Deep Learning to predict glass properties

F. Pigeonneau

October 26, 2024

Contents

1 Description of the repository

This python module has been developed to determine the following glass properties:

- density,
- Young's modulus,
- melting temperature,
- softening temperature,
- glass transition temperature (assuming equivalent to the annealing temperature).

Artificial Neural Networks have been fitted using datasets extracted from Interglad V8 and SciGlass databases. The details of the development of the models can be found in the article:

F. Pigeonneau, M. Rondet, O. de Lataulade and E. Hachem (2024). *Physical-informed deep learning prediction of solid and fluid mechanical properties of oxide glasses* J. Non-Cryst. Solids, under review, `http://dx.doi.org/10.2139/ssrn.4997217`.

2 Requirements

Python modules involved in this module are:

- `numpy`
- `pandas`
- `matplotlib`
- `seaborn`
- `sklearn`
- `tensorflow`

3 Repository constitution

Under this repository, various files are provided. The list is the following:

- `dissociationenergy.csv`
- `glassdata.py`
- `network.py`
- `glassproperties.py`
- `nntrainingproperty.py`
- `nntrainingrho.py`
- `nntrainingE.py`
- `chartglassproperties.py`

3.1 `dissociationenergy.csv`

The `dissociationenergy.csv` gathers for the twenty oxides the dissociation energy, second column, in GPa and the packing volume in cm^3/mol. This `csv` file is used in the determination of the packing atomic factor to compute the Young's modulus.

3.2 `glassdata.py`

`glassdata.py` file defines the GlassData class. It gathers various functions to clean datasets and manages data for the training stage. It is composed by the following variables

- `oxide`: Array of string Names of the oxides of the glass composition.
- `nameproperty`: String Name of the property of the glasses.
- `nsample`: Integer Number of the glass.

- `noxide`: Integer Number of oxides constituting the glass composition.
- `nbO`: Array of integer of the noxide oxides number of oxygen in each oxide.
- `cation`: Array of string of the noxide oxides name of the cation in each oxide.
- `nbcation`: Array of integer of the noxide oxides number of cation in each oxide.
- `Moxide`: Array of float of the noxide oxides molar mass of each oxide.
- `x`: Array of float (nsample,noxide) Molar fraction of the noxide oxides of the nsample glass composition.
- `y`: Array of float of the nsample glass composition. Property of glass of the data set.
- `xmin`: Array of float of the noxide oxides Minimum of the molar fraction of each oxide (=0)
- `xmax`: Array of float of the noxide oxides Maximum of the molar fraction of each oxide in the data set.
- `occurence`: Array of integer Numbers of composition for which the molar fraction of each oxide is stricly different to zero.
- `MolarM`: Array of float of the nsample glass composition. Molar mass of each glass composition in kg/mol.

The methods associated to this GlassData class are:

- `load(self,filename)`: Loading of dataset from csv file called filename.
- `info(self)`: Print information about dataset.
- `shape(self)`: Return shape of input dataset.
- `oxidemolarmass(self)`: Molar mass of each oxide in kg/mol.
- `molarmass(self)`: Molar mass for each glass.
- `bounds(self)`: Determination of the minimum, maximum and occurence of each oxide.
- `normalize_x(self)`: Normalize of x dataset.
- `normalize_y(self)`: Normalize of y dataset.
- `split(self,train,valid)`: Define split indices into training, validation and test subsets.
- `physicalx(self,x)`: Computational of x from the norlaized values.
- `physicaly(self,y)`: Computational of y from the norlaized values.
- `savedata(self,filesavedata,listtodrop=None)`: Saving of the data set in a csv file with a removal of a list of samples if it exists.

- `xglasstoxdb(self,oxide,xglass)`: Copy the array of the glass composition xglass with a list of oxide in the

formatted array of the dataset.

- `datacleaning(self,filename,xtotal,probamin,probamax,xminor,minoxidefraction, filteringoxide,Plot)`: Cleaning of the dataset gathered in the filename extracted from Interglad. Here,

all unexpected characters and lines are removed. Alphanumeric characters are transformed into float.

- `FilteringProperty(self,probamin,probamax,Plot)`: Cleaning of database by keeping glass samples for which the probability of occurrence

is in tha range `[probamin,probamax]`.

- `FilteringOxide(self,probamin,probamax,ioxide,Plot)`: Cleaning of database by keeping glass samples for which the probability of

occurrence of oxide ioxide is in the range `[probamin,probamax]`.

- `pdfoxide(self,ioxide,density,Plot,filename)`: Determination of the pdf of the oxide[ioxide].

- `familyrandomcomposition(self,Nglass,I,J,xmol0,dx0)`: Determination of random composition of Nglass composition centered on a composition

constituted of a list I of a sub-set oxides with centered composition xmol0.

- `randomcomposition(self,Nglass,xmax)`: Determination of random composition of Nglass composition.

- `GlassDensity(self,nnmodel,oxide,x)`: Determination of the glass density from the molar volume.

- `YoungModulus(self,nnmodel,datadisso,oxide,x)`: Determination of the Young's modulus as a function of a composition of glass with

the list of oxide and molar composition x.

