BeyondML

BeyondML Labs

DOCUMENTATION:

	Installation 1.1 beyondml	3			
	Changelog	47			
Ру	ython Module Index				
In	dex	51			



BeyondML is a Python package which enables creating sparse multitask artificial neural networks (MANNs) compatible with TensorFlow and PyTorch. This package contains custom layers and utilities to facilitate the training and optimization of models using the Reduction of Sub-Network Neuroplasticity (RSN2) training procedure developed by AI Squared, Inc.

View this Documentation in PDF Format

DOCUMENTATION: 1

2 DOCUMENTATION:

CHAPTER

ONE

INSTALLATION

This package is available through Pypi and can be installed by running the following command:

```
pip install beyondml
```

Alternatively, the latest version of the software can be installed directly from GitHub using the following command:

```
pip install git+https://github.com/beyond-ml-labs/beyondml
```

1.1 beyondml

1.1.1 beyondml package

Subpackages

beyondml.pt package

Subpackages

beyondml.pt.layers package

Submodules

beyondml.pt.layers.Conv2D module

Bases: Module

Convolutional 2D layer initialized directly with weights, rather than with hyperparameters

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

torch.Tensor

beyondml.pt.layers.Conv3D module

```
class beyondml.pt.layers.Conv3D.Conv3D(kernel, bias, padding='same', strides=1, device=None,
                                               dtype=None)
     Bases: Module
     Convolutional 3D layer initialized directly with weights, rather than with hyperparameters
     forward(inputs)
           Call the layer on input data
```

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.Dense module

```
class beyondml.pt.layers.Dense(weight, bias, device=None, dtype=None)
     Bases: Module
     Fully-connected layer initialized directly with weights, rather than hyperparameters
     forward(inputs)
          Call the layer on input data
              Parameters
```

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.FilterLayer module

```
class beyondml.pt.layers.FilterLayer.FilterLayer(is_on=True, device=None, dtype=None)
     Bases: Module
     Layer which filters input data, either returning values or all zeros depending on state
     forward(inputs)
           Call the layer on input data
               Parameters
                   inputs (torch.Tensor) - Inputs to call the layer's logic on
               Returns
                   results – The results of the layer's logic
```

torch.Tensor

property is_on

turn_off()

Turn off the layer

turn_on()

Turn on the layer

beyondml.pt.layers.MaskedConv2D module

Bases: Module

Masked 2D Convolutional layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

property in_channels

property kernel_size

property out_channels

prune(percentile)

Prune the layer by updating the layer's mask

Parameters

percentile (*int*) – Integer between 0 and 99 which represents the proportion of weights to be inactive

Notes

Acts on the layer in place

beyondml.pt.layers.MaskedConv3D module

```
class beyondml.pt.layers.MaskedConv3D.MaskedConv3D(in_channels, out_channels, kernel_size=3,
                                                             padding='same', strides=1, device=None,
                                                             dtype=None)
     Bases: Module
     Masked 3D Convolutional layer
     forward(inputs)
          Call the layer on input data
               Parameters
                   inputs (torch. Tensor) – Inputs to call the layer's logic on
               Returns
                   results – The results of the layer's logic
               Return type
                  torch.Tensor
     property in_channels
     property kernel_size
     property out_channels
     prune(percentile)
          Prune the layer by updating the layer's masks
                  percentile (int) – Integer between 0 and 99 which represents the proportion of weights to
                  be inactive
          Notes
          Acts on the layer in place
beyondml.pt.layers.MaskedDense module
class beyondml.pt.layers.MaskedDense.MaskedDense(in_features, out_features, device=None,
                                                           dtype=None)
     Bases: Module
     Masked fully-connected layer
     forward(inputs)
          Call the layer on input data
               Parameters
                   inputs (torch. Tensor) - Inputs to call the layer's logic on
               Returns
                   results - The results of the layer's logic
               Return type
                  torch.Tensor
```

prune(percentile)

Prune the layer by updating the layer's mask

Parameters

percentile (*int*) – Integer between 0 and 99 which represents the proportion of weights to be inactive

Notes

Acts on the layer in place

beyondml.pt.layers.MaskedMultiHeadAttention module

 $\textbf{class} \ \, \textbf{beyondml.pt.layers.MaskedMultiHeadAttention.MaskedMultiHeadAttention} (embed_dim, \\ num_heads, \\ dropout=0, \\ batch_first=False, \\ \end{cases}$

device=None,
dtype=None)

Bases: Module

Masked Multi-Headed Attention Layer

Call the layer on input data

Parameters

- query (torch Tensor) Query tensor
- key (torch Tensor) Key tensor
- value (torch Tensor) Value tensor
- **key_padding_mask** (*None or torch Tensor (default None)*) If specified, a mask indicating which elements in **key** to ignore
- need_weights(Bool (default True))—If specified, returns attn_output_weights as well as attn_outputs
- attn_mask (None or torch Tensor (default None)) If specified, a 2D or 3D mask preventing attention
- average_attn_weights (Bool (default True)) If True, indicates that returned attn_weights should be averaged across heads

prune(percentile)

Prune the layer by updating the layer's mask

Parameters

percentile (*int*) – Integer between 0 and 99 which represents the proportion of weights to be made inactive

Notes

Acts on the layer in place

beyondml.pt.layers.MaskedTransformerDecoderLayer module

 $\textbf{class} \ \ beyond \texttt{ml.pt.layers.MaskedTransformerDecoderLayer.MaskedTransformerDecoderLayer} (\textit{d_model:} \\$

```
int,
nhead:
int,
dim feedforward:
int =
2048,
dropout:
float
0.1.
acti-
va-
tion:
str |
~typ-
ing.Callable[[~torch.Te
~torch.Tensor]
<func-
tion
relu>,
layer_norm_eps:
float
= 1e-
05,
batch_first:
bool
False,
norm_first:
bool
False,
de-
vice=None,
dtype=None)
```

Bases: Module

TransformerDecoderLayer is made up of self-attn, multi-head-attn and feedforward network. This standard decoder layer is based on the paper "Attention Is All You Need". :param d_model: the number of expected features in the input (required). :param nhead: the number of heads in the multiheadattention models (required). :param dim_feedforward: the dimension of the feedforward network model (default=2048). :param dropout: the dropout value (default=0.1). :param activation: the activation function of the intermediate layer, can be a string

("relu" or "gelu") or a unary callable. Default: relu

Parameters

- layer_norm_eps the eps value in layer normalization components (default=1e-5).
- **batch_first** If True, then the input and output tensors are provided as (batch, seq, feature). Default: False (seq, batch, feature).
- **norm_first** if True, layer norm is done prior to self attention, multihead attention and feedforward operations, respectively. Otherwise it's done after. Default: False (after).

forward(tgt: Tensor, memory: Tensor)

Pass the inputs (and mask) through the decoder layer. :param tgt: the sequence to the decoder layer. :param memory: the sequence from the last layer of the encoder.

Shape

see the docs in Pytorch Transformer class.

prune(percentile)

beyondml.pt.layers.MaskedTransformerEncoderLayer module

class beyondml.pt.layers.MaskedTransformerEncoderLayer.MaskedTransformerEncoderLayer(d_model:

```
int.
nhead:
int.
dim_feedforward:
int =
2048.
dropout:
float
0.1,
acti-
va-
tion:
str |
~typ-
ing.Callable[[~torch.Te
~torch.Tensor]
<func-
tion
relu>,
layer_norm_eps:
float
= 1e-
05.
batch_first:
bool
False,
norm_first:
bool
False,
de-
vice=None,
dtype=None)
```

Bases: Module

TransformerEncoderLayer is made up of self-attn and feedforward network. :param d_model: the number of expected features in the input (required). :param nhead: the number of heads in the multiheadattention models (required). :param dim_feedforward: the dimension of the feedforward network model (default=2048). :param dropout: the dropout value (default=0.1). :param activation: the activation function of the intermediate layer, can be a string

