Requirements

```
absl-py==1.3.0
astunparse==1.6.3
cachetools==5.2.0
certifi==2022.9.24
charset-normalizer==2.1.1
filelock==3.8.0
flatbuffers==22.9.24
gast==0.4.0
google-auth==2.12.0
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
grpcio==1.49.1
h5py==3.7.0
huggingface-hub==0.10.1
idna==3.4
keras==2.10.0
Keras-Preprocessing==1.1.2
libclang==14.0.6
Markdown==3.4.1
MarkupSafe==2.1.1
numpy==1.23.4
oauthlib==3.2.1
opt-einsum==3.3.0
packaging==21.3
protobuf==3.19.6
pyasn1==0.4.8
pyasn1-modules==0.2.8
pyparsing==3.0.9
PyYAML==6.0
regex==2022.9.13
requests==2.28.1
requests-oauthlib==1.3.1
rsa==4.9
six==1.16.0
tensorboard==2.10.1
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.10.0
tensorflow-estimator==2.10.0
tensorflow-io-gcs-filesystem==0.27.0
termcolor==2.0.1
tokenizers==0.13.1
tqdm==4.64.1
transformers==4.23.1
typing_extensions==4.4.0
urllib3==1.26.12
Werkzeug==2.2.2
wrapt==1.14.1
```

BERT_Notes

This Google AI Blog summarizes how BERT has been pre-trained using Wikipedia

https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

Text classification can be done by BERT; and also linear SVM,
Multinomial Naive Bayes, logistic regression, and Random Forest
https://www.kaggle.com/selener/multi-class-text-classification-tfidf
https://monkeylearn.com/text-classification/

some recommendations on step-by-step guide in BERT
https://stackoverflow.com/questions/69025750/how-to-fine-tune-huggingface-bert-model-for-text-classification

multi-label, multi-class text classification, but I was blocked unless
I register or sign up with google.
https://towardsdatascience.com/multi-label-multi-class-text-classification-with-bert-transformer-and-keras-c6355eccb63a

neuronet_workflow

```
# Artificial neural networks workflow, published in 2012 in Neural
# Computing and Applications by Al-Bulushi et al.
# https://link.springer.com/article/10.1007/s00521-010-0501-6
#
# Also see https://www.tmwr.org/tuning.html for tuning parameters
# and overfitting. This reminds me that the alpha in BNP is a
# tuning parameter. This post shows that a poor choice of a tuning
# parameter  can cause problems.
#
```

The key operation in the development of an ANN is the learning process, by which the free parameters (weight and bias) are modified through a continuous process of stimulation by the environment in which the network is embedded(Haykin, 1998). The learning process continues until a satisfactory error is reached. One complete presentation of the entire training data to the network during the training process is called an epoch or one training cycle (Haykin, 1998). The overall objective of the MLP learning is to optimize the performance function. Optimizing means finding the minimum of the performance function(Hagan, Demuth, & Beale, 1996).

[....]

Many learning algorithms, such as back-propagation (BP), BP with momentum,resilient propagation (PROP), conjugate gradient and Levenberg-Marquardt (Hagan et al, 1996) are available to train the network.

[....]

The data in the neural network are divided into three main parts: training, validation, and testing subsets [13, 17]. The training data are used to train the network and to adapt its internal structure. The validation data are used during training along with the training data to monitor the performance of the network. They are not used to adapt the network. The testing data are kept aside until the whole training process has been completed. This set is used as a biased to investigate the generalization capability of the trained network on new data. It is basically used to test whether the network captured the general trend and did not memorize (fitted) the noise on the training data(over-training).

There are two main modes for training [13, 20]: sequential (stochastic) and batch modes. In stochastic mode, the free parameter updating is performed after the presentation of each training example. In batch mode, the weight updating is performed after the presentation of all the training examples that constitute an epoch (one training cycle). The performance function is then the average sum of the square of the error (average of the whole training data samples). The sequential mode is much faster since it requires less local storage of connection weight [13, 20]. On the other hand, the batch mode has the advantage that conditions of the convergence are well understood [20].
Furthermore, many advanced learning algorithms such as conjugate gradients operate only in batch mode. Stopping criteria and generalization

The aim of neural network model training is to obtain a low enough error solution for the problem under investigation. The learning network algorithm searches for the global lowest error. The main challenge in neural network modeling is how to set the criteria for network training termination. In other words, how can we stop the network from training before memorization (fitting the noise) takes place, where the lowest error found by the network might not be necessarily the best solution in order to generalize the model [13]. The ability of an ANN to execute well on hidden patterns (testing data subset) is called its ability to generalize [13, 16]. Besides the generalization issue, it is not always certain that the training error converges to a minimum or that it achieves it in a reasonable time. All these issues make the stopping criterion a complex issue in neural network modeling.

