

## tf.layers.AveragePooling3D

### Contents

Class `AveragePooling3D`

### Properties

`activity_regularizer`

`dtype`

## Class `AveragePooling3D`

Defined in `tensorflow/python/layers/pooling.py`.

Average pooling layer for 3D inputs (e.g. volumes).

### Arguments:

- `pool_size`: An integer or tuple/list of 3 integers: (`pool_depth`, `pool_height`, `pool_width`) specifying the size of the pooling window. Can be a single integer to specify the same value for all spatial dimensions.
- `strides`: An integer or tuple/list of 3 integers, specifying the strides of the pooling operation. Can be a single integer to specify the same value for all spatial dimensions.
- `padding`: A string. The padding method, either 'valid' or 'same'. Case-insensitive.
- `data_format`: A string. The ordering of the dimensions in the inputs. `channels_last` (default) and `channels_first` are supported. `channels_last` corresponds to inputs with shape `(batch, depth, height, width, channels)` while `channels_first` corresponds to inputs with shape `(batch, channels, depth, height, width)`.
- `name`: A string, the name of the layer.

## Properties

### `activity_regularizer`

Optional regularizer function for the output of this layer.

### `dtype`

### `graph`

### `input`

Retrieves the input tensor(s) of a layer.

Only applicable if the layer has exactly one input, i.e. if it is connected to one incoming layer.

### Returns:

Input tensor or list of input tensors.

Raises:

- `AttributeError` : if the layer is connected to more than one incoming layers.

Raises:

- `RuntimeError` : If called in Eager mode.
- `AttributeError` : If no inbound nodes are found.

## **input\_shape**

Retrieves the input shape(s) of a layer.

Only applicable if the layer has exactly one input, i.e. if it is connected to one incoming layer, or if all inputs have the same shape.

Returns:

Input shape, as an integer shape tuple (or list of shape tuples, one tuple per input tensor).

Raises:

- `AttributeError` : if the layer has no defined input\_shape.
- `RuntimeError` : if called in Eager mode.

## **losses**

**name**

**non\_trainable\_variables**

**non\_trainable\_weights**

**output**

Retrieves the output tensor(s) of a layer.

Only applicable if the layer has exactly one output, i.e. if it is connected to one incoming layer.

Returns:

Output tensor or list of output tensors.

Raises:

- `AttributeError` : if the layer is connected to more than one incoming layers.
- `RuntimeError` : if called in Eager mode.

**output\_shape**

Retrieves the output shape(s) of a layer.

Only applicable if the layer has one output, or if all outputs have the same shape.

Returns:

Output shape, as an integer shape tuple (or list of shape tuples, one tuple per output tensor).

Raises:

- `AttributeError` : if the layer has no defined output shape.
- `RuntimeError` : if called in Eager mode.

**scope\_name**

**trainable\_variables**

**trainable\_weights**

**updates**

**variables**

Returns the list of all layer variables/weights.

Returns:

A list of variables.

**weights**

Returns the list of all layer variables/weights.

Returns:

A list of variables.

## Methods

---

**`--init--`**

```
--init__(
    pool_size,
    strides,
    padding='valid',
    data_format='channels_last',
    name=None,
    **kwargs
)
```

## `__call__`

```
__call__(
    inputs,
    *args,
    **kwargs
)
```

Wraps `call`, applying pre- and post-processing steps.

### Arguments:

- `inputs`: input tensor(s).
- `*args`: additional positional arguments to be passed to `self.call`.
- `**kwargs`: additional keyword arguments to be passed to `self.call`. **Note:** kwarg `scope` is reserved for use by the layer.

### Returns:

Output tensor(s).

★ **Note:** - If the layer's `call` method takes a `scope` keyword argument, this argument will be automatically set to the current variable scope. - If the layer's `call` method takes a `mask` argument (as some Keras layers do), its default value will be set to the mask generated for `inputs` by the previous layer (if `input` did come from a layer that generated a corresponding mask, i.e. if it came from a Keras layer with masking support).

### Raises:

- `ValueError`: if the layer's `call` method returns `None` (an invalid value).

## `__deepcopy__`

```
__deepcopy__(memo)
```

## `add_loss`

```
add_loss(
    losses,
    inputs=None
)
```

Add loss tensor(s), potentially dependent on layer inputs.

Some losses (for instance, activity regularization losses) may be dependent on the inputs passed when calling a layer. Hence, when reusing a same layer on different inputs `a` and `b`, some entries in `layer.losses` may be dependent on `a` and some on `b`. This method automatically keeps track of dependencies.

The `get_losses_for` method allows to retrieve the losses relevant to a specific set of inputs.

### Arguments:

- `losses`: Loss tensor, or list/tuple of tensors.

- `inputs` : Optional input tensor(s) that the loss(es) depend on. Must match the `inputs` argument passed to the `__call__` method at the time the losses are created. If `None` is passed, the losses are assumed to be unconditional, and will apply across all dataflows of the layer (e.g. weight regularization losses).