("relu" or "gelu") or a unary callable. Default: relu

Parameters

- layer_norm_eps the eps value in layer normalization components (default=1e-5).
- **batch_first** If True, then the input and output tensors are provided as (batch, seq, feature). Default: False (seq, batch, feature).
- **norm_first** if True, layer norm is done prior to attention and feedforward operations, respectively. Otherwise it's done after. Default: False (after).

```
forward(src: Tensor)
          Pass the input through the encoder layer. :param src: the sequence to the encoder layer (required).
     prune(percentile)
beyondml.pt.layers.MultiConv2D module
class beyondml.pt.layers.MultiConv2D.MultiConv2D(kernel, bias, padding='same', strides=1,
                                                          device=None, dtype=None)
     Bases: Module
     Multi- 2D Convolutional layer initialized with weights rather than hyperparameters
     forward(inputs)
          Call the layer on input data
              Parameters
                  inputs (torch. Tensor) – Inputs to call the layer's logic on
              Returns
                  results – The results of the layer's logic
              Return type
                  torch.Tensor
beyondml.pt.layers.MultiConv3D module
class beyondml.pt.layers.MultiConv3D.MultiConv3D(kernel, bias, padding='same', strides=1,
                                                          device=None, dtype=None)
     Bases: Module
     Multitask 3D Convolutional layer initialized with weights rather than with hyperparameters
     forward(inputs)
          Call the layer on input data
              Parameters
                  inputs (torch. Tensor) – Inputs to call the layer's logic on
              Returns
                  results – The results of the layer's logic
              Return type
                  torch.Tensor
beyondml.pt.layers.MultiDense module
class beyondml.pt.layers.MultiDense(weight, bias, device=None, dtype=None)
     Bases: Module
     Multi-Fully-Connected layer initialized with weights rather than hyperparameters
```

```
forward(inputs)
          Call the layer on input data
              Parameters
                  inputs (torch. Tensor) – Inputs to call the layer's logic on
              Returns
                  results – The results of the layer's logic
              Return type
                  torch.Tensor
beyondml.pt.layers.MultiMaskedConv2D module
class beyondml.pt.layers.MultiMaskedConv2D.MultiMaskedConv2D(in_channels, out_channels,
                                                                        num_tasks, kernel_size=3,
                                                                        padding='same', strides=1,
                                                                        device=None, dtype=None)
     Bases: Module
     Multi 2D Convolutional layer which supports masking and pruning
     forward(inputs)
```

Parameters

Call the layer on input data

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

```
property in_channels
```

property kernel_size

property out_channels

prune(percentile)

Prune the layer by updating the layer's mask

Parameters

percentile (*int*) – Integer between 0 and 99 which represents the proportion of weights to be inactive

Notes

Acts on the layer in place

beyondml.pt.layers.MultiMaskedConv3D module

```
class beyondml.pt.layers.MultiMaskedConv3D.MultiMaskedConv3D(in_channels, out_channels,
                                                                         num_tasks, kernel_size=3,
                                                                         padding='same', strides=1,
                                                                         device=None, dtype=None)
     Bases: Module
     Masked Multitask 3D Convolutional layer
     forward(inputs)
          Call the layer on input data
              Parameters
                  inputs (torch.Tensor) - Inputs to call the layer's logic on
                  results – The results of the layer's logic
              Return type
                  torch.Tensor
     property in_channels
     property kernel_size
     property out_channels
     prune(percentile)
          Prune the layer by updating the layer's masks
              Parameters
                  percentile (int) – Integer between 0 and 99 which represents the proportion of weights to
                  be inactive
          Notes
          Acts on the layer in place
beyondml.pt.layers.MultiMaskedDense module
class beyondml.pt.layers.MultiMaskedDense.MultiMaskedDense(in_features, out_features, num_tasks,
                                                                      device=None, dtype=None)
     Bases: Module
     Multi-Fully-Connected layer which supports masking and pruning
     forward(inputs)
          Call the layer on input data
              Parameters
                  inputs (torch. Tensor) – Inputs to call the layer's logic on
              Returns
                  results – The results of the layer's logic
              Return type
```

1.1. beyondml

torch.Tensor

```
prune(percentile)
```

Prune the layer by updating the layer's mask

Parameters

percentile (*int*) – Integer between 0 and 99 which represents the proportion of weights to be inactive

Notes

Acts on the layer in place

beyondml.pt.layers.MultiMaxPool2D module

Bases: Module

Multitask implementation of 2-dimensional Max Pooling layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch. Tensor) – Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.MultiMaxPool3D module

Bases: Module

Multitask implementation of 2-dimensional Max Pooling layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.MultitaskNormalization module

```
class beyondml.pt.layers.MultitaskNormalization.MultitaskNormalization(device=None,
                                                                                  dtvpe=None)
     Bases: Module
     Layer which normalizes a set of inputs to sum to 1
     forward(inputs)
          Call the layer on input data
              Parameters
                  inputs (torch. Tensor or list of Tensors) – Inputs to call the layer's logic on
              Returns
```

torch. Tensor or list of Tensors

results – The results of the layer's logic

beyondml.pt.layers.SelectorLayer module

Return type

```
class beyondml.pt.layers.SelectorLayer.SelectorLayer(sel_index)
     Bases: Module
     Layer which selects an individual input based on index and only returns that one
     forward(inputs)
           Call the layer on input data
               Parameters
                   inputs (torch. Tensor) – Inputs to call the layer's logic on
               Returns
                   results – The results of the layer's logic
               Return type
                   torch.Tensor
     property sel_index
```

beyondml.pt.layers.SparseConv2D module

```
class beyondml.pt.layers.SparseConv2D.SparseConv2D(kernel, bias, padding='same', strides=1,
                                                               device=None, dtype=None)
     Bases: Module
     Sparse implementation of a 2D Convolutional layer, expected to be converted from a trained, pruned layer
     forward(inputs)
           Call the layer on input data
               Parameters
                   inputs (torch. Tensor) - Inputs to call the layer's logic on
               Returns
                   results – The results of the layer's logic
```

torch.Tensor

beyondml.pt.layers.SparseConv3D module

```
{\bf class} \ \ beyond {\tt ml.pt.layers.SparseConv3D.SparseConv3D} ({\it kernel, bias, padding='same', strides=1, device=None, dtype=None}) \\
```

Bases: Module

Sparse 3D Convolutional layer, expected to be converted from a trained, pruned layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results - The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.SparseDense module

```
class beyondml.pt.layers.SparseDense.SparseDense(weight, bias, device=None, dtype=None)
```

Bases: Module

Sparse implementation of a fully-connected layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch. Tensor) – Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.SparseMultiConv2D module

Bases: Module

Sparse implementation of a Multi 2D Convolutional layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

```
Returns
```

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.SparseMultiConv3D module

Bases: Module

Sparse implementation of a Multitask 3D Convolutional layer, expected to be converted from a trained, pruned layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

beyondml.pt.layers.SparseMultiDense module

Bases: Module

Sparse implementation of the Multi-Fully-Connected layer

forward(inputs)

Call the layer on input data

Parameters

inputs (torch.Tensor) - Inputs to call the layer's logic on

Returns

results – The results of the layer's logic

Return type

torch.Tensor

Module contents

Layers compatible with PyTorch models

beyondml.pt.utils package

Submodules

beyondml.pt.utils.utils module

beyondml.pt.utils.utils.prune_model(model, percentile)

Prune a compatible model

Parameters

- **model** (*PyTorch model*) A model that has been developed to have a .*layers* property containing layers to be pruned
- **percentile** (*int*) An integer between 0 and 99 which corresponds to how much to prune the model

Returns

pruned_model - The pruned model

Return type

PyTorch model

Notes

- The model input **must** have a *.layers* property to be able to function. Only layers within the *.layers* property which are recognized as prunable are pruned, via their own *.prune()* method
- Also acts on the model in place, but returns the model for ease of use

Module contents

Some additional utilities for building MANN models in PyTorch.

Module contents

PyTorch compatibility for building MANN models

The *beyondml.pt* subpackage contains layers and utilities for creating and pruning models using [PyTorch](https://pytorch.org). The package contains two subpackages, the *beyondml.pt.layers* package, and the *beyondml.pt.utils* package.

Within the *layers* package, there is current functionality for the the following layers: - *beyondml.pt.layers.Conv2D* - *beyondml.pt.layers.Dense* - *beyondml.pt.layers.FilterLayer* - *beyondml.pt.layers.MaskedConv2D* - *beyondml.pt.layers.MultiDense* - *beyondml.pt.layers.MultiDense* - *beyondml.pt.layers.MultiMaskedDense* - *beyondml.pt.layers.SparseDense* - *beyondml.pt.layers.SparseMultiConv2D* - *beyondml.pt.layers.SparseDense* - *beyondml.pt.layers.SparseMultiConv2D* - *beyondml.pt.layers.SparseMultiDense*

Within the *beyondml.pt.utils* package, there is currently only one function, the *prune_model* function. Because of the openness of developing with PyTorch in comparison to TensorFlow, there is far less functionality that can be supplied directly via BeyondML. Instead, for converting models from training to inference, the user is left to devise the best way to do so by building his or her own classes.

