```
In [1]: import sys
        sys.path.append('../deepsurv')
        import deep_surv

        from deepsurv_logger import DeepSurvLogger, TensorboardLogger
        import utils
        import viz

        import numpy as np
        import pandas as pd

        import lasagne
        import matplotlib
        import matplotlib.pyplot as plt
        %matplotlib inline
```

```
/Users/timfingerhut/anaconda3/lib/python3.6/site-packages/h5py/__i
nit__.py:36: FutureWarning: Conversion of the second argument of i
ssubdtype from `float` to `np.floating` is deprecated. In future,
it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

```
In [2]: train_dataset_fp = './2006_final.csv'
        train_df = pd.read_csv(train_dataset_fp)
        train_df.head()
```

Out[2]:

	fail_1	dur	country1	country2	country4	country5	country6	country7	country8	count
0	1	30	1	0	0	0	0	0	0	
1	1	1	1	0	0	0	0	0	0	
2	1	60	1	0	0	0	0	0	0	
3	1	60	1	0	0	0	0	0	0	
4	1	45	1	0	0	0	0	0	0	

5 rows × 67 columns

```python
In [3]:  # event_col is the header in the df that represents the 'Event / St
         atus' indicator
         # time_col is the header in the df that represents the event time
         def dataframe_to_deepsurv_ds(df, event_col = 'fail_1', time_col = '
         dur'):
             # Extract the event and time columns as numpy arrays
             e = df[event_col].values.astype(np.int32)
             t = df[time_col].values.astype(np.float32)

             # Extract the patient's covariates as a numpy array
             x_df = df.drop([event_col, time_col], axis = 1)
             x = x_df.values.astype(np.float32)

             # Return the deep surv dataframe
             return {
                 'x' : x,
                 'e' : e,
                 't' : t
             }

         # If the headers of the csv change, you can replace the values of
         # 'event_col' and 'time_col' with the names of the new headers
         # You can also use this function on your training dataset, validati
         on dataset, and testing dataset
         train_data = dataframe_to_deepsurv_ds(train_df, event_col = 'fail_1
         ', time_col= 'dur')
```

```python
In [4]:  hyperparams = {
             'L2_reg': 10.0,
             'batch_norm': True,
             'dropout': 0.4,
             'hidden_layers_sizes': [100, 100],
             'learning_rate': 3.1e-2,
             'lr_decay': 0.001,
             'momentum': 0.9,
             'n_in': train_data['x'].shape[1],
             'standardize': True
         }
```

```python
In [5]:  # Create an instance of DeepSurv using the hyperparams defined abov
         e
         model = deep_surv.DeepSurv(**hyperparams)

         # DeepSurv can now leverage TensorBoard to monitor training and val
         idation
         # This section of code is optional. If you don't want to use the te
         nsorboard logger
         # Uncomment the below line, and comment out the other three lines:
         # logger = None

         experiment_name = 'test_experiment_tim'
         logdir = './logs/tensorboard/'
         logger = TensorboardLogger(experiment_name, logdir=logdir)

         # Now we train the model
         update_fn=lasagne.updates.amsgrad # The type of optimizer to use. \
                                           # Check out http://lasa
         gne.readthedocs.io/en/latest/modules/updates.html \
                                           # for other optimizers
         to use
         n_epochs = 2000

         # If you have validation data, you can add it as the second paramet
         er to the function
         metrics = model.train(train_data, n_epochs=n_epochs, logger=logger,
         update_fn=update_fn)
```

```
2019-07-30 15:50:49,413 - Training step 0/2000      |
| - loss: 28.9713 - ci: 0.6022
2019-07-30 15:52:01,795 - Training step 250/2000  |***
| - loss: 6.5006 - ci: 0.8121
2019-07-30 15:53:04,677 - Training step 500/2000  |******
| - loss: 6.3358 - ci: 0.8592
2019-07-30 15:54:09,099 - Training step 750/2000  |*********
| - loss: 6.2895 - ci: 0.8801
2019-07-30 15:55:13,949 - Training step 1000/2000 |************
| - loss: 6.2164 - ci: 0.8936
2019-07-30 15:56:17,927 - Training step 1250/2000 |***************
| - loss: 6.1735 - ci: 0.8989
2019-07-30 15:57:17,820 - Training step 1500/2000 |***************
***        | - loss: 6.1856 - ci: 0.9013
2019-07-30 15:58:17,354 - Training step 1750/2000 |***************
******     | - loss: 6.1675 - ci: 0.9061
2019-07-30 15:59:16,552 - Finished Training with 2000 iterations i
n 507.40s
```

```
In [6]:  # Print the final metrics
         print('Train C-Index:', metrics['c-index'][-1])
         # print('Valid C-Index: ',metrics['valid_c-index'][-1])

         # Plot the training / validation curves
         viz.plot_log(metrics)
```

Train C-Index: (1999, 0.907842079277384)