**Deep Neural Network (DNN) inversion of gravity and magnetic data to subsurface modeling:**

**Description**

This package implements advanced deep neural network (DNN) inversion techniques to model subsurface properties using gravity and magnetic data.

The intelligent inversion problem requires substantial training data, which can be limited in real-world scenarios. To overcome this challenge, my package introduces a novel technique for simulating geopotential datasets that closely resemble real-world subsurface properties and their corresponding geopotential data.

Installation Instructions:

a. Installation via pip:

pip install MagInv-DNN

b. Installation from Source:

1. Clone the Repository:

git clone https://github.com/Zahra-Ashena/GrvMag\_DNNinv.git

2. Navigate to the Package Directory:

cd MagInv-DNN

3. Install the Package:

Regular installation:

python setup.py install

Development installation:

python setup.py develop

4. Dependencies:

numpy, scipy, matplotlib, Jason, math, random, tensorflow

**Dataset Simulation and DNN Inversion Technique:**

The MagInv-DNN package introduces a magnetic inversion technique using DNN to estimate basement topography. The training dataset is simulated by adopting a new technique. Employing parallel computing algorithms, thousands of forward models of the subsurface with their corresponding gravity anomalies are simulated in a few minutes. Each forward model randomly selects the values of its parameter from a set of predefined ranges based on the geological and structural characteristics of the target area (Figure. 1).

**Diagram

Description automatically generated**

Figure. 1 The step-by-step research procedure.

**1. Dataset simulation:**

A 2.5D forward model is created that partitions the upper crust into two layers, as sediments and basement (Figure. 2). Each layer is discretized into an assembly of rectangular prisms isolated from their surrounding geological environment. The physical parameters including susceptibility of the layers, are incorporated into the model as a priori information.

The total length of the model and each prism are specified based on the dimensions and resolutions of the anomalies being investigated. In the context of geophysical modeling, a 2.5D model refers to a simplified representation of a 3D subsurface structure. It assumes that the subsurface varies in two dimensions (along the X direction (or length of the model) and depth) but remains constant in the third dimension (Y direction (or width of the model)). This simplification allows for computational efficiency while still capturing important variations in the subsurface.

A width of 20 times a prism's length is used to mathematically satisfy the infinite width of the 2.5D model. The model is then extended at both lateral sides to avoid potential edge effects.

The adopted dimensions of the forward model cannot be modified during the dataset simulation and remain fixed.

To create numerous representations of the forward model structure, the topography of the salts and basement layer randomly changes. To this end, several random structural parameters are defined that select their values from predefined ranges.

Chart, bar chart, histogram

Description automatically generated

Figure 2: Schematic view of the 2.5D forward model and observation points (red dots) on the surface.

The following bullet points discuss examples of the random parameters briefly:

* The average depths of the salts and basement layers
* The number of anomalies to be loaded on top of the salts and basement layers
* The origin coordinate of the anomalies
* The heights of the anomalies (topography variations)
* The lengths of the anomalies

Given the number of training data, by assignation from the list of the parameters, a different forward model is generated in each iteration, and its synthetic gravity and magnetic anomalies are calculated correspondingly.

First, in each iteration, two average depths are randomly selected from the corresponding range parameter and assigned to salts and basement layers. To prevent the salt and basement layers to overlap, a tolerance of is imposed as the required minimum gap between these two layers. To change the topography of the layers, my strategy is to create some random anomalies with different lengths and heights and have them loaded on top of the prisms. The Gaussian function is used to generate anomalies with zero mean and standard deviation equal to the lengths of the anomalies.

The number of anomalies and their corresponding lengths and heights are randomly selected from the number of anomalies, length of anomalies, and height of anomalies ranges. Moreover, the center location of each anomaly is also randomly chosen from a range equal to the number of prisms. Having the width, random length , height , and center location , the Gaussian function is used to create each anomaly with zero mean and standard deviation equal to half of the random length,

Once the forward models are simulated their gravity and magnetic effects are calculated over the observation profile, with the center of the profile set as the origin coordinate of the model.

To simulate the dataset, use the file “Dataset\_simulation.py”.

**2. DNN Inversion:**

Given a training dataset, a DNN model is trained by approximating weightparameters **.** The DNN model to conduct nonlinear mapping from the geophysical data to subsurface model

To train a DNN model we need to design the architecture of the network, including the number of layers, the number of nodes of each layer, how these layers should be connected to each other, the activation function, optimization parameters, etc.

Use file “DNN\_training.py” to train the DNN model.

**3. DNN Model Predictions:**

Use file “TopoBase\_predict.py” to evaluate the trained DNN model on unseen data.

**4. Real Data:**

The file “RealData\_predict.py” can be used to estimate basement topography over real magnetic data.

License: Clearly state the license under which your package is distributed. This informs users about the permissions and restrictions for using and modifying your code.

Contribution Guidelines: If you are open to contributions from the community, provide guidelines for how users can contribute code, report issues, and submit pull requests.

Acknowledgments and Credits: If your package uses third-party libraries, datasets, or any other resources, acknowledge and credit them appropriately.

Contact Information: Provide a way for users to get in touch with you if they have questions, feedback, or need support related to the package.