Best Practices for Pruning In order to use the *utils.prune_model* function, the model itself must have a *.layers* property. This property is used to determine which layers can be pruned. **Only layers which support pruning and which are included in the `.layers` property are pruned,** meaning the user can determine which exact layers in the model he or she wants pruned. Alternatively, the user can create their own pruning function or method on the class itself and prune that way, utilizing each of the *.prune()* methods of the layers provided.

beyondml.tflow package

Subpackages

beyondml.tflow.layers package

Submodules

beyondml.tflow.layers.FilterLayer module

```
\textbf{class} \ \ beyond \texttt{ml.tflow.layers.FilterLayer.FilterLayer} (*args, **kwargs)
```

Bases: Layer

Layer which filters inputs based on status of on or off

Example:

```
>>> # Create a model with just a FilterLayer
>>> input_layer = tf.keras.layers.Input(10)
>>> filter_layer = mann.layers.FilterLayer()(input_layer)
>>> model = tf.keras.models.Model(input_layer, filter_layer)
>>> model.compile()
>>> # Call the model with the layer turned on
>>> data = np.arange(10).reshape((1, 10))
>>> model.predict(data)
array([[0., 1., 2., 3., 4., 5., 6., 7., 8., 9.]], dtype=float32)
>>> # Turn off the FilterLayer and call it again
>>> model.layers[-1].turn_off()
>>> # Model must be recompiled after turning the layer on or off
>>> model.compile()
>>> model.predict(data)
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

```
inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer
```

Returns

outputs - The outputs of the layer's logic

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

turn_off()

Turn the layer off so inputs are destroyed and all-zero tensors are output

turn_on()

Turn the layer on so inputs are returned unchanged as outputs

beyondml.tflow.layers.MaskedConv2D module

```
class beyondml.tflow.layers.MaskedConv2D.MaskedConv2D(*args, **kwargs)
```

Bases: Layer

Masked 2-dimensional convolutional layer. For full documentation of the convolutional architecture, see the TensorFlow Keras Convolutional2D layer documentation.

This layer implements masking consistent with the BeyondML API to support developing sparse models.

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (*TensorFlow Tensor or Tensor-like*) – The inputs to the layer

Returns

outputs – The outputs of the layer's logic

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property kernel_size

```
set_masks(new_masks)
```

Set the masks for the layer

Parameters

new_masks (list of arrays or array-likes) - The new masks to set for the layer

beyondml.tflow.layers.MaskedConv3D module

```
class beyondml.tflow.layers.MaskedConv3D.MaskedConv3D(*args, **kwargs)
```

Bases: Layer

Masked 3-dimensional convolutional layer. For full documentation of the convolutional architecture, see the TensorFlow Keras Convolutional3D layer documentation.

This layer implements masking consistent with the BeyondML API to support developing sparse models

```
build(input_shape)
```

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs – The outputs of the layer's logic

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property kernel_size

```
set_masks(new_masks)
```

Set the masks for the layer

Parameters

new_masks (list of arrays or array-likes) - The new masks to set for the layer

beyondml.tflow.layers.MaskedDense module

```
class beyondml.tflow.layers.MaskedDense.MaskedDense(*args, **kwargs)
```

Bases: Layer

Masked fully connected layer. For full documentation of the fully-connected architecture, see the TensorFlow Keras Dense layer documentation.

This layer implements masking consistent with the BeyondML API to support developing sparse models.

```
build(input_shape)
```

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

set_masks(new_masks)

Set the masks for the layer

Parameters

new_masks (list of arrays or array-likes) - The new masks to set for the layer

beyondml.tflow.layers.MultiConv2D module

```
class beyondml.tflow.layers.MultiConv2D.MultiConv2D(*args, **kwargs)
```

Bases: Layer

Multitask 2-dimensional convolutional layer

This layer implements multiple stacks of convolutional weights to account for different ways individual neurons activate for various tasks. It is expected that to train using the RSN2 algorithm that MultiMaskedConv2D layers be used during training and then those layers be converted to this layer type.

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property kernel_size

beyondml.tflow.layers.MultiConv3D module

class beyondml.tflow.layers.MultiConv3D.MultiConv3D(*args, **kwargs)

Bases: Layer

Multitask 3-dimensional convolutional layer

This layer implements multiple stacks of convolutional weights to account for different ways individual neurons activate for various tasks. It is expected that to train using the RSN2 algorithm that MultiMaskedConv3D layers be used during training and then those layers be converted to this layer type.

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property kernel_size

beyondml.tflow.layers.MultiDense module

class beyondml.tflow.layers.MultiDense.MultiDense(*args, **kwargs)

Bases: Layer

Multitask fully connected layer

This layer implements multiple stacks of fully connected weights to account for different ways neurons can activate for various tasks. It is expected that to train using the RSN2 algorithm that MultiMaskedDense layers be used during training and then those layers be converted to this layer type.

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.MultiMaskedConv2D module

class beyondml.tflow.layers.MultiMaskedConv2D.MultiMaskedConv2D(*args, **kwargs)

Bases: Layer

Masked multitask 2-dimensional convolutional layer. This layer implements multiple stacks of the convolutional architecture and implements masking consistent with the BeyondML API to support developing sparse multitask models.

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property kernel_size

set_masks(new_masks)

beyondml.tflow.layers.MultiMaskedConv3D module

```
class beyondml.tflow.layers.MultiMaskedConv3D.MultiMaskedConv3D(*args, **kwargs)
```

Bases: Layer

Masked multitask 3-dimensional convoluational layer. This layer implements multiple stacks of the convolutional architecture and implements masking consistent with the BeyondML API to support developing sparse multitask models.

```
build(input_shape)
```

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

```
inputs (TensorFlow Tensor or Tensor-like) – The inputs to the layer
```

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that $get_config()$ does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property kernel_size

set_masks(new_masks)

beyondml.tflow.layers.MultiMaskedDense module

```
class beyondml.tflow.layers.MultiMaskedDense.MultiMaskedDense(*args, **kwargs)
```

Bases: Layer

Masked multitask fully connected layer. This layer implements multiple stacks of the fully-connected architecture and implements masking with the BeyondML API to support developing sparse multitask models.

```
build(input_shape)
```

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that $get_config()$ does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

set_masks(new masks)

Set the masks for the layer

Parameters

new_masks (list of arrays or array-likes) - The new masks to set for the layer

beyondml.tflow.layers.MultiMaxPool2D module

```
class beyondml.tflow.layers.MultiMaxPool2D.MultiMaxPool2D(*args, **kwargs)
```

Bases: Layer

Multitask Max Pooling Layer. This layer implements the Max Pooling algorithm across multiple inputs for developing multitask models

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) – The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that $get_config()$ does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.MultiMaxPool3D module

class beyondml.tflow.layers.MultiMaxPool3D.MultiMaxPool3D(*args, **kwargs)

Bases: Layer

Multitask 3D Max Pooling Layer. This layer implements the Max Pooling algorithm across multiple inputs for developing multitask models

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.MultitaskNormalization module

class beyondml.tflow.layers.MultitaskNormalization.MultitaskNormalization(*args, **kwargs)

Bases: Layer

Multitask layer which normalizes all inputs to sum to 1

build(input_shape)

Creates the variables of the layer (for subclass implementers).

This is a method that implementers of subclasses of *Layer* or *Model* can override if they need a state-creation step in-between layer instantiation and layer call. It is invoked automatically before the first execution of *call()*.

This is typically used to create the weights of *Layer* subclasses (at the discretion of the subclass implementer).

Parameters

input_shape – Instance of *TensorShape*, or list of instances of *TensorShape* if the layer expects a list of inputs (one instance per input).

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SelectorLayer module

```
class beyondml.tflow.layers.SelectorLayer.SelectorLayer(*args, **kwargs)
```

Bases: Layer

Layer which selects individual inputs

Example:

(continues on next page)

(continued from previous page)

