tf.contrib.layers.conv2d_in_plane

Aliases:

- `tf.contrib.layers.conv2d_in_plane`
- `tf.contrib.layers.convolution2d_in_plane`

```
conv2d_in_plane(
    inputs,
    kernel_size,
    stride=1,
    padding='SAME',
    activation_fn=tf.nn.relu,
    normalizer_fn=None,
    normalizer_params=None,
    weights_initializer=initializers.xavier_initializer(),
    weights_regularizer=None,
    biases_initializer=tf.zeros_initializer(),
    biases_regularizer=None,
    reuse=None,
    variables_collections=None,
    outputs_collections=None,
    trainable=True,
    scope=None
)
```

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

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

Performs the same in-plane convolution to each channel independently.

This is useful for performing various simple channel-independent convolution operations such as image gradients:

image = tf.constant(..., shape=(16, 240, 320, 3)) vert_gradients = layers.conv2d_in_plane(image, kernel=[1, -1], kernel_size=[2, 1]) horz_gradients = layers.conv2d_in_plane(image, kernel=[1, -1], kernel_size=[1, 2])

Args:

- `inputs`: A 4-D tensor with dimensions [batch_size, height, width, channels].
- `kernel_size`: A list of length 2 holding the [kernel_height, kernel_width] of of the pooling. Can be an int if both values are the same.
- `stride`: A list of length 2 `[stride_height, stride_width]`. Can be an int if both strides are the same. Note that presently both strides must have the same value.
- `padding`: The padding type to use, either 'SAME' or 'VALID'.
- `activation_fn`: Activation function. The default value is a ReLU function. Explicitly set it to None to skip it and maintain a linear activation.

- `normalizer_fn`: Normalization function to use instead of `biases`. If `normalizer_fn` is provided then `biases_initializer` and `biases_regularizer` are ignored and `biases` are not created nor added. default set to None for no normalizer function
- `normalizer_params`: Normalization function parameters.
- `weights_initializer`: An initializer for the weights.
- `weights_regularizer`: Optional regularizer for the weights.
- `biases_initializer`: An initializer for the biases. If None skip biases.
- `biases_regularizer`: Optional regularizer for the biases.
- `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 list of collections for all the variables or a dictionary containing a different list of collection per variable.
- `outputs_collections`: Collection to add the outputs.
- `trainable`: If `True` also add variables to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see tf.Variable).
- `scope`: Optional scope for `variable_scope`.

Returns:

A `Tensor` representing the output of the operation.