**National Cancer Institute (NCI)-US Department of Energy (DOE) Collaboration Use Case for GGMD Pipeline on Microsoft Azure Machine Learning Studio**

**Introduction**

This Use Case Template is intended to describe and interpret real-world examples of computational resources developed by the NCI-DOE Collaboration—and others—that can be used to explore solutions to complex cancer-risk factors, cancer diagnosis and treatments, as well as related biomedical research challenges. This template was developed with input from external subject matter experts. We ask you to share your experience in using these computational resources by completing this template in accordance with the instructions below. To provide feedback please contact *computational-cancer@nih.gov* Thank you!

**Instructions**

Please fill in the information requested. Each section includes the level of expected length or level of detail: **brief** (one-two sentences), **informative** (two-four sentences), and **detailed** (as much detail as required to properly inform the user). As each use case is different, please provide additional information as necessary.

**Use Case Overview**

Provide a brief overview of: *(1) use case goals and research focus; (2) methods, data type, and computational resource used; and (3) results and expected outputs. What is the expected time to reproduce this use case?*

This use case aims to provide users of the Generalized Generative Molecular Design (GGMD) pipeline with a template for application on Azure Machine Learning Studio (AML). For this example, we have used an autoencoder trained with a subset of the zinc dataset, to generate compounds using a roulette selection type with an initial population size of 50, max population of 50, and 8 epochs. This use case uses previously generated models and datasets, with other examples currently available being an autoencoder trained with a subset of the Zinc database. A CPU cluster (Standard\_DS3\_v2 (4 cores, 14 GB RAM, 28 GB disk)) was used for this use case. After completion of this use case, the user can expect generated compounds with information on the fitness, and information on how the molecule was generated. This exact use case with the provided files performed will take 2-20 minutes depending on the generation type and the scoring type. You can find all necessary files in the GitHub repo at https://github.com/joverhul/GGMD\_Azure. For instructions for how to run GGMD not on AML please go to https://github.com/CBIIT/GGMD.

**Use Case Details**

Scientific Background and Goals

* **Problem**: This use case uses a previously trained autoencoders (AE) as a part of the generalized generative molecular design (GGMD) pipeline to predict novel molecules. Users can use Azure Machine Learning (AML) resources to perform the generalized generative molecular design pipeline.
* **Background**: Generative molecular design is a useful tool that drug discovery groups have been using to fully explore the chemical space. This use case provided the setup to perform this pipeline using cloud services such as AML.Previously, users have been able to use the GGMD process on local HPCs, including FRCE and Biowulf. A user needs to have a previously trained autoencoder that can be used with the GGMD to produce compounds on a set scorer. With this use case, we implemented the GGMD process on Microsoft Azure, speeding up the computational time and broadening the use for the pipeline.
* **Scientific purpose**: This use case shows the implementation of the GGMD pipeline on a cloud computing platform, specifically this is a working example using AML. A user can generate novel compounds using the Junction Tree Variational AutoEncoder (JTVAE) or AutoGrow implementation in GGMD. This use is best positioned for individuals who do not have access to their own HPC resources but are able to use cloud resources. Use of cloud resources also gives access to using large datasets with computational times being reduced.
* **Goal**: The goal of this use case is to provide an example for future users to use on Microsoft Azure, or other cloud computing services. The long-term goal is to allow users to utilize the cloud services to run generative design on much larger datasets than what was provided. This use case also highlights the ability that users who do not have their own HPC resources are able to use the provided resources from cloud services.
* **Objective(s)**: This use case provides an example of using a previously trained AE to generate compounds using the GGMD pipeline on Microsoft Azure. This use case highlights how to use an AMPL or LogP scoring function with the compound generator AutoGrow or JTVAE. This use case shows how to use the GGMD method using AML, which can be used to speed up computational time, as well as gives access to users without direct access to a HPC.
* **Constraints**:Users who have access to their own HPC or are working with a smaller dataset may not benefit from this specific use case on Azure. This use case is developed specifically for Microsoft Azure. In general, a user is limited to the chemical space that the model has been trained on.