```
to select the second input and return it
>>> model = tf.keras.models.Model([input_1, input_2], selector)
>>> model.compile()
>>> # Call the model
>>> data1 = np.arange(10).reshape((1, 10))
>>> data2 = 2*np.arange(10).reshape((1, 10))
>>> model.predict([data1, data2])
array([[ 0., 2., 4., 6., 8., 10., 12., 14., 16., 18.]], dtype=float32)
```

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) – The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

property sel_index

beyondml.tflow.layers.SparseConv2D module

class beyondml.tflow.layers.SparseConv2D.SparseConv2D(*args, **kwargs)

Bases: Layer

Sparse implementation of the Convolutional layer. If used in a model, must be saved and loaded via pickle

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

classmethod from_layer(layer)

Create a layer from an instance of another layer

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SparseConv3D module

class beyondml.tflow.layers.SparseConv3D.SparseConv3D(*args, **kwargs)

Bases: Layer

Sparse implementation of the Convolutional layer. If used in a model, must be saved and loaded via pickle

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (*TensorFlow Tensor or Tensor-like*) – The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

classmethod from_layer(layer)

Create a layer from an instance of another layer

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SparseDense module

class beyondml.tflow.layers.SparseDense.SparseDense(*args, **kwargs)

Bases: Layer

Sparse implementation of the Dense layer. If used in a model, must be saved and loaded via pickle

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) – The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

classmethod from_layer(layer)

Create a layer from an instance of another layer

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SparseMultiConv2D module

class beyondml.tflow.layers.SparseMultiConv2D.SparseMultiConv2D(*args, **kwargs)

Bases: Layer

Sparse implementation of the MultiConv layer. If used in a model, must be saved and loaded via pickle

build(input_shapes)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (*TensorFlow Tensor or Tensor-like*) – The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

classmethod from_layer(layer)

Create a layer from an instance of another layer

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SparseMultiConv3D module

class beyondml.tflow.layers.SparseMultiConv3D.SparseMultiConv3D(*args, **kwargs)

Bases: Layer

Sparse implementation of the MultiConv layer. If used in a model, must be saved and loaded via pickle

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

classmethod from_layer(layer)

Create a layer from an instance of another layer

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SparseMultiDense module

class beyondml.tflow.layers.SparseMultiDense.SparseMultiDense(*args, **kwargs)

Bases: Layer

Sparse implementation of the MultiDense layer. If used in a model, must be saved and loaded via pickle

build(input_shape)

Build the layer in preparation to be trained or called. Should not be called directly, but rather is called when the layer is added to a model

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (*TensorFlow Tensor or Tensor-like*) – The inputs to the layer

Returns

outputs – The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

classmethod from_layer(layer)

Create a layer from an instance of another layer

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that *get_config()* does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

beyondml.tflow.layers.SumLayer module

```
class beyondml.tflow.layers.SumLayer.SumLayer(*args, **kwargs)
```

Bases: Layer

Layer which adds all inputs together. All inputs must have compatible shapes

Example:

```
>>> # Create a model with just a SumLayer and two inputs
>>> input_1 = tf.keras.layers.Input(10)
>>> input_2 = tf.keras.layers.SumLayer()([input_1, input_2])
>>> sum_layer = mann.layers.SumLayer()([input_1, input_2], sum_layer)
>>> model = tf.keras.models.Model([input_1, input_2], sum_layer)
>>> model.compile()
>>> # Call the model
>>> data = np.arange(10).reshape((1, 10))
>>> model.predict([data, data])
array([[ 0., 2., 4., 6., 8., 10., 12., 14., 16., 18.]], dtype=float32)
```

call(inputs)

This is where the layer's logic lives and is called upon inputs

Parameters

inputs (TensorFlow Tensor or Tensor-like) - The inputs to the layer

Returns

outputs - The outputs of the layer's logic

Return type

TensorFlow Tensor

classmethod from_config(config)

Creates a layer from its config.

This method is the reverse of *get_config*, capable of instantiating the same layer from the config dictionary. It does not handle layer connectivity (handled by Network), nor weights (handled by *set_weights*).

Parameters

config – A Python dictionary, typically the output of get_config.

Returns

A layer instance.

get_config()

Returns the config of the layer.

A layer config is a Python dictionary (serializable) containing the configuration of a layer. The same layer can be reinstantiated later (without its trained weights) from this configuration.

The config of a layer does not include connectivity information, nor the layer class name. These are handled by *Network* (one layer of abstraction above).

Note that $get_config()$ does not guarantee to return a fresh copy of dict every time it is called. The callers should make a copy of the returned dict if they want to modify it.

Returns

Python dictionary.

Module contents

Custom layers to use when building MANN models

beyondml.tflow.utils package

Submodules

beyondml.tflow.utils.transformer module

beyondml.tflow.utils.transformer.build_token_position_embedding_block(sequence_length, vocab_size, embed_dim)

Builds a token and position embedding block

Parameters

- **sequence_length** (*int*) The length of each sequence
- vocab_size (int) The size of the vocabulary used
- **embed_dim** (*int*) The desired embedding dimension

Returns

embedding_block – The embedding block, which can be used alone or as a layer in another model

Return type

TensorFlow keras Functional model

beyondml.tflow.utils.transformer.build_transformer_block(input_shape, embed_dim, num_heads, neurons, dropout_rate=0.1)

Build a Transformer Block

Parameters

- input_shape (int or tuple of int) The input shape for the model to use
- **embed_dim** (*int*) The dimension of the embedding
- **num_heads** (*int*) The number of attention heads to use
- **neurons** (*int*) The number of hidden neurons to use in the hidden layer
- **dropout_rate** (*float* (*default* 0.1)) Rate at which dropout is applied
- value_dim (int or None (default None)) The dimension to use for the value matrix, if provided

Returns

 ${\bf transformer_block} - {\bf The\ transformer\ block}, which \ can \ then\ be\ used\ alone\ or\ as\ a\ layer\ in\ another\ model$

Return type

TensorFlow keras Functional model

beyondml.tflow.utils.utils module

class beyondml.tflow.utils.utils.ActiveSparsification(performance_cutoff,

performance_measure='auto', starting_sparsification=None, max_sparsification=99, sparsification_rate=1, sparsification_patience=10, stopping_delta=0.01, stopping_patience=5, restore_best_weights=True, verbose=1)

Bases: Callback

Keras-compatible callback object which enables active sparsification, allowing for increased sparsification as models train.

on_epoch_end(epoch, logs=None)

Called at the end of an epoch.

Subclasses should override for any actions to run. This function should only be called during TRAIN mode.

Parameters

- **epoch** Integer, index of epoch.
- **logs** Dict, metric results for this training epoch, and for the validation epoch if validation is performed. Validation result keys are prefixed with *val*_. For training epoch, the values of the *Model*'s metrics are returned. Example: ['loss': 0.2, 'accuracy': 0.7].

on_train_begin(logs=None)

Called at the beginning of training.

Subclasses should override for any actions to run.

Parameters

logs – Dict. Currently no data is passed to this argument for this method but that may change in the future.

beyondml.tflow.utils.utils.add_layer_masks(model, additional custom objects=None)

Convert a trained model from one that does not have masking weights to one that does have masking weights

Parameters

- model (TensorFlow Keras model) The model to be converted
- additional_custom_objects(dict or None (default None)) Additional custom layers to use

Returns

new model - The converted model

Return type

TensorFlow Keras model

beyondml.tflow.utils.utils.get_custom_objects()

Return a dictionary of custom objects (layers) to use when loading models trained using this package

 $beyond \verb|ml.tflow.utils.utils.get_task_masking_gradients|(model, task_num)|$

Get the gradients of masking weights within a model

Parameters

model (TensorFlow Keras model) – The model to retrieve the gradients of

Notes

• This function should only be run before the model has been trained

or used to predict. There is an unknown bug related to TensorFlow which is leading to incorrect results after initial training

When running this function, randomized input and output data is sent

through the model to retrieve gradients respective to each task. If the model is compiled using <code>sparse_categorical_crossentropy</code> 'loss, this will break this function's functionality. As a result, please use 'categorical_crossentropy (or even better, mse) before running this function. After retrieving gradients, the model can be recompiled with whatever parameters are desired.

Returns

gradients – The gradients of the masking weights of the model

Return type

list of TensorFlow tensors

beyondml.tflow.utils.utils.mask_model(model, percentile, method='gradients', exclusive=True, x=None, y=None)

Mask the multitask model for training respective using the gradients for the tasks at hand

Parameters

- model (keras model with MANN masking layers) The model to be masked
- **percentile** (*int*) Percentile to use in masking. Any weights less than the *percentile* value will be made zero
- **method** (*str* (*default* '*gradients*')) One of either 'gradients' or 'magnitude' the method for how to identify weights to mask If method is 'gradients', utilizes the gradients with respect to the passed x and y variables to identify the subnetwork to activate for each task If method is 'magnitude', uses the magnitude of the weights to identify the subnetwork to activate for each task
- **exclusive** (*bool* (*default True*)) Whether to restrict previously-used weight indices for each task. If *True*, this identifies disjoint subsets of weights within the layer which perform the tasks requested.
- **x** (*list of np.ndarray or array-like*) The training data input values, ignored if "method" is 'magnitude'
- **y** (*list of np.ndarray or array-like*) The training data output values, ignored if "method" is 'magnitude'

beyondml.tflow.utils.utils.mask_task_weights(model, task_masking_gradients, percentile, respect_previous_tasks=True)

Parameters

- $\bullet \ \ model \ (\textit{TensorFlow Keras model}) The \ model \ to \ be \ masked$
- task_masking_gradients (list of TensorFlow tensors) The gradients for the specific task requested
- **percentile** (*int*) The percentile to mask/prune
- respect_previous_tasks (bool (default True)) Whether to respect the weights used for previous tasks and not use them for subsequent tasks

Returns

masked model – The masked model

Return type

TensorFlow Keras model

beyondml.tflow.utils.utils.quantize_model(model, dtype='float16', additional_custom_objects=None)
Apply model quantization

Parameters

- model (TensorFlow Keras Model) The model to quantize
- dtype(str or TensorFlow datatype (default 'float16')) The datatype to quantize to
- additional_custom_objects(None or dict (default None)) Additional custom objects to use to instantiate the model

Returns

new_model - The quantized model

Return type

TensorFlow Keras Model

beyondml.tflow.utils.utils.replace_config(config)

Replace the model config to remove masking layers

beyondml.tflow.utils.utils.replace_weights(new_model, old_model)

Replace the weights of a newly created model with the weights (sans masks) of an old model

beyondml.tflow.utils.utils.train_model(model, train_x, train_y, loss, metrics, optimizer, cutoff, batch_size=32, epochs=100, starting_sparsification=0, max_sparsification=99, sparsification_rate=5, sparsification_patience=10, stopping_patience=5)

beyondml.tflow.utils.utils.train_model_iteratively(model, task_gradients, train_x, train_y, validation_split, delta, batch_size, losses, optimizer='adam', metrics=None, starting_pruning=0, pruning_rate=10, patience=5, max_epochs=100)

Train a model iteratively on each task, first obtaining baseline performance on each task and then iteratively training and pruning each task as far back as possible while maintaining acceptable performance on each task

Parameters

- model (TensorFlow Keras model) The model to be trained
- task_gradients (list of TensorFlow tensors) Gradients for each task, output from the get_task_masking_gradients function
- train_x (list of numpy arrays, TensorFlow Datasets, or other)—data types models can train with The input data to use to train on
- **train_y**(*list of numpy arrays*, *TensorFlow Datasets*, *or other*)—data types model can train with The output data to use to train on
- validation_split (float, or list of float) The proportion of data to use for validation
- **delta** (*float*) The tolerance between validation losses to be considered "acceptable" performance to continue

- batch_size (int) The batch size to train with
- **losses** (*str*, *list*, *or Keras loss function*) The loss or losses to use when training
- **optimizer** (*str*, *list*, *or Keras optimizer*) The optimizer to use when training (default 'adam')
- starting_pruning (int or list of int (default 0)) The starting pruning rate to use for each task
- pruning_rate (int or list of int (default [10, 5, 2, 1])) The pruning rate to use
- **patience** (*int* (*default* 5)) The patience for number of epochs to wait for performance to improve sufficiently
- max_epochs (int or list of int (default 100)) The maximum number of epochs to use for training each task

Module contents

Some utilities to use when building, loading, and training MANN models

Module contents

TensorFlow compatibility for building MANN models.

The *beyondml.tflow* package contains two subpackages, *beyondml.tflow.layers* and *beyondml.tflow.utils*, which contain the functionality to create and train MANN layers within TensorFlow. For individuals who are familiar with the former name of this package, *mann*, backwards compatibility can be achieved (assuming only TensorFlow support is needed), by replacing the following line of code:

