```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 10, 300, 64)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 10, 300, 16)       9232
_________________________________________________________________
batch_normalization (BatchNo (None, 10, 300, 16)       64
_________________________________________________________________
activation (Activation)      (None, 10, 300, 16)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 2, 75, 16)         0
_________________________________________________________________
dropout (Dropout)            (None, 2, 75, 16)         0
_________________________________________________________________
permute (Permute)            (None, 75, 2, 16)         0
_________________________________________________________________
reshape_1 (Reshape)          (None, 60, 40)            0
_________________________________________________________________
time_distributed (TimeDistri (None, 60, 128)           5248
_________________________________________________________________
time_distributed_1 (TimeDist (None, 60, 7)             903
=================================================================
Total params: 15,447
Trainable params: 15,415
Non-trainable params: 32
_________________________________________________________________
WARNING:tensorflow:Layer regular_transfer_nn is casting an input tensor from dtype float64 to the layer's dtype of float32, which is new behavior in TensorFlow 2.  The layer has dtype float32 because its dtype defaults to floatx.

If you intended to run this layer in float32, you can safely ignore this warning. If in doubt, this warning is likely only an issue if you are porting a TensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backend.set_floatx('float64')`. To change just this layer, pass dtype='float64' to the layer constructor. If you are the author of this layer, you can disable autocasting by passing autocast=False to the base Layer constructor.

Computing src dataset size...
Done!
Computing tgt dataset size...
```

```
Done!
Epoch 1/3

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-be1b144e3a4e> in <module>
     33                                    lambdas=0.,
     34                                    random_state=19)
---> 35 reg.fit(source_data, batch_size=10, epochs=3)

/usr/local/lib/python3.6/dist-packages/adapt/parameter_based/_regular.py in
fit(self, Xt, yt, **fit_params)
    397         Xs = Xt
    398         ys = yt
--> 399         return super().fit(Xs, ys, Xt=Xt, yt=yt, **fit_params)
    400
    401

/usr/local/lib/python3.6/dist-packages/adapt/base.py in fit(self, X, y, Xt, yt,
domains, **fit_params)
   1144         self.pretrain_ = False
   1145
-> 1146         hist = super().fit(dataset, validation_data=validation_data,
**fit_params)
   1147
   1148         for k, v in hist.history.items():

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/
training.py in _method_wrapper(self, *args, **kwargs)
    106   def _method_wrapper(self, *args, **kwargs):
    107     if not self._in_multi_worker_mode():  # pylint: disable=protected-
access
--> 108       return method(self, *args, **kwargs)
    109
    110     # Running inside `run_distribute_coordinator` already.

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/
training.py in fit(self, x, y, batch_size, epochs, verbose, callbacks,
validation_split, validation_data, shuffle, class_weight, sample_weight,
initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,
validation_freq, max_queue_size, workers, use_multiprocessing)
   1096                 batch_size=batch_size):
   1097               callbacks.on_train_batch_begin(step)
-> 1098               tmp_logs = train_function(iterator)
   1099               if data_handler.should_sync:
   1100                 context.async_wait()
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py
in __call__(self, *args, **kwds)
    778       else:
    779         compiler = "nonXla"
--> 780         result = self._call(*args, **kwds)
    781
    782       new_tracing_count = self._get_tracing_count()

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py
in _call(self, *args, **kwds)
    821       # This is the first call of __call__, so we have to initialize.
    822       initializers = []
--> 823       self._initialize(args, kwds, add_initializers_to=initializers)
    824     finally:
    825       # At this point we know that the initialization is complete (or less

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py
in _initialize(self, args, kwds, add_initializers_to)
    695     self._concrete_stateful_fn = (
    696
self._stateful_fn._get_concrete_function_internal_garbage_collected(  # pylint:
disable=protected-access
--> 697             *args, **kwds))
    698
    699     def invalid_creator_scope(*unused_args, **unused_kwds):

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in
_get_concrete_function_internal_garbage_collected(self, *args, **kwargs)
   2853       args, kwargs = None, None
   2854     with self._lock:
-> 2855       graph_function, _, _ = self._maybe_define_function(args, kwargs)
   2856     return graph_function
   2857

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in
_maybe_define_function(self, args, kwargs)
   3211
   3212       self._function_cache.missed.add(call_context_key)
-> 3213       graph_function = self._create_graph_function(args, kwargs)
   3214       self._function_cache.primary[cache_key] = graph_function
   3215       return graph_function, args, kwargs

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/function.py in
_create_graph_function(self, args, kwargs, override_flat_arg_shapes)
   3073             arg_names=arg_names,
   3074             override_flat_arg_shapes=override_flat_arg_shapes,
```

```
-> 3075           capture_by_value=self._capture_by_value),
   3076       self._function_attributes,
   3077       function_spec=self.function_spec,

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/
func_graph.py in func_graph_from_py_func(name, python_func, args, kwargs,
signature, func_graph, autograph, autograph_options,
add_control_dependencies, arg_names, op_return_value, collections,
capture_by_value, override_flat_arg_shapes)
    984       _, original_func = tf_decorator.unwrap(python_func)
    985
--> 986       func_outputs = python_func(*func_args, **func_kwargs)
    987
    988       # invariant: `func_outputs` contains only Tensors,
CompositeTensors,

/usr/local/lib/python3.6/dist-packages/tensorflow/python/eager/def_function.py
in wrapped_fn(*args, **kwds)
    598       # __wrapped__ allows AutoGraph to swap in a converted function.
We give
    599       # the function a weak reference to itself to avoid a reference cycle.
--> 600       return weak_wrapped_fn().__wrapped__(*args, **kwds)
    601     weak_wrapped_fn = weakref.ref(wrapped_fn)
    602

/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/
func_graph.py in wrapper(*args, **kwargs)
    971           except Exception as e:  # pylint:disable=broad-except
    972             if hasattr(e, "ag_error_metadata"):
--> 973               raise e.ag_error_metadata.to_exception(e)
    974             else:
    975               raise

TypeError: in user code:

    /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/
training.py:806 train_function  *
        return step_function(self, iterator)
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/
training.py:796 step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/distribute/
distribute_lib.py:1211 run
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/distribute/
distribute_lib.py:2585 call_for_each_replica
        return self._call_for_each_replica(fn, args, kwargs)
```

```
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/distribute/
distribute_lib.py:2945 _call_for_each_replica
        return fn(*args, **kwargs)
    /usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/
training.py:789 run_step  **
        outputs = model.train_step(data)
    /usr/local/lib/python3.6/dist-packages/adapt/base.py:1351 train_step
        self.optimizer.minimize(loss, self.trainable_variables, tape=tape)

    TypeError: minimize() got an unexpected keyword argument 'tape'
```