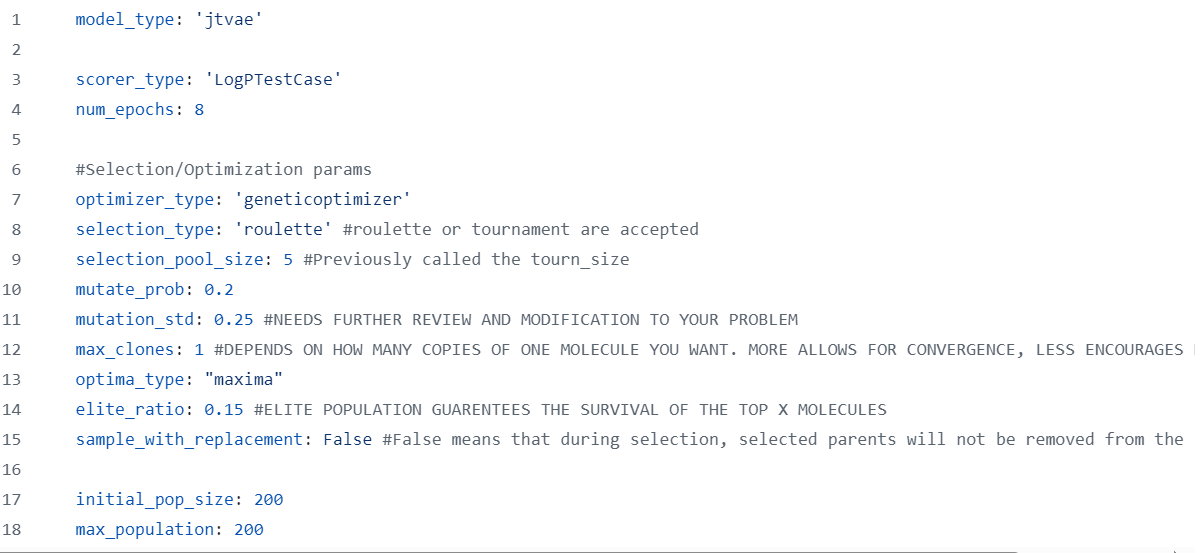
User Background and Skills

* **Education**: This use case is specifically developed for Microsoft Azure, so having a working knowledge of how to run .yaml files and submit jobs on Microsoft Azure Machine Learning is recommended. Also, having a general knowledge of generative models and chemistry is recommended.
* **Skills required**: This use case requires a basic understanding of python coding, and running jobs with .yaml files, specifically the format needed for Azure Machine Learning. Additionally, having a basic understanding of the scorers you are interested in implementing is recommended. Some experience with Microsoft Azure Machine Learning is recommended.