```
>>> import mann
```

with the following line:

```
>>> import beyondml.tflow as mann
```

in all existing scripts.

Within the *layers* package, there is current functionality for the the following layers: - *beyondml.tflow.layers.FilterLayer* - *beyondml.tflow.layers.MaskedConv2D* - *beyondml.tflow.layers.MaskedDense* - *beyondml.tflow.layers.MultiDense* - *beyondml.tflow.layers.MultiMaskedConv2D* - *beyondml.tflow.layers.MultiMaskedDense* - *beyondml.tflow.layers.MultiMaxPool2D* - *beyondml.tflow.layers.SelectorLayer* - *beyondml.tflow.layers.SumLayer* - *beyondml.tflow.layers.SparseDense* - *beyondml.tflow.layers.SparseConv* - *beyondml.tflow.layers.SparseMultiDense* - *beyondml.tflow.layers.SparseMultiConv*

Note that with any of the sparse layers (such as the `SparseDense` layer), any model which utilizes these layers will not be loadable using the traditional `load_model` functions available in TensorFlow. Instead, the model should be saved using either joblib or pickle.

Within the *utils* package, there are the current functions and classes: - *ActiveSparsification* - *build_transformer_block* - *build_token_position_embedding_block* - *get_custom_objects* - *mask_model* - *remove_layer_masks* - *add_layer_masks* - *quantize_model* - *get_task_masking_gradients* - *mask_task_weights* - *train_model_iteratively*

Module contents

BeyondML (formerly MANN) is a Python package which enables creating sparse multitask artificial neural networks (MANNs) compatible with [TensorFlow](https://tensorflow.org) and [PyTorch](https://pytorch.org). This package contains custom layers and utilities to facilitate the training and optimization of models using the Reduction of Sub-Network Neuroplasticity (RSN2) training procedure developed by [AI Squared, Inc](https://squared.ai).

Installation

This package is available through [PyPi](https://pypi.org) and can be installed via the following command:

`bash pip install beyondml

Capabilities

There are two major subpackages within the BeyondML package, the *beyondml.tflow* and the *beyondml.pt* packages. The *beyondml.tflow* package contains functionality for building multitask models using TensorFlow, and the *beyondml.pt* package contains functionality for building multitask models using PyTorch.

CHANGELOG

• Version 0.1.0

- Refactored existing MANN repository to rename to BeyondML

• Version 0.1.1

- Added the *SparseDense*, *SparseConv*, *SparseMultiDense*, and *SparseMultiConv* layers to beyondml.tflow.layers, giving users the functionality to utilize sparse tensors during inference

Version 0.1.2

- Added the MaskedMultiHeadAttention, MaskedTransformerEncoderLayer, and MaskedTransformerDecoderLayer layers to beyondml.pt.layers to add pruning to the transformer architecture
- Added MaskedConv3D, MultiMaskedConv3D, MultiConv3D, MultiMaxPool3D, SparseConv3D, and SparseMultiConv3D layers to beyondml.tflow.layers
- Added MaskedConv3D, MultiMaskedConv3D, MultiConv3D, MultiMaxPool3D, SparseConv3D, SparseMultiConv3D, and MultiMaxPool2D layers to beyondml.pt.layers

• Version 0.1.3

- Added beyondml.pt compatibility with more native PyTorch functionality for using models on different devices and datatypes
- Added train_model function to beyondml.tflow.utils
- Added MultitaskNormalization layer to beyondml.tflow.layers and beyondml.pt.layers

• Version 0.1.4

- Updated documentation to use Sphinx

• Version 0.1.5

- Updated requirements to use newer version of TensorFlow
- Fixed errors with changes to types of *input_shape* in TensorFlow Keras layers
- Fixed errors resulting from model/configuration changes with TensorFlow

• Version 0.1.6

- Fixed issues with converting between masked and unmasked models in TensorFlow

• Version 0.1.7

- Updated Pytorch implementation of Transformer-based architectures

PYTHON MODULE INDEX

