tf.contrib.tpu.CrossShardOptimizer

Class `CrossShardOptimizer`

Inherits From: `Optimizer`

Defined in `tensorflow/contrib/tpu/python/tpu/tpu_optimizer.py`.

An optimizer that averages gradients across TPU shards.

Methods

`__init__`

```
__init__(
    opt,
    reduction=losses.Reduction.MEAN,
    name='CrossShardOptimizer'
)
```

Construct a new cross-shard optimizer.

Args:

- `opt`: An existing `Optimizer` to encapsulate.
- `reduction`: The reduction to apply to the shard losses.
- `name`: Optional name prefix for the operations created when applying gradients. Defaults to "CrossShardOptimizer".

Raises:

- `ValueError`: If reduction is not a valid cross-shard reduction.

`apply_gradients`

```
apply_gradients(
    grads_and_vars,
    global_step=None,
    name=None
)
```

Apply gradients to variables.

Calls tpu_ops.cross_replica_sum() to sum gradient contributions across replicas, and then applies the real optimizer.

Args:

- `grads_and_vars`: List of (gradient, variable) pairs as returned by compute_gradients().
- `global_step`: Optional Variable to increment by one after the variables have been updated.
- `name`: Optional name for the returned operation. Default to the name passed to the Optimizer constructor.

Returns:

An **`Operation`** that applies the gradients. If **`global_step`** was not None, that operation also increments **`global_step`**.

Raises:

- `ValueError`: If the grads_and_vars is malformed.

`compute_gradients`

```
compute_gradients(
    loss,
    var_list=None,
    **kwargs
)
```

Compute gradients of "loss" for the variables in "var_list".

This simply wraps the compute_gradients() from the real optimizer. The gradients will be aggregated in the apply_gradients() so that user can modify the gradients like clipping with per replica global norm if needed. The global norm with aggregated gradients can be bad as one replica's huge gradients can hurt the gradients from other replicas.

Args:

- `loss`: A Tensor containing the value to minimize.
- `var_list`: Optional list or tuple of **`tf.Variable`** to update to minimize **`loss`**. Defaults to the list of variables collected in the graph under the key **`GraphKey.TRAINABLE_VARIABLES`**.
- `**kwargs`: Keyword arguments for compute_gradients().

Returns:

A list of (gradient, variable) pairs.

Raises:

- `ValueError`: If not within a tpu_shard_context.

`get_name`

```
get_name()
```

get_slot

```
get_slot(
    *args,
    **kwargs
)
```

Return a slot named "name" created for "var" by the Optimizer.

This simply wraps the get_slot() from the actual optimizer.

Args:

- `*args`: Arguments for get_slot().
- `**kwargs`: Keyword arguments for get_slot().

Returns:

The **`Variable`** for the slot if it was created, **`None`** otherwise.

get_slot_names

```
get_slot_names(
    *args,
    **kwargs
)
```

Return a list of the names of slots created by the **`Optimizer`**.

This simply wraps the get_slot_names() from the actual optimizer.

Args:

- `*args`: Arguments for get_slot().
- `**kwargs`: Keyword arguments for get_slot().

Returns:

A list of strings.

minimize

```
minimize(
    loss,
    global_step=None,
    var_list=None,
    gate_gradients=GATE_OP,
    aggregation_method=None,
    colocate_gradients_with_ops=False,
    name=None,
    grad_loss=None
)
```

Add operations to minimize **`loss`** by updating **`var_list`**.

This method simply combines calls `compute_gradients()` and `apply_gradients()`. If you want to process the gradient before applying them call `compute_gradients()` and `apply_gradients()` explicitly instead of using this function.

Args:

- `loss`: A `Tensor` containing the value to minimize.
- `global_step`: Optional `Variable` to increment by one after the variables have been updated.
- `var_list`: Optional list or tuple of `Variable` objects to update to minimize `loss`. Defaults to the list of variables collected in the graph under the key `GraphKeys.TRAINABLE_VARIABLES`.
- `gate_gradients`: How to gate the computation of gradients. Can be `GATE_NONE`, `GATE_OP`, or `GATE_GRAPH`.
- `aggregation_method`: Specifies the method used to combine gradient terms. Valid values are defined in the class `AggregationMethod`.
- `colocate_gradients_with_ops`: If True, try colocating gradients with the corresponding op.
- `name`: Optional name for the returned operation.
- `grad_loss`: Optional. A `Tensor` holding the gradient computed for `loss`.

Returns:

An Operation that updates the variables in `var_list`. If `global_step` was not `None`, that operation also increments `global_step`.

Raises:

- `ValueError`: If some of the variables are not `Variable` objects.

Class Members

GATE_GRAPH

GATE_NONE

GATE_OP