TimeDistributed [source]

```
keras.layers.wrappers.TimeDistributed(layer)
```

This wrapper allows to apply a layer to every temporal slice of an input.

The input should be at least 3D, and the dimension of index one will be considered to be the temporal dimension.

Consider a batch of 32 samples, where each sample is a sequence of 10 vectors of 16 dimensions. The batch input shape of the layer is then `(32, 10, 16)` (and the `input_shape`, not including the samples dimension, is `(10, 16)`).

You can then use `TimeDistributed` to apply a `Dense` layer to each of the 10 timesteps, independently:

```
# as the first layer in a model
model = Sequential()
model.add(TimeDistributed(Dense(8), input_shape=(10, 16)))
# now model.output_shape == (None, 10, 8)

# subsequent layers: no need for input_shape
model.add(TimeDistributed(Dense(32)))
# now model.output_shape == (None, 10, 32)
```

The output will then have shape `(32, 10, 8)`.

Note this is strictly equivalent to using `layers.core.TimeDistributedDense`. However what is different about `TimeDistributed` is that it can be used with arbitrary layers, not just `Dense`, for instance with a `Convolution2D` layer:

```
model = Sequential()
model.add(TimeDistributed(Convolution2D(64, 3, 3), input_shape=(10, 3, 299, 299)))
```

Arguments

- **layer**: a layer instance.