Module: tf.contrib.layers

Contents
Modules
Classes
Functions
Other Members

Defined in `tensorflow/contrib/layers/__init__.py`.

Ops for building neural network layers, regularizers, summaries, etc.

See the Layers (contrib) guide.

Modules

`feature_column` module: This API defines FeatureColumn abstraction.

`summaries` module: Utility functions for summary creation.

Classes

`class GDN`: Generalized divisive normalization layer.

Functions

`apply_regularization(...)`: Returns the summed penalty by applying `regularizer` to the `weights_list`.

`avg_pool2d(...)`: Adds a 2D average pooling op.

`avg_pool3d(...)`: Adds a 3D average pooling op.

`batch_norm(...)`: Adds a Batch Normalization layer from http://arxiv.org/abs/1502.03167.

`bias_add(...)`: Adds a bias to the inputs.

`bow_encoder(...)`: Maps a sequence of symbols to a vector per example by averaging embeddings.

`bucketized_column(...)`: Creates a _BucketizedColumn for discretizing dense input.

`check_feature_columns(...)`: Checks the validity of the set of FeatureColumns.

`conv2d(...)`: Adds an N-D convolution followed by an optional batch_norm layer.

`conv2d_in_plane(...)`: Performs the same in-plane convolution to each channel independently.

`conv2d_transpose(...)`: Adds a convolution2d_transpose with an optional batch normalization layer.

`conv3d(...)`: Adds an N-D convolution followed by an optional batch_norm layer.

`conv3d_transpose(...)`: Adds a convolution3d_transpose with an optional batch normalization layer.

`convolution2d(...)` : Adds an N-D convolution followed by an optional batch_norm layer.

`convolution2d_in_plane(...)` : Performs the same in-plane convolution to each channel independently.

`convolution2d_transpose(...)` : Adds a convolution2d_transpose with an optional batch normalization layer.

`convolution3d(...)` : Adds an N-D convolution followed by an optional batch_norm layer.

`convolution3d_transpose(...)` : Adds a convolution3d_transpose with an optional batch normalization layer.

`create_feature_spec_for_parsing(...)` : Helper that prepares features config from input feature_columns.

`crossed_column(...)` : Creates a _CrossedColumn for performing feature crosses.

`dropout(...)` : Returns a dropout op applied to the input.

`embed_sequence(...)` : Maps a sequence of symbols to a sequence of embeddings.

`embedding_column(...)` : Creates an `_EmbeddingColumn` for feeding sparse data into a DNN.

`embedding_lookup_unique(...)` : Version of embedding_lookup that avoids duplicate lookups.

`flatten(...)` : Flattens the input while maintaining the batch_size.

`fully_connected(...)` : Adds a fully connected layer.

`gdn(...)` : Functional interface for GDN layer.

`infer_real_valued_columns(...)`

`input_from_feature_columns(...)` : A tf.contrib.layers style input layer builder based on FeatureColumns.

`instance_norm(...)` : Functional interface for the instance normalization layer.

`joint_weighted_sum_from_feature_columns(...)` : A restricted linear prediction builder based on FeatureColumns.

`l1_l2_regularizer(...)` : Returns a function that can be used to apply L1 L2 regularizations.

`l1_regularizer(...)` : Returns a function that can be used to apply L1 regularization to weights.

`l2_regularizer(...)` : Returns a function that can be used to apply L2 regularization to weights.

`layer_norm(...)` : Adds a Layer Normalization layer.

`legacy_fully_connected(...)` : Adds the parameters for a fully connected layer and returns the output.

`make_place_holder_tensors_for_base_features(...)` : Returns placeholder tensors for inference.

`max_pool2d(...)` : Adds a 2D Max Pooling op.

`max_pool3d(...)` : Adds a 3D Max Pooling op.

`maxout(...)` : Adds a maxout op from https://arxiv.org/abs/1302.4389

`multi_class_target(...)` : Creates a _TargetColumn for multi class single label classification. (deprecated)

`one_hot_column(...)` : Creates an `_OneHotColumn` for a one-hot or multi-hot repr in a DNN.

`one_hot_encoding(...)` : Transform numeric labels into onehot_labels using `tf.one_hot`.

`optimize_loss(...)` : Given loss and parameters for optimizer, returns a training op.

`parse_feature_columns_from_examples(...)` : Parses tf.Examples to extract tensors for given feature_columns.

`parse_feature_columns_from_sequence_examples(...)` : Parses tf.SequenceExamples to extract tensors for given

`FeatureColumn` s.

`real_valued_column(...)` : Creates a `_RealValuedColumn` for dense numeric data.

`regression_target(...)` : Creates a _TargetColumn for linear regression. (deprecated)

`repeat(...)` : Applies the same layer with the same arguments repeatedly.

`safe_embedding_lookup_sparse(...)` : Lookup embedding results, accounting for invalid IDs and empty features.

`scattered_embedding_column(...)` : Creates an embedding column of a sparse feature using parameter hashing.

`separable_conv2d(...)` : Adds a depth-separable 2D convolution with optional batch_norm layer.

`separable_convolution2d(...)` : Adds a depth-separable 2D convolution with optional batch_norm layer.

`sequence_input_from_feature_columns(...)` : Builds inputs for sequence models from `FeatureColumn` s. (experimental)

`shared_embedding_columns(...)` : Creates a list of `_EmbeddingColumn` sharing the same embedding.

`softmax(...)` : Performs softmax on Nth dimension of N-dimensional logit tensor.

`sparse_column_with_hash_bucket(...)` : Creates a _SparseColumn with hashed bucket configuration.

`sparse_column_with_integerized_feature(...)` : Creates an integerized _SparseColumn.

`sparse_column_with_keys(...)` : Creates a _SparseColumn with keys.

`sparse_column_with_vocabulary_file(...)` : Creates a _SparseColumn with vocabulary file configuration.

`stack(...)` : Builds a stack of layers by applying layer repeatedly using stack_args.

`sum_regularizer(...)` : Returns a function that applies the sum of multiple regularizers.

`summarize_activation(...)` : Summarize an activation.

`summarize_activations(...)` : Summarize activations, using `summarize_activation` to summarize.

`summarize_collection(...)` : Summarize a graph collection of tensors, possibly filtered by name.

`summarize_tensor(...)` : Summarize a tensor using a suitable summary type.

`summarize_tensors(...)` : Summarize a set of tensors.

`transform_features(...)` : Returns transformed features based on features columns passed in.

`unit_norm(...)` : Normalizes the given input across the specified dimension to unit length.

`variance_scaling_initializer(...)` : Returns an initializer that generates tensors without scaling variance.

`weighted_sparse_column(...)` : Creates a _SparseColumn by combining sparse_id_column with a weight column.

`weighted_sum_from_feature_columns(...)` : A tf.contrib.layers style linear prediction builder based on FeatureColumn.

`xavier_initializer(...)` : Returns an initializer performing "Xavier" initialization for weights.

`xavier_initializer_conv2d(...)` : Returns an initializer performing "Xavier" initialization for weights.

Other Members

`OPTIMIZER_CLS_NAMES`

`OPTIMIZER_SUMMARIES`

`SPARSE_FEATURE_CROSS_DEFAULT_HASH_KEY`

`elu`

`legacy_linear`

`legacy_relu`

`linear`

`relu`

`relu6`

`scale_gradient`

Last updated November 2, 2017.