```
b
                                               beyondml.tflow.layers.MultiConv3D, 24
                                               beyondml.tflow.layers.MultiDense, 25
beyondml, 45
                                               beyondml.tflow.layers.MultiMaskedConv2D, 26
beyondml.pt, 18
                                               beyondml.tflow.layers.MultiMaskedConv3D, 27
beyondml.pt.layers, 18
                                               beyondml.tflow.layers.MultiMaskedDense, 28
beyondml.pt.layers.Conv2D, 3
                                               beyondml.tflow.layers.MultiMaxPool2D, 29
beyondml.pt.layers.Conv3D, 4
                                               beyondml.tflow.layers.MultiMaxPool3D, 30
beyondml.pt.layers.Dense, 4
                                               beyondml.tflow.layers.MultitaskNormalization,
beyondml.pt.layers.FilterLayer, 4
beyondml.pt.layers.MaskedConv2D, 5
                                               beyondml.tflow.layers.SelectorLayer, 31
beyondml.pt.layers.MaskedConv3D, 6
                                               beyondml.tflow.layers.SparseConv2D, 33
beyondml.pt.layers.MaskedDense, 6
                                               beyondml.tflow.layers.SparseConv3D, 34
beyondml.pt.layers.MaskedMultiHeadAttention,
                                               beyondml.tflow.layers.SparseDense, 35
beyondml.pt.layers.MaskedTransformerDecoderLayer, heyondml.tflow.layers.SparseMultiConv2D, 36
                                               beyondml.tflow.layers.SparseMultiConv3D, 37
beyondml.pt.layers.MaskedTransformerEncoderLayerYondml.tflow.layers.SparseMultiDense, 38
                                               beyondml.tflow.layers.SumLayer, 39
                                               beyondml.tflow.utils,44
beyondml.pt.layers.MultiConv2D, 11
                                               beyondml.tflow.utils.transformer, 40
beyondml.pt.layers.MultiConv3D, 11
                                               beyondml.tflow.utils.utils,41
beyondml.pt.layers.MultiDense, 11
beyondml.pt.layers.MultiMaskedConv2D, 12
beyondml.pt.layers.MultiMaskedConv3D, 13
beyondml.pt.layers.MultiMaskedDense, 13
beyondml.pt.layers.MultiMaxPool2D, 14
beyondml.pt.layers.MultiMaxPool3D, 14
beyondml.pt.layers.MultitaskNormalization, 15
beyondml.pt.layers.SelectorLayer, 15
beyondml.pt.layers.SparseConv2D, 15
beyondml.pt.layers.SparseConv3D, 16
beyondml.pt.layers.SparseDense, 16
beyondml.pt.layers.SparseMultiConv2D, 16
beyondml.pt.layers.SparseMultiConv3D, 17
beyondml.pt.layers.SparseMultiDense, 17
beyondml.pt.utils, 18
beyondml.pt.utils.utils, 18
beyondml.tflow, 44
beyondml.tflow.layers, 40
beyondml.tflow.layers.FilterLayer, 19
beyondml.tflow.layers.MaskedConv2D, 20
beyondml.tflow.layers.MaskedConv3D, 21
beyondml.tflow.layers.MaskedDense, 22
beyondml.tflow.layers.MultiConv2D, 23
```

50 Python Module Index

INDEX

A	module, 13
ActiveSparsification (class in be-	beyondml.pt.layers.MultiMaskedDense
yondml.tflow.utils.utils), 41	module, 13
add_layer_masks() (in module be-	beyondml.pt.layers.MultiMaxPool2D
yondml.tflow.utils.utils), 41	module, 14
	beyondml.pt.layers.MultiMaxPool3D
В	module, 14
beyondml	beyondml.pt.layers.MultitaskNormalization
module, 45	module, 15
beyondml.pt	beyondml.pt.layers.SelectorLayer
module, 18	module, 15
beyondml.pt.layers	beyondml.pt.layers.SparseConv2D
module, 18	module, 15
beyondml.pt.layers.Conv2D	beyondml.pt.layers.SparseConv3D
module, 3	module, 16
beyondml.pt.layers.Conv3D	beyondml.pt.layers.SparseDense
module, 4	module, 16
beyondml.pt.layers.Dense	beyondml.pt.layers.SparseMultiConv2D
module, 4	module, 16
beyondml.pt.layers.FilterLayer	beyondml.pt.layers.SparseMultiConv3D
module, 4	module, 17
beyondml.pt.layers.MaskedConv2D	beyondml.pt.layers.SparseMultiDense
module, 5	module, 17
beyondml.pt.layers.MaskedConv3D	beyondml.pt.utils
module, 6	module, 18
beyondml.pt.layers.MaskedDense	beyondml.pt.utils.utils
module, 6	module, 18
beyondml.pt.layers.MaskedMultiHeadAttention	beyondml.tflow
modulo 7	module, 44
beyondml.pt.layers.MaskedTransformerDecoderLa	beyondmi.tilow.layers lyer
module, 8 beyondml.pt.layers.MaskedTransformerEncoderLa	beyondmi.tilow.layers.riiterLayer lyer
module, 9	
beyondml.pt.layers.MultiConv2D	beyondml.tflow.layers.MaskedConv2D
module, 11	module, 20
beyondml.pt.layers.MultiConv3D	beyondml.tflow.layers.MaskedConv3D
module, 11	module, 21
beyondml.pt.layers.MultiDense	<pre>beyondml.tflow.layers.MaskedDense module, 22</pre>
module, 11	beyondml.tflow.layers.MultiConv2D
beyondml.pt.layers.MultiMaskedConv2D	module, 23
module, 12	beyondml.tflow.layers.MultiConv3D
beyondml.pt.layers.MultiMaskedConv3D	beyondmi.ciiow.iayeis.nuiciconv3D

module, 24	build() (beyondml.tflow.layers.MultiMaskedDense.MultiMaskedDense
beyondml.tflow.layers.MultiDense	method), 28
module, 25	$\verb build() (beyond ml. tflow. layers. Multitask Normalization. Multit$
beyondml.tflow.layers.MultiMaskedConv2D	method), 30
module, 26	build() (beyondml.tflow.layers.SparseConv2D.SparseConv2D
beyondml.tflow.layers.MultiMaskedConv3D	method), 33
module, 27	build() (beyondml.tflow.layers.SparseConv3D.SparseConv3D
beyondml.tflow.layers.MultiMaskedDense	method), 34
module, 28	build() (beyondml.tflow.layers.SparseDense.SparseDense
beyondml.tflow.layers.MultiMaxPool2D	method), 35
module, 29	build() (beyondml.tflow.layers.SparseMultiConv2D.SparseMultiConv2D
beyondml.tflow.layers.MultiMaxPool3D	method), 36
module, 30	build() (beyondml.tflow.layers.SparseMultiConv3D.SparseMultiConv3D
beyondml.tflow.layers.MultitaskNormalization	method), 37
module, 30	build() (beyondml.tflow.layers.SparseMultiDense.SparseMultiDense
beyondml.tflow.layers.SelectorLayer	method), 38
module, 31	build_token_position_embedding_block() (in
beyondml.tflow.layers.SparseConv2D	module beyondml.tflow.utils.transformer), 40
module, 33	build_transformer_block() (in module be-
beyondml.tflow.layers.SparseConv3D	yondml.tflow.utils.transformer), 40
module, 34	C
beyondml.tflow.layers.SparseDense	
module, 35	call() (beyondml.tflow.layers.FilterLayer.FilterLayer
beyondml.tflow.layers.SparseMultiConv2D	method), 19
module, 36	call() (beyondml.tflow.layers.MaskedConv2D.MaskedConv2D
beyondml.tflow.layers.SparseMultiConv3D	method), 20
module, 37	call() (beyondml.tflow.layers.MaskedConv3D.MaskedConv3D
beyondml.tflow.layers.SparseMultiDense	method), 21
module, 38	call() (beyondml.tflow.layers.MaskedDense.MaskedDense
beyondml.tflow.layers.SumLayer module,39	method), 22
beyondml.tflow.utils	call() (beyondml.tflow.layers.MultiConv2D.MultiConv2D
module, 44	method), 23
beyondml.tflow.utils.transformer	call() (beyondml.tflow.layers.MultiConv3D.MultiConv3D
module, 40	method), 24
beyondml.tflow.utils.utils	call() (beyondml.tflow.layers.MultiDense.MultiDense
module, 41	method), 25
build() (beyondml.tflow.layers.MaskedConv2D.MaskedC	call() (beyondml.tflow.layers.MultiMaskedConv2D.MultiMaskedConv2D
method), 20	<i></i>
build() (beyondml.tflow.layers.MaskedConv3D.MaskedC	call() (beyondml.tflow.layers.MultiMaskedConv3D.MultiMaskedConv3D
method), 21	memou), 27
