tf.contrib.signal.linear_to_mel_weight_matrix

```
linear_to_mel_weight_matrix(
    num_mel_bins=20,
    num_spectrogram_bins=129,
    sample_rate=8000,
    lower_edge_hertz=125.0,
    upper_edge_hertz=3800.0,
    dtype=tf.float32,
    name=None
)
```

Defined in `tensorflow/contrib/signal/python/ops/mel_ops.py`.

See the guide: Signal Processing (contrib) > Computing log-mel spectrograms

Returns a matrix to warp linear scale spectrograms to the mel scale.

Returns a weight matrix that can be used to re-weight a `Tensor` containing `num_spectrogram_bins` linearly sampled frequency information from `[0, sample_rate / 2]` into `num_mel_bins` frequency information from `[lower_edge_hertz, upper_edge_hertz]` on the mel scale.

For example, the returned matrix `A` can be used to right-multiply a spectrogram `S` of shape `[frames, num_spectrogram_bins]` of linear scale spectrum values (e.g. STFT magnitudes) to generate a "mel spectrogram" `M` of shape `[frames, num_mel_bins]`.

```
# `S` has shape [frames, num_spectrogram_bins]
# `M` has shape [frames, num_mel_bins]
M = tf.matmul(S, A)
```

The matrix can be used with `tf.tensordot` to convert an arbitrary rank `Tensor` of linear-scale spectral bins into the mel scale.

```
# S has shape [..., num_spectrogram_bins].
# M has shape [..., num_mel_bins].
M = tf.tensordot(S, A, 1)
# tf.tensordot does not support shape inference for this case yet.
M.set_shape(S.shape[:-1].concatenate(A.shape[-1:]))
```

Args:

- `num_mel_bins`: Python int. How many bands in the resulting mel spectrum.
- `num_spectrogram_bins`: Python int. How many bins there are in the source spectrogram data, which is understood to be `fft_size // 2 + 1`, i.e. the spectrogram only contains the nonredundant FFT bins.
- `sample_rate`: Python float. Samples per second of the input signal used to create the spectrogram. We need this to figure out the actual frequencies for each spectrogram bin, which dictates how they are mapped into the mel scale.
- `lower_edge_hertz`: Python float. Lower bound on the frequencies to be included in the mel spectrum. This corresponds to the lower edge of the lowest triangular band.
- `upper_edge_hertz`: Python float. The desired top edge of the highest frequency band.
- `dtype`: The `DType` of the result matrix. Must be a floating point type.

- `name`: An optional name for the operation.

Returns:

A `Tensor` of shape `[num_spectrogram_bins, num_mel_bins]`.

Raises:

- `ValueError`: If num_mel_bins/num_spectrogram_bins/sample_rate are not positive, lower_edge_hertz is negative, or frequency edges are incorrectly ordered.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License. For details, see our Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.