Generalization is one of the critical issues in developing an ANN model. It is more significant than the network's ability to map the training patterns correctly (finding the lowest error in the training subset), since the network objective is to solve the unknown case [16, 21]. The generalization is affected by three main factors: (1) the size of the data, (2) network size and (3) the complexity of the problem under investigation [13, 16]. The last factor though is out of our control.

Specifying the network size is an important task. If the network is very small, then its ability to provide a good solution of the problem might be limited. On the other hand, if the network is too big, then the danger of memorizing the data (not being able to generalize) will be high [16, 22]. Hush and Horne [16] pointed out that in general, it is not known what size of network works best for a problem under investigation. Furthermore, it is difficult to specify a network size for a general case since each problem will demand different capabilities. With little or no prior knowledge of the problem, the trial and error method can be used to determine the network size.

There are several approaches that may be used as a trial and error procedure to determine network size. One approach is to start with the smallest possible network and gradually increase the size [16, 23]. The optimal network size is then that point at which the performance begins to level off. After this point, the network will begin to memorize the training data. Another approach is to start with an oversized network and then apply a pruning technique that removes weight/nodes, which end up contributing little or nothing to the solution [20].
In this approach, an idea of what size constitutes a large network needs to be known [16].

About tuning parameters: https://www.tmwr.org/tuning.html

In the classic single-layer artificial neural network (a.k.a. the multilayer perceptron), the predictors are combined using two or more hidden units. The hidden units are linear combinations of the predictors that are captured in an activation function (typically a nonlinear function, such as a sigmoid). The hidden units are then connected to the outcome units; one outcome unit is used for regression models and multiple outcome units are required for classification. The number of hidden units and the type of activation functionare important structural tuning parameters.

Modern gradient descent methods are improved by finding the right optimization parameters. Examples are learning rates, momentum, and the number of optimization iterations/epochs (Goodfellow, Bengio, and Courville 2016). Neural networks and some ensemble models use gradient descent to estimate the model parameters. While the tuning parameters associated with gradient descent are not structural parameters, they often require tuning.

Hagan M, Demuth HB, Beale M (1996) Neural network design. PWS Publishing Company, USA

Haykin S (1998) neural networks, a comprehensive foundation. Prentice Hall PTR,
2nd edn. USA

README file

[10/16/2022]
You might want to create a working directory structure like this:

```
python/
|-- data               # where to store data (if any)
|-- deliver            # final analyses, code, and presentations
|-- develop            # notebooks for explorary analyses
|-- src                # scripts and local project modules
```

Next, we install and setup python programming environment.

1. Create a python virtual environment and name it "transformers_venv".

```
$ /usr/local/bin/python3 -m venv transformers_venv
```

[NOTE: my python3 is under /usr/local/bin, yours may be installed
in a different directory]

Afterwards ./transformers_venv is added to the current direcotry tree.

```
python/
|-- data
|-- deliver
|-- develop
|-- src
|-- transformers_venv  # transformer packages installed here
```

Activate the virtual environment
```
$ source ./transformers_venv/bin/activate
```

You will notice that the command prompt changes to something like:
(transformers_venv) [z python]%

After activating transformers_venv, install packages within it.
[NOTE: Python uses virtual environments to keep things organized.
Search "why use a python virtual environment" for explanations.]

```
(transformers_venv) [z python]% pip install transformers
(transformers_venv) [z python]% pip install tensorflow
(transformers_venv) [z python]% pip install torch
(transformers_venv) [z python]% pip install ktrain
(transformers_venv) [z python]% pip install sklearn    # ML library
(transformers_venv) [z python]% pip install ipython    # python IDE
```

Use 'pip freeze > requirements.txt' to save the system requirements
into a file, e.g., correct versions of Python and packages such as
transformers and tensorlow, etc. that make everything work.
The requirements.txt file can be used by others to duplicate the
development environment.
[See https://towardsdatascience.com/virtual-environments-104c62d48c54]

This shows the actual version of Python used by this venv.
```
$ ls -al ./transformers_venv/bin/python
```

In an activated venv, a call to python places you in the right version.
[NOTE: no need to call 'python3', a simple 'python' will give you 3.10]

```
(transformers_venv) [z python]% python
Python 3.10.7 (main, Sep 15 2022, 01:51:29) [Clang 14.0.0
(clang-1400.0.29.102)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

I use 'ipython' to work with Python interactively and to debug programs.

```
(transformers_venv) [z src]% ../transformers_venv/bin/ipython
Python 3.10.7 (main, Sep 15 2022, 01:51:29) [Clang 14.0.0 (clang-
1400.0.29.102)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.5.0 -- An enhanced Interactive Python. Type '?' for help.

[ins] In [1]:
```

Finally, type deactivate to exit venv and return to regular OS.

```
(transformers_venv) [z python]% deactivate
[z python]%
```