build() (beyondml.tflow.layers.MaskedDense.MaskedDen	call() (beyondml.tflow.layers.MultiMaskedDense.MultiMaskedDense
method), 22	mentod), 20
build() (beyondml.tflow.layers.MultiConv2D.MultiConv2	call() (beyondml.tflow.layers.MultiMaxPool2D.MultiMaxPool2D
method), 23	,, =-
build() (beyondml.tflow.layers.MultiConv3D.MultiConv3	call() (beyondml.tflow.layers.MultiMaxPool3D.MultiMaxPool3D
method), 24	memour), eo
build() (beyondml.tflow.layers.MultiDense.MultiDense	call() (beyondml.tflow.layers.MultitaskNormalization.MultitaskNormal
method), 25	method), 31
build() (beyondml.tflow.layers.MultiMaskedConv2D.Mul	call() (beyondml.tflow.layers.SelectorLayer.SelectorLayer tiMaskedConv2D.p. 222
method), 26	
build() (beyondml.tflow.layers.MultiMaskedConv3D.Mul	call() (beyondml.tflow.layers.SparseConv2D.SparseConv2D tiMaskedCanv3D.p. 22
method), 27	, ,
monod), 21	call() (beyondml.tflow.layers.SparseConv3D.SparseConv3D
	method), 34

call() (beyondml.tflow.layers.SparseDense.SparseDense method), 35	forward() (beyondml.pt.layers.MultiMaxPool3D.MultiMaxPool3D method), 14
	P. MatriCardi QD beyondml.pt.layers.MultitaskNormalization.MultitaskNormalization, 15
	method), 15 eMairiGwalQDbeyondml.pt.layers.SelectorLayer.SelectorLayer
method), 37	method), 15
	InditiDeause() (beyondml.pt.layers.SparseConv2D.SparseConv2D method), 15
call() (beyondml.tflow.layers.SumLayer.SumLayer method), 39	
Conv2D (class in beyondml.pt.layers.Conv2D), 3	forward() (beyondml.pt.layers.SparseDense.SparseDense
Conv3D (class in beyondml.pt.layers.Conv3D), 4	method), 16 forward() (hover draft at layers SpanseMultiComy2D SpanseMultiComy2D)
D	forward() (beyondml.pt.layers.SparseMultiConv2D.SparseMultiConv2D method), 16
Dense (class in beyondml.pt.layers.Dense), 4	forward() (beyondml.pt.layers.SparseMultiConv3D.SparseMultiConv3D method), 17
F	forward() (beyondml.pt.layers.SparseMultiDense.SparseMultiDense method), 17
FilterLayer (class in beyondml.pt.layers.FilterLayer),	from_config() (beyondml.tflow.layers.FilterLayer.FilterLayer
4 Filter aver (class in be	class method), 20
FilterLayer (class in be- yondml.tflow.layers.FilterLayer), 19	<pre>from_config() (beyondml.tflow.layers.MaskedConv2D.MaskedConv2D</pre>
forward() (beyondml.pt.layers.Conv2D.Conv2D	class method), 21
method), 3	${\tt from_config()} \ (beyond ml. {\it tflow.layers.} Masked Conv3D. Masked Conv3D$
forward() (beyondml.pt.layers.Conv3D.Conv3D	class method), 22
method), 4	from_config() (beyondml.tflow.layers.MaskedDense.MaskedDense
<pre>forward() (beyondml.pt.layers.Dense.Dense method), 4</pre>	class method), 23
${\tt forward()} \textit{(beyondml.pt.layers.FilterLayer.FilterLayer)}$	from_config() (beyondml.tflow.layers.MultiConv2D.MultiConv2D
method), 4	class method), 24 from config () (heyondral tflow layers MultiConv3D MultiConv3D)
forward() (beyondml.pt.layers.MaskedConv2D.M	from_config() (beyondml.tflow.layers.MultiConv3D.MultiConv3D class method), 24
method), 5	from_config() (beyondml.tflow.layers.MultiDense.MultiDense
1 1 (cuss memous, 25
method), 6 forward() (havendral at lovers MaskedDense MaskedDen	from_config() (beyondml.tflow.layers.MultiMaskedConv2D.MultiMasked
mathad) 6	ciass memoa), 20
forward() (beyondml.pt.layers.MaskedMultiHeadAttentio method), 7	n from config() (bayondral tflow.layers.MultiMaskedConv3D.MultiMasked class method), 27
forward() (beyondml.pt.layers.MaskedTransformerDecod	lerLayer.MaskedTransformerDecoderLayers.MultiMaskedDense.MultiMaskedD class method), 28
forward() (beyondml.pt.layers.MaskedTransformerEncod	lerEayer.Masked FransformerEncoderEayer MultiMaxPool2D.MultiMaxPool2L class method), 29
method), 11 forward() (hevondral at lovers MultiConv2D MultiConv2	from_config() (beyondml.tflow.layers.MultiMaxPool3D.MultiMaxPool3L
44 at la a d\ 11	ciass memoa), 50
forward() (beyondml.pt.layers.MultiConv3D.MultiConv3 method), 11	pfrom_config() (beyondml.tflow.layers.MultitaskNormalization.Multita
forward() (beyondml.pt.layers.MultiDense.MultiDense	<pre>from_config() (beyondml.tflow.layers.SelectorLayer.SelectorLayer</pre>
	class method), 32
forward() (beyondml.pt.layers.MultiMaskedConv2D.Mult	tiMaskeaConv2D (beyondml.tflow.layers.SparseConv2D.SparseConv2D class method), 33
forward() (beyondml.pt.layers.MultiMaskedConv3D.Multi	timaskeaConv3D (beyondml.tflow.layers.SparseConv3D.SparseConv3D class method), 34
1 1 1 2	Afrom Sonfig() (beyondml.tflow.layers.SparseDense.SparseDense class method), 35
method), 13 forward() (hevendral at lovers MultiMaxPool2D MultiMax	from config() (beyondml.tflow.layers.SparseMultiConv2D.SparseMulti
method), 14	class method), 36

from_config() (beyondml.tflow.layers.SparseMultiConv3lg.Space class method), 37	SarMing(Oche3Dndml.tflow.layers.SparseMultiConv2D.SparseMultiComethod), 36
	Mcthod), 30 MthdigQn(beyondml.tflow.layers.SparseMultiConv3D.SparseMultiComethod), 37
	onfig() (beyondml.tflow.layers.SparseMultiDense.SparseMultiDense.method), 38
from_layer() (beyondml.tflow.layers.SparseConv2D.Spar.getoro class method), 33	
from_layer() (beyondml.tflow.layers.SparseConv3D.Spar.gactoroclass method), 34	
from_layer() (beyondml.tflow.layers.SparseDense.SparseDense&class method), 35	ask_masking_gradients() (in module be- yondml.tflow.utils.utils), 41
from_layer() (beyondml.tflow.layers.SparseMultiConv2D.Sparse class method), 36	
from_layer() (beyondml.tflow.layers.SparseMultiConv3D.Sparse class method), 37	property). 5
from_layer() (beyondml.tflow.layers.SparseMultiDense.SparseM class method), 38	httiDrs Beyondml.pt.layers.MaskedConv3D.MaskedConv3D property), 6
G	annels (beyondml.pt.layers.MultiMaskedConv2D.MultiMaskedConv property), 12
method), 20	annels (beyondml.pt.layers.MultiMaskedConv3D.MultiMaskedConv property), 13
get_config() (beyondml.tflow.layers.MaskedConv2D.Mas ks d 6n method), 21	erty), 5
get_config() (beyondml.tflow.layers.MaskedConv3D.MaskedCo method), 22	
get_config() (beyondml.tflow.layers.MaskedDense.MaskedDense method), 23	property) 5
get_config() (beyondml.tflow.layers.MultiConv2D.MultiGenv2L method), 24	nronarty) 6
methoa), 25	1_size(beyondml.pt.layers.MultiMaskedConv2D.MultiMaskedConv property), 12
method), 26	1_size(beyondml.pt.layers.MultiMaskedConv3D.MultiMaskedConv property), 13
get_config() (beyondml.tflow.layers.MultiMaskedConv2DeMulti method), 26	property) 21
get_config() (beyondml.tflow.layers.MultiMaskedConv3D _k Multi method), 27	property) 22
get_config() (beyondml.tflow.layers.MultiMaskedDense.Mult	property), 24
get_config() (beyondml.tflow.layers.MultiMaxPool2D.MultiMax method), 29	property), 25
method), 30	Past 20 (beyondml.tflow.layers.MultiMaskedConv2D.MultiMaskedCoproperty), 27
method), 31 get_config() (beyondml.tflow.layers.SelectorLayer.SelectorLaye	dtites Norgaliathicthow.layers.MultiMaskedConv3D
method), 32	
<pre>get_config() (beyondml.tflow.layers.SparseConv2D.SparseConv mask_ method), 33 mask_ get_config() (beyondml.tflow.layers.SparseConv3D.SparseConv</pre>	task_weights() (in module be-
method), 34 get_config() (beyondml.tflow.layers.SparseDense.SparseDense	yonami.ijiow.uiiis.uiiis), 42
method), 35	yondml.pt.layers.MaskedConv2D), 5

