tf.contrib.layers.layer_norm

```
layer_norm(
    inputs,
    center=True,
    scale=True,
    activation_fn=None,
    reuse=None,
    variables_collections=None,
    outputs_collections=None,
    trainable=True,
    begin_norm_axis=1,
    begin_params_axis=-1,
    scope=None
)
```

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

See the guide: Layers (contrib) > Higher level ops for building neural network layers

Adds a Layer Normalization layer.

Based on the paper:

"Layer Normalization"

Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton

https://arxiv.org/abs/1607.06450.

Can be used as a normalizer function for conv2d and fully_connected.

Given a tensor `inputs` of rank `R`, moments are calculated and normalization is performed over axes `begin_norm_axis ... R - 1`. Scaling and centering, if requested, is performed over axes `begin_shift_axis .. R - 1`.

By default, `begin_norm_axis = 1` and `begin_params_axis = -1`, meaning that normalization is performed over all but the first axis (the `HWC` if `inputs` is `NHWC`), while the `beta` and `gamma` trainable parameters are calculated for the rightmost axis (the `C` if `inputs` is `NHWC`). Scaling and recentering is performed via broadcast of the `beta` and `gamma` parameters with the normalized tensor.

The shapes of `beta` and `gamma` are `inputs.shape[begin_params_axis:]`, and this part of the inputs' shape must be fully defined.

Args:

- `inputs`: A tensor having rank `R`. The normalization is performed over axes `begin_norm_axis ... R - 1` and centering and scaling parameters are calculated over `begin_params_axis ... R - 1`.
- `center`: If True, add offset of `beta` to normalized tensor. If False, `beta` is ignored.
- `scale`: If True, multiply by `gamma`. If False, `gamma` is not used. When the next layer is linear (also e.g. `nn.relu`), this can be disabled since the scaling can be done by the next layer.
- `activation_fn`: Activation function, default set to None to skip it and maintain a linear activation.
- `reuse`: Whether or not the layer and its variables should be reused. To be able to reuse the layer scope must be

given.

- `variables_collections`: Optional collections for the variables.
- `outputs_collections`: Collections to add the outputs.
- `trainable`: If `True` also add variables to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see tf.Variable).
- `begin_norm_axis`: The first normalization dimension: normalization will be performed along dimensions `begin_norm_axis : rank(inputs)`
- `begin_params_axis`: The first parameter (beta, gamma) dimension: scale and centering parameters will have dimensions `begin_params_axis : rank(inputs)` and will be broadcast with the normalized inputs accordingly.
- `scope`: Optional scope for `variable_scope`.

Returns:

A `Tensor` representing the output of the operation, having the same shape and dtype as `inputs`.

Raises:

- `ValueError`: If the rank of `inputs` is not known at graph build time, or if `inputs.shape[begin_params_axis:]` is not fully defined at graph build time.
