(input)
        self._backend.SpatialConvolutionLocal_updateOutput(
```

```
            self._backend.library_state,
            input,
            self.output,
            self.weight,
            self.bias,
            self.finput,
            self.fgradInput,
            self.kW, self.kH,
            self.dW, self.dH,
            self.padW, self.padH,
            self.iW, self.iH,
            self.oW, self.oH
        )
        self._unviewWeight()
        return self.output

    def updateGradInput(self, input, gradOutput):
        if self.gradInput is None:
            return

        self._checkInputSize(input)
        self._checkOutputSize(input, gradOutput)

        self._viewWeight()
        input, gradOutput = self._makeContiguous(input, gradOutput)
        self._backend.SpatialConvolutionLocal_updateGradInput(
            self._backend.library_state,
            input,
            gradOutput,
            self.gradInput,
            self.weight,
            self.finput,
            self.fgradInput,
            self.kW, self.kH,
            self.dW, self.dH,
            self.padW, self.padH,
            self.iW, self.iH,
            self.oW, self.oH
        )
        self._unviewWeight()
        return self.gradInput

    def accGradParameters(self, input, gradOutput, scale=1):
        self._checkInputSize(input)
        self._checkOutputSize(input, gradOutput)
        input, gradOutput = self._makeContiguous(input, gradOutput)
        self._viewWeight()
        self._backend.SpatialConvolutionLocal_accGradParameters(
```

```
            self._backend.library_state,
            input,
            gradOutput,
            self.gradWeight,
            self.gradBias,
            self.finput,
            self.fgradInput,
            self.kW, self.kH,
            self.dW, self.dH,
            self.padW, self.padH,
            self.iW, self.iH,
            self.oW, self.oH,
            scale
        )
        self._unviewWeight()


    def type(self, type=None, tensorCache=None):
        if self.finput is not None:
            self.finput = torch.Tensor()
        if self.fgradInput is not None:
            self.fgradInput = torch.Tensor()
        return super(SpatialConvolutionLocal, self).type(type,
tensorCache)


    def __tostring__(self, ):
        s = super(SpatialConvolution, self).__repr__()
        s += '({} -> {}, {}x{}, {}x{}'.format(self.nInputPlane,
self.nOutputPlane, self.iW, self.iH, self.kW, self.kH)

        if self.dW != 1 or self.dH != 1 or self.padW != 0 or
self.padH != 0:
            s += ', {}, {}'.format(self.dW, self.dH)


        if self.padW != 0 or self.padH != 0:
            s += ', {}, {}'.format(self.padW, self.padH)

        s += ')'
        return s

    def clearState(self):
        clear(self, 'finput', 'fgradInput', '_input', '_gradOutput')
        return super(SpatialConvolutionLocal, self).clearState()
```

Help

Docs

- We're Hiring!
- About
- Blog
- Legal
- Security
- Status
- @codeclimate