MaskedConv2D	(class	in	be-	beyondml.pt.ut			
•	w.layers.Maskea	lConv2D), 20		beyondml.tflow			
MaskedConv3D	(class	in	be-	beyondml.tflow	.layers, 40		
	layers.MaskedCo	pnv3D), 6		beyondml.tflow	ı.layers.Filt	erLayer, 1	9
MaskedConv3D	(class	in	be-	beyondml.tflow	ı.layers.Mask	cedConv2D,	20
yondml.tflo	w.layers.Maskea	lConv3D), 21		beyondml.tflow	ı.layers.Mask	cedConv3D,	21
MaskedDense	(class	in	be-	beyondml.tflow	.layers.Mask	cedDense, 2	2
yondml.pt.l	layers.MaskedDe	ense), 6		beyondml.tflow	v.layers.Mult	ciConv2D, 2	3
MaskedDense	(class	in	be-	beyondml.tflow	.layers.Mult	ciConv3D, 2	4
yondml.tflo	w.layers.Maskea	lDense), 22		beyondml.tflow	.layers.Mult	iDense, 25	
MaskedMultiHeadA	ttention (class in	be-	beyondml.tflow	.layers.Mult	iMaskedCo	nv2D,
yondml.pt.l	layers.MaskedMi	ultiHeadAtten	ition),	26	-		
7	•			beyondml.tflow	.layers.Mult	iMaskedCo	nv3D,
MaskedTransforme	rDecoderLayer	c (class i	n be-	27	-		
	layers.MaskedTr		coderLaver).	beyondml.tflow	.lavers.Mult	iMaskedDe	nse.
8		J	, , ,	28	,		,
MaskedTransforme	rEncoderLave	c (class i	n be-	beyondml.tflow	.lavers.Mult	iMaxPool2	D. 29
	layers.MaskedTr						
9		,	, ,	beyondml.tflow			
module				30	.,		
beyondml, 45				beyondml.tflow	.lavers.Sele	ctorLaver	. 31
beyondml.pt,	18			beyondml.tflow			
beyondml.pt.				beyondml.tflow	-		
-	layers.Conv2I) 3		beyondml.tflow	-		
-	layers.Conv3I			beyondml.tflow	-		
-	layers.Dense,			36		Dellar er co	,
-	layers.Filter			beyondml.tflow	.lavers.Spar	rseMultiCo	nv3D
	layers.Masked			37		Dellar er co	111 32,
	layers.Masked			beyondml.tflow	, lavers Snar	rseMultiDe	nse
-	layers.Masked			38	···uy ci b. bpui	Schultbe	iioc,
-	layers.Masked		ttention	beyondml.tflow	, lavers Simi	aver 30	
7	rayers.nasked	mur crineaux	iccencion,	beyondml.tflow	-	layer, 37	
hovondml nt	lavore Macked	Transforme	rDecoderI	aykeeryondml.tflow		former 40	
8	rayers.nasked	iii ansioime	i Decouei La	beyondml.tflow			
O .	layers.Masked	Transforme	rEncodeMu	-	(class	in	be-
g	iayeis.nasket	iii alistoime	ET ETICOGEM U	-	yers.MultiConv2		De-
	layers.Multi(Conv2D 11	M117	yonαπα.ρι.ια _. LtiConv2D	class	in	be-
-	layers.MultiC layers.MultiC		nul		.layers.MultiCoi		De-
•	•	,	м., 7	yonama.gow LtiConv3D			ha
-	layers.MultiD				(class	in 2D) 11	be-
-	layers.MultiN				yers.MultiConv3		h a
-	layers.MultiN			ltiConv3D	(class	in 2D) 24	be-
	layers.MultiN			•	.layers.MultiCoi		3)
-	layers.MultiM			ltiDense (<i>class ii</i>	ı <i>реуопат</i> і.рі.ы	ayers.MuitiL	Jense),
	layers.MultiN			11	(1		7
	layers.Multit	asknormali	zation, Mul		(class	in	be-
15	1 C-1	15			.layers.MultiDei	* * * * * * * * * * * * * * * * * * * *	1
-	layers.Select		Mul	ltiMaskedConv2D	(class	in	be-
-	layers.Sparse			1	yers.MultiMaske	eaConv2D),	
	layers.Sparse			12	/ 1		7
	layers.Sparse			ltiMaskedConv2D	(class	in	be-
-	layers.Sparse			•	.layers.MultiMa	skedConv2L	J),
-	layers.Sparse			26			7
-	layers.Sparse	emultiDense	e, 17 Mul	ltiMaskedConv3D	(class	in	be-
hevondml.nt.	DTIIS IX			vondml nt la	vers MultiMaske	Pat onv3D)	

13	,	prune() (beyondml.pt.layers.MultiMaskedConv3D.MultiMaskedConv3D
MultiMaskedConv3D (class in	be-	method), 13
yondml.tflow.layers.MultiMaskedConv31 27	9),	<pre>prune() (beyondml.pt.layers.MultiMaskedDense.MultiMaskedDense</pre>
MultiMaskedDense (class in	be-	prune_model() (in module beyondml.pt.utils.utils), 18
yondml.pt.layers.MultiMaskedDense), 13		prune_moder() (in modute beyondmi.pr.ditis.ditis), 10
MultiMaskedDense (class in	be-	Q
yondml.tflow.layers.MultiMaskedDense)		<pre>quantize_model() (in module be-</pre>
28	,	yondml.tflow.utils.utils), 43
MultiMaxPool2D (class in	be-	
yondml.pt.layers.MultiMaxPool2D), 14		R
MultiMaxPool2D (class in	be-	replace_config() (in module be-
yondml.tflow.layers.MultiMaxPool2D), 2	29	yondml.tflow.utils.utils), 43
MultiMaxPool3D (class in	be-	replace_weights() (in module be-
yondml.pt.layers.MultiMaxPool3D), 14	_	yondml.tflow.utils.utils), 43
MultiMaxPool3D (class in	be-	0
yondml.tflow.layers.MultiMaxPool3D), 3		S
MultitaskNormalization (class in	be-	$\verb sel_index (beyond ml. pt. layers. Selector Layer. Selector Layer$
yondml.pt.layers.MultitaskNormalization	1),	property), 15
MultitaskNormalization (class in	be-	$\verb sel_index (beyond ml.t flow. layers. Selector Layer. Selector Layer$
yondml.tflow.layers.MultitaskNormalization		property), 32
30	ion),	SelectorLayer (class in be-
_		yondml.pt.layers.SelectorLayer), 15
0		SelectorLayer (class in be-
on epoch end() (beyondml.tflow.utils.utils.Activ	eSparsifi	yondml.tflow.layers.SelectorLayer), 31 acation_masks() (beyondml.tflow.layers.MaskedConv2D.MaskedConv2D
method), 41		method), 21
on_train_begin()	(be-	set_masks() (beyondml.tflow.layers.MaskedConv3D.MaskedConv3D
yondml.tflow.utils.utils. Active Sparsification for the property of the prop	ion	method), 22
method), 41		sat masks () (howardml tflow layers Masked Donse Masked Donse
$\verb"out_channels" (beyond ml. pt. layers. Masked Conv2 to the convergence of the converge$	2D.Mask	redConv2D method), 23
property), 5		sat masks() (hevendml tflow layers MultiMaskedConv2D MultiMaskedCo
$\verb"out_channels" (beyond ml. pt. layers. Masked Conviction and the conviction of th$	BD.Mask	
property), 6	G 2D	set_masks() (beyondml.tflow.layers.MultiMaskedConv3D.MultiMaskedCo
out_channels (beyondml.pt.layers.MultiMasked0	Conv2D.	, ==
property), 12	Cam.2D	set_masks() (beyondml.tflow.layers.MultiMaskedDense.MultiMaskedDen
<pre>out_channels(beyondml.pt.layers.MultiMasked0</pre>	CONV3D.	· · · · · · · · · · · · · · · · · · ·
property), 13		SparseConv2D (class in be-
P		yondml.pt.layers.SparseConv2D), 15
	als a dC and	SparseConv2D (class in be-
<pre>prune() (beyondml.pt.layers.MaskedConv2D.Ma. method), 5</pre>	skeaCon	
prune() (beyondml.pt.layers.MaskedConv3D.Ma.	skadCom	SparseConv3D (class in be-
method), 6	skeuCon	
prune() (beyondml.pt.layers.MaskedDense.Mask	edDense	
method), 6	euDense	SparseDense (class in beyondml.pt.layers.SparseDense),
prune() (beyondml.pt.layers.MaskedMultiHeadA	ttention]	MaskedMultiHeadAttention
method), 7		SparseDense (class in be-
prune() (beyondml.pt.layers.MaskedTransformer.	Decoder	
method), 9		SparseMultiConv2D (class in be-
**	Encoderi	Layer.MaskeditimisfiritmeerEstSpodesdMyteriConv2D), 16
method), 11		SparseMultiConv2D (class in be-
	D.MultiN	MaskedConv2Mdml.tflow.layers.SparseMultiConv2D),
method), 12		36

SparseMultiConv3D	(class	in	be-
yondml.pt.layer:	s.SparseMulti	Conv3D), 1	.7
SparseMultiConv3D	(class	in	be-
yondml.tflow.lay	ers.SparseMu	ltiConv3D),
37			
SparseMultiDense	(class	in	be-
yondml.pt.layer:	s.SparseMulti	Dense), 17	
SparseMultiDense	(class	in	be-
yondml.tflow.lay	ers.SparseMu	ltiDense),	38
SumLayer (class in beyon	dml.tflow.laye	rs.SumLay	er), 39
T train_model() (in mod 43	dule beyondm	ıl.tflow.utils	s.utils),
train_model_iterativ yondml.tflow.uti		module	be-
turn_off() (beyondml.p method), 5	ot.layers.Filter	Layer.Filte	rLayer
turn_off() (beyondml.tf method), 20	low.layers.Fili	terLayer.Fi	lterLaye
turn_on() (beyondml.p method), 5	ot.layers.Filter	Layer.Filte	rLayer
turn_on() (beyondml.tflo method), 20	ow.layers.Filte	rLayer.Filt	erLayer