tf.contrib.layers.optimize_loss

```
optimize_loss(
    loss,
    global_step,
    learning_rate,
    optimizer,
    gradient_noise_scale=None,
    gradient_multipliers=None,
    clip_gradients=None,
    learning_rate_decay_fn=None,
    update_ops=None,
    variables=None,
    name=None,
    summaries=None,
    colocate_gradients_with_ops=False,
    increment_global_step=True
)
```

Defined in `tensorflow/contrib/layers/python/layers/optimizers.py`.

See the guide: Layers (contrib) > Optimization

Given loss and parameters for optimizer, returns a training op.

Various ways of passing optimizers include:

- by string specifying the name of the optimizer. See OPTIMIZER_CLS_NAMES for full list. E.g. `optimize_loss(..., optimizer='Adam')`.
- by function taking learning rate `Tensor` as argument and returning an `Optimizer` instance. E.g. `optimize_loss(..., optimizer=lambda lr: tf.train.MomentumOptimizer(lr, momentum=0.5))`. Alternatively, if `learning_rate` is `None`, the function takes no arguments. E.g. `optimize_loss(..., learning_rate=None, optimizer=lambda: tf.train.MomentumOptimizer(0.5, momentum=0.5))`.
- by a subclass of `Optimizer` having a single-argument constructor (the argument is the learning rate), such as AdamOptimizer or AdagradOptimizer. E.g. `optimize_loss(..., optimizer=tf.train.AdagradOptimizer)`.
- by an instance of a subclass of `Optimizer`. E.g., `optimize_loss(..., optimizer=tf.train.AdagradOptimizer(0.5))`.

Args:

- `loss`: Scalar `Tensor`.
- `global_step`: Scalar int `Tensor`, step counter to update on each step unless `increment_global_step` is `False`. If not supplied, it will be fetched from the default graph (see `tf.train.get_global_step` for details). If it has not been created, no step will be incremented with each weight update. `learning_rate_decay_fn` requires `global_step`.
- `learning_rate`: float or `Tensor`, magnitude of update per each training step. Can be `None`.
- `optimizer`: string, class or optimizer instance, used as trainer. string should be name of optimizer, like 'SGD', 'Adam', 'Adagrad'. Full list in OPTIMIZER_CLS_NAMES constant. class should be sub-class of `tf.Optimizer` that implements `compute_gradients` and `apply_gradients` functions. optimizer instance should be instantiation of `tf.Optimizer` sub-class and have `compute_gradients` and `apply_gradients` functions.

- `gradient_noise_scale` : float or None, adds 0-mean normal noise scaled by this value.
- `gradient_multipliers` : dict of variables or variable names to floats. If present, gradients for specified variables will be multiplied by given constant.
- `clip_gradients` : float, callable or `None` . If float, is provided, a global clipping is applied to prevent the norm of the gradient to exceed this value. Alternatively, a callable can be provided e.g.: adaptive_clipping. This callable takes a `list` of `(gradients, variables)` `tuple` s and returns the same thing with the gradients modified.
- `learning_rate_decay_fn` : function, takes `learning_rate` and `global_step` `Tensor` s, returns `Tensor` . Can be used to implement any learning rate decay functions. For example: `tf.train.exponential_decay` . Ignored if `learning_rate` is not supplied.
- `update_ops` : list of update `Operation` s to execute at each step. If `None` , uses elements of UPDATE_OPS collection. The order of execution between `update_ops` and `loss` is non-deterministic.
- `variables` : list of variables to optimize or `None` to use all trainable variables.
- `name` : The name for this operation is used to scope operations and summaries.
- `summaries` : List of internal quantities to visualize on tensorboard. If not set, the loss, the learning rate, and the global norm of the gradients will be reported. The complete list of possible values is in OPTIMIZER_SUMMARIES.
- `colocate_gradients_with_ops` : If True, try colocating gradients with the corresponding op.
- `increment_global_step` : Whether to increment `global_step` . If your model calls `optimize_loss` multiple times per training step (e.g. to optimize different parts of the model), use this arg to avoid incrementing `global_step` more times than necessary.

Returns:

Training op.

Raises:

- `ValueError` : if:
    - `loss` is an invalid type or shape.
    - `global_step` is an invalid type or shape.
    - `learning_rate` is an invalid type or value.
    - `optimizer` has the wrong type.
    - `clip_gradients` is neither float nor callable.
    - `learning_rate` and `learning_rate_decay_fn` are supplied, but no `global_step` is available.
    - `gradients` is empty.

Last updated November 2, 2017.