tf.contrib.layers.legacy_fully_connected

```
legacy_fully_connected(
    x,
    num_output_units,
    activation_fn=None,
    weight_init=initializers.xavier_initializer(),
    bias_init=tf.zeros_initializer(),
    name=None,
    weight_collections=(ops.GraphKeys.WEIGHTS,),
    bias_collections=(ops.GraphKeys.BIASES,),
    output_collections=(ops.GraphKeys.ACTIVATIONS,),
    trainable=True,
    weight_regularizer=None,
    bias_regularizer=None
)
```

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

Adds the parameters for a fully connected layer and returns the output.

A fully connected layer is generally defined as a matrix multiply: `y = f(w * x + b)` where `f` is given by `activation_fn`. If `activation_fn` is `None`, the result of `y = w * x + b` is returned.

If `x` has shape [\(\text{dim}*0, \text{dim}_1, ..., \text{dim}_n\)] with more than 2 dimensions (n > 1), then we repeat the matrix multiply along the first dimensions. The result r is a tensor of shape [\(\text{dim}_0, ..., \text{dim}*{n-1},\) `num_output_units`], where \(r_{i_0, ..., i_{n-1}, k} = \sum_{0 \leq j < \text{dim}*n} x{i_0, ... i_{n-1}, j} \cdot w_{j, k}\). This is accomplished by reshaping `x` to 2-D [\(\text{dim}*0 \cdot ... \cdot \text{dim}*{n-1}, \text{dim}*n\)] before the matrix multiply and afterwards reshaping it to [\(\text{dim}_0, ..., \text{dim}*{n-1},\) `num_output_units`].

This op creates `w` and optionally `b`. Bias (`b`) can be disabled by setting `bias_init` to `None`.

The variable creation is compatible with `tf.variable_scope` and so can be reused with `tf.variable_scope` or `tf.make_template`.

Most of the details of variable creation can be controlled by specifying the initializers (`weight_init` and `bias_init`) and in which collections to place the created variables (`weight_collections` and `bias_collections`; note that the variables are always added to the `VARIABLES` collection). The output of the layer can be placed in custom collections using `output_collections`. The collections arguments default to `WEIGHTS`, `BIASES` and `ACTIVATIONS`, respectively.

A per layer regularization can be specified by setting `weight_regularizer` and `bias_regularizer`, which are applied to the weights and biases respectively, and whose output is added to the `REGULARIZATION_LOSSES` collection.

Args:

- `x`: The input `Tensor`.
- `num_output_units`: The size of the output.
- `activation_fn`: Activation function, default set to None to skip it and maintain a linear activation.
- `weight_init`: An optional weight initialization, defaults to `xavier_initializer`.
- `bias_init`: An initializer for the bias, defaults to 0. Set to `None` in order to disable bias.
- `name`: The name for this operation is used to name operations and to find variables. If specified it must be unique for

this scope, otherwise a unique name starting with "fully_connected" will be created. See `tf.variable_scope` for details.

- `weight_collections`: List of graph collections to which weights are added.
- `bias_collections`: List of graph collections to which biases are added.
- `output_collections`: List of graph collections to which outputs are added.
- `trainable`: If `True` also add variables to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see tf.Variable).
- `weight_regularizer`: A regularizer like the result of `l1_regularizer` or `l2_regularizer`. Used for weights.
- `bias_regularizer`: A regularizer like the result of `l1_regularizer` or `l2_regularizer`. Used for biases.

Returns:

The output of the fully connected layer.

Raises:

- `ValueError`: If x has rank less than 2 or if its last dimension is not set.
