tf.layers.separable_conv2d

```
separable_conv2d(
    inputs,
    filters,
    kernel_size,
    strides=(1, 1),
    padding='valid',
    data_format='channels_last',
    dilation_rate=(1, 1),
    depth_multiplier=1,
    activation=None,
    use_bias=True,
    depthwise_initializer=None,
    pointwise_initializer=None,
    bias_initializer=tf.zeros_initializer(),
    depthwise_regularizer=None,
    pointwise_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    depthwise_constraint=None,
    pointwise_constraint=None,
    bias_constraint=None,
    trainable=True,
    name=None,
    reuse=None
)
```

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

Functional interface for the depthwise separable 2D convolution layer.

This layer performs a depthwise convolution that acts separately on channels, followed by a pointwise convolution that mixes channels. If `use_bias` is True and a bias initializer is provided, it adds a bias vector to the output. It then optionally applies an activation function to produce the final output.

Arguments:

- `inputs`: Input tensor.
- `filters`: Integer, the dimensionality of the output space (i.e. the number of filters in the convolution).
- `kernel_size`: A tuple or list of 2 integers specifying the spatial dimensions of the filters. Can be a single integer to specify the same value for all spatial dimensions.
- `strides`: A tuple or list of 2 positive integers specifying the strides of the convolution. Can be a single integer to specify the same value for all spatial dimensions. Specifying any `stride` value != 1 is incompatible with specifying any `dilation_rate` value != 1.
- `padding`: One of `"valid"` or `"same"` (case-insensitive).
- `data_format`: A string, one of `channels_last` (default) or `channels_first`. The ordering of the dimensions in the inputs. `channels_last` corresponds to inputs with shape `(batch, height, width, channels)` while `channels_first` corresponds to inputs with shape `(batch, channels, height, width)`.
- `dilation_rate`: An integer or tuple/list of 2 integers, specifying the dilation rate to use for dilated convolution. Can be a single integer to specify the same value for all spatial dimensions. Currently, specifying any `dilation_rate`

value != 1 is incompatible with specifying any stride value != 1.

- `depth_multiplier`: The number of depthwise convolution output channels for each input channel. The total number of depthwise convolution output channels will be equal to `num_filters_in * depth_multiplier`.
- `activation`: Activation function. Set it to None to maintain a linear activation.
- `use_bias`: Boolean, whether the layer uses a bias.
- `depthwise_initializer`: An initializer for the depthwise convolution kernel.
- `pointwise_initializer`: An initializer for the pointwise convolution kernel.
- `bias_initializer`: An initializer for the bias vector. If None, no bias will be applied.
- `depthwise_regularizer`: Optional regularizer for the depthwise convolution kernel.
- `pointwise_regularizer`: Optional regularizer for the pointwise convolution kernel.
- `bias_regularizer`: Optional regularizer for the bias vector.
- `activity_regularizer`: Optional regularizer function for the output.
- `depthwise_constraint`: Optional projection function to be applied to the depthwise kernel after being updated by an `Optimizer` (e.g. used for norm constraints or value constraints for layer weights). The function must take as input the unprojected variable and must return the projected variable (which must have the same shape). Constraints are not safe to use when doing asynchronous distributed training.
- `pointwise_constraint`: Optional projection function to be applied to the pointwise kernel after being updated by an `Optimizer`.
- `bias_constraint`: Optional projection function to be applied to the bias after being updated by an `Optimizer`.
- `trainable`: Boolean, if `True` also add variables to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see `tf.Variable`).
- `name`: A string, the name of the layer.
- `reuse`: Boolean, whether to reuse the weights of a previous layer by the same name.

Returns:

Output tensor.