3.3 `network.py`

`network.py` is a class used to define, compile, fit an ANN model. The parameters of this class are:

- `shape`: Integers corresponding of the number of features (= number of oxides) and number of samples in the data-set.

- `arch`: Array defining the number of layers and each associated neurals.

- `actv`: Activation function of hidden layers in ANN model.

- `final`: Function of the output layer.
- `k_init`: Kernel of initialization of parameters of the ANN.
- `history`: Information about the fitting of the ANN.
- `namearch`: Name of architecture used to save and call a model.

The methods associated to this are:

- `build(self,shape,arch,actv,final)`: Build model.
- `compile(self, lr=1.0e-3)`: Compile model.
- `info(self)`: Print infos about model.
- `fit(self,x_train,y_train,x_val,y_val,epochs,batch_size)`: Fitting model on data.
- `plot(self,outputfile`'loss.png',savefig=False)=: Plot accuracy and loss.
- `save(self, filename)`: Save model to file
- `load(self,filename)`: Load model from file
- `ArchName(self,arch)`: Name summarizing the architecture of the model.

3.4 `glassproperties.py`

This file compiles various functions useful in glass and melting sciences.

- `muVFT(T,Amu,Bmu,Tmu)`: Determination of the dynamic viscosity of glass forming liquid using VFT's law given by

$$\mu = A_\mu \exp\left(\frac{B_\mu}{T - T_\mu}\right).$$

- `Henry(A,B,T)`: Solubility of gas in a glass forming liquid from the Henry's law given by

$$L = A \exp\left(\frac{B}{T}\right).$$

  The unit of L is mol/(m^3 Pa$\hat{}\beta$) with beta the exponent in pressure depenence:

$$C = LP^\beta.$$

- `diffusivity(A,B,T)`: Diffusivity of gas in a glass forming liquid from the law given by

$$D = A \exp\left(-\frac{B}{T}\right).$$

  The unit of D is m^2/s.

- `Kequi(A,B,T)`: Computation of the equilibrium constant of a chemical reaction of oxidation-reduction in glass forming liquid.
- `VFTcoefficients(Tm,Ts,Tg)`: This function determines the three coefficients, A, B and T_0 of the VFT's law

$$\log(\eta) = A + \frac{B}{T - T_0}.$$

  The temperatures are given in Celsius or in Kelvin. The coefficient A is determined to have the viscosity in Pa.s.
- `Tsoft(Amu,Bmu,Tmu)`: Computation of the Littletown temperature based on the VFT's law of a glass forming liquid given by

$$\mu = A_\mu \exp\left(\frac{B_\mu}{T - T_\mu}\right).$$

- `Tglass(Amu,Bmu,Tmu)`: Computation of the glass transition temperature based on the VFT's law of a glass forming liquid given by

$$\mu = A_\mu \exp\left(\frac{B_\mu}{T - T_\mu}\right).$$

- `MolarMass(name)`: Molar mass of a chemical molecule given by name in kg/mol.
- `PoissonRatio(dbE,nnmodelEsG,db,nnmodel,oxide,x)`: Determination of the Poisson's ratio from the Makishima-Mackensie's model.

3.5 `nntrainingproperty.py`

This script is an example achieving the training of an ANN model using our own GlassData and NeuralNetwork classes.

3.6 `nntrainingrho.py`

Script used to train an ANN model to fit the molar volume obtained for the data-set on density.

3.7 `nntrainingE.py`

Script used to train an ANN model to fit the atomic packing factor obtained for the data-set on Young's modulus.

3.8 `chartglassproperties.py`

Script used to determine charts of mechanical properties.

4 Data sets

Data sets for the various properties are gathered under the directory `Database`.

The list of data sets are the following:

- `E20oxides.csv`: File given the Young' modulus in GPa as a function of 20 oxides.
- `rho20oxides.csv`: File given the density in kg/m^3 as a function of 20 oxides.
- `Tannealing20oxides.csv`: File given the annealing temperature in K as a function of 20 oxides.
- `Tliq20oxides.csv`: File given the liquidus temperature in K as a function of 20 oxides.
- `Tmelt19oxides.csv`: File given the melting temperature in K as a function of 19 oxides.
- `Tsoft20oxides.csv`: File given the softening temperature in K as a function of 20 oxides.

5 Trained ANN models

Models of artifial neural networks used to predict properties has been previously trained. They are under the directory `Models`.

- `nnEsG3c20.h5`: ANN of 3 layers of 20 neurals determined the atomic packing factor from the Young's data set `E20oxides.csv`.
- `nnmolarvol3c20.h5`: ANN of 3 layers of 20 neurals determined the molar volume from the density data set `rho20oxides.csv`.
- `nnTannealing3c20h5`: ANN of 3 layers of 20 neurals determined the annealing temperature corresponding to the glass transition temperature from the data set `Tannealing20oxides.csv`.
- `nnTmelt3c20.h5`: ANN of 3 layers of 20 neurals determined the melting temperature from the data set `Tmelt20oxides.csv`.
- `nnTsoft3c20.h5`: ANN of 3 layers of 20 neurals determined the softening temperature from the data set `Tsoft20oxides.csv`.