Raises:

- `RuntimeError` : If called in Eager mode.

## add\_update

```
add_update(
    updates,
    inputs=None
)
```

Add update op(s), potentially dependent on layer inputs.

Weight updates (for instance, the updates of the moving mean and variance in a BatchNormalization layer) may be dependent on the inputs passed when calling a layer. Hence, when reusing a same layer on different inputs `a` and `b`, some entries in `layer.updates` may be dependent on `a` and some on `b`. This method automatically keeps track of dependencies.

The `get_updates_for` method allows to retrieve the updates relevant to a specific set of inputs.

This call is ignored in Eager mode.

Arguments:

- `updates` : Update op, or list/tuple of update ops.
- `inputs` : Optional input tensor(s) that the update(s) depend on. Must match the `inputs` argument passed to the `__call__` method at the time the updates are created. If `None` is passed, the updates are assumed to be unconditional, and will apply across all dataflows of the layer.

## add\_variable

```
add_variable(
    name,
    shape,
    dtype=None,
    initializer=None,
    regularizer=None,
    trainable=True,
    constraint=None
)
```

Adds a new variable to the layer, or gets an existing one; returns it.

Arguments:

- `name` : variable name.
- `shape` : variable shape.
- `dtype` : The type of the variable. Defaults to `self.dtype` or `float32`.
- `initializer` : initializer instance (callable).

- `regularizer` : regularizer instance (callable).
- `trainable` : whether the variable should be part of the layer's "trainable\_variables" (e.g. variables, biases) or "non\_trainable\_variables" (e.g. BatchNorm mean, stddev).
- `constraint` : constraint instance (callable).

Returns:

The created variable.

Raises:

- `RuntimeError` : If called in Eager mode with regularizers.

## apply

```
apply(
    inputs,
    *args,
    **kwargs
)
```

Apply the layer on a input.

This simply wraps `self.__call__`.

Arguments:

- `inputs` : Input tensor(s).
- `*args` : additional positional arguments to be passed to `self.call`.
- `**kwargs` : additional keyword arguments to be passed to `self.call`.

Returns:

Output tensor(s).

## build

```
build(_)
```

Creates the variables of the layer.

## call

```
call(inputs)
```

## count\_params

```
count_params()
```

Count the total number of scalars composing the weights.

Returns:

An integer count.

Raises:

- `ValueError` : if the layer isn't yet built (in which case its weights aren't yet defined).

## **get\_input\_at**

```
get_input_at(node_index)
```

Retrieves the input tensor(s) of a layer at a given node.

Arguments:

- `node_index` : Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

Returns:

A tensor (or list of tensors if the layer has multiple inputs).

Raises:

- `RuntimeError` : If called in Eager mode.

## **get\_input\_shape\_at**

```
get_input_shape_at(node_index)
```

Retrieves the input shape(s) of a layer at a given node.

Arguments:

- `node_index` : Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

Returns:

A shape tuple (or list of shape tuples if the layer has multiple inputs).

Raises:

- `RuntimeError` : If called in Eager mode.

## **get\_losses\_for**

```
get_losses_for(inputs)
```

Retrieves losses relevant to a specific set of inputs.

#### Arguments:

- `inputs`: Input tensor or list/tuple of input tensors. Must match the `inputs` argument passed to the `__call__` method at the time the losses were created. If you pass `inputs=None`, unconditional losses are returned, such as weight regularization losses.

#### Returns:

List of loss tensors of the layer that depend on `inputs`.

#### Raises:

- `RuntimeError`: If called in Eager mode.

### **get\_output\_at**

```
get_output_at(node_index)
```

Retrieves the output tensor(s) of a layer at a given node.

#### Arguments:

- `node_index`: Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

#### Returns:

A tensor (or list of tensors if the layer has multiple outputs).

#### Raises:

- `RuntimeError`: If called in Eager mode.

### **get\_output\_shape\_at**

```
get_output_shape_at(node_index)
```

Retrieves the output shape(s) of a layer at a given node.

#### Arguments:

- `node_index`: Integer, index of the node from which to retrieve the attribute. E.g. `node_index=0` will correspond to the first time the layer was called.

#### Returns:

A shape tuple (or list of shape tuples if the layer has multiple outputs).



Raises:

- `RuntimeError` : If called in Eager mode.

## get\_updates\_for

```
get_updates_for(inputs)
```

Retrieves updates relevant to a specific set of inputs.

Arguments:

- `inputs` : Input tensor or list/tuple of input tensors. Must match the `inputs` argument passed to the `__call__` method at the time the updates were created. If you pass `inputs=None`, unconditional updates are returned.

Returns:

List of update ops of the layer that depend on `inputs`.

Raises:

- `RuntimeError` : If called in Eager mode.

---

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.

### Stay Connected

[Blog](#)

[GitHub](#)

[Twitter](#)

### Support

[Issue Tracker](#)

[Release Notes](#)

[Stack Overflow](#)

English

[Terms](#) | [Privacy](#)