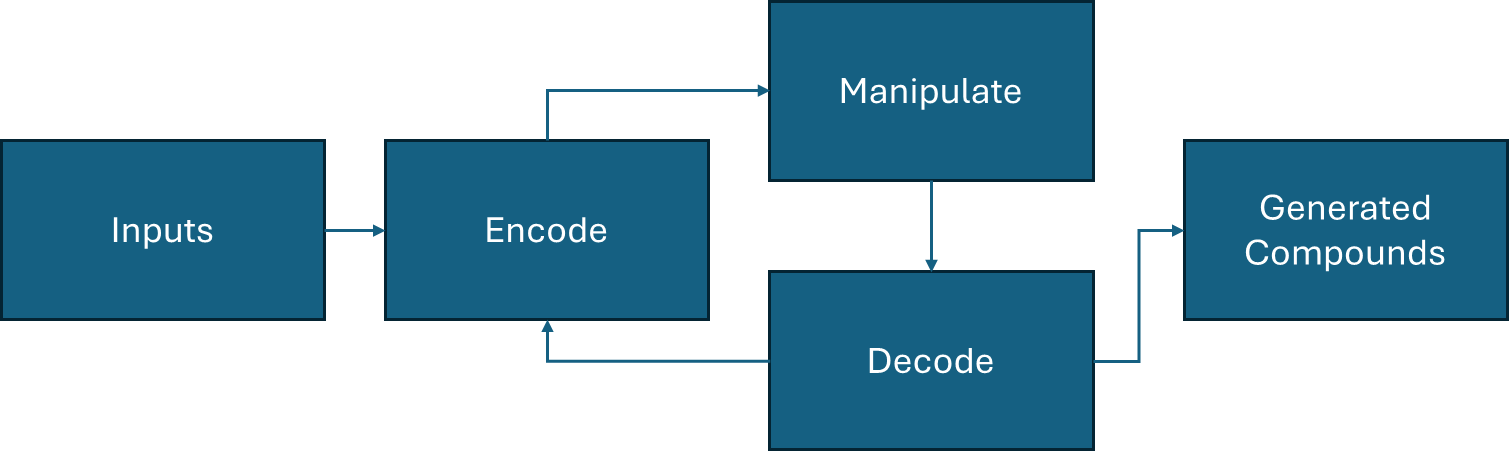
Methods

* **Computational resource selection**: Microsoft AML was selected due to its cloud use capabilities. Users can upload data assets to their storage bin, and then access these assets to run jobs. These resources give users the ability to use cloud services if they do not have access to their own HPC which could also be beneficial to run larger datasets.
* **Inputs**: Using the github, the inputs needed are clearly described and an example is provided (https://github.com/joverhul/GGMD\_Azure). The inputs from the example include the smiles input file (zinc\_smiles.txt), the vocabulary used for the model (all\_vocab.txt), the AE model trained (model.epoch-35), and a path to the working directory for an output. This list of inputs can be stored as data assets on AML. The inputs necessary for using the GGMD method on AML are added to AML\_GGMD.yaml file that is used to submit a job to AML. All assets are added to the AML\_GGMD.yaml file that is then used to submit the python script for the GGMD pipeline. These inputs are all customizable and users should provide their own models they are interested in.
* **Methods**:This use case provides the versatility to use either AutoGrow or a JTVAE compound generator, with the LogP or AMPL trained model scoring function. Before this use case, the autoencoder was prepared using the pipeline highlighted on GitHub (<https://github.com/CBIIT/JTVAE/tree/main>). This process requires a vocabulary generated, that is necessary for the GGMD pipeline. This use-case assumes that the user has already trained an autoencoder and followed those steps. This use case only requires the CPU cluster Standard\_DS3\_v2 (4 cores, 14 GB RAM, 28 GB disk) provided by AML. Instructions for using this tool on AML can be found at <https://github.com/joverhul/GGMD_Azure/blob/main/AML_instructions.md>.

Inside of the example folder, users can find three provided examples. Below is an example input of the config.yaml file. Inside of the examples README.md file you will find documentation for GGMD parameter input.



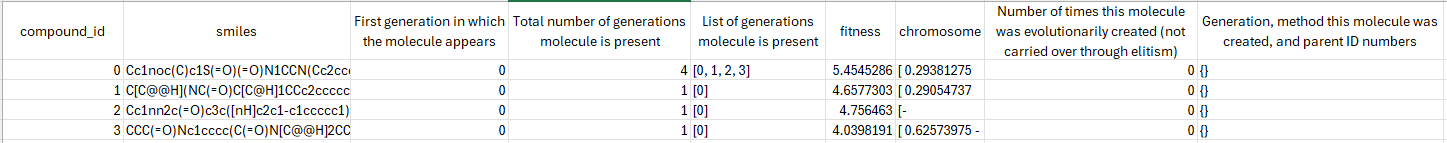
* **Workflow figure**:

**

* **Methods customizability**: This use case is highly customizable. The provided examples include two different compound generators and two different scoring functions. Users can apply any desired scoring function, following a similar code setup to get proper implementation. Users can also train their own AE to generate compounds or provide their own set of SMILES strings as an initial population.
* **Common pitfalls**:Some common issues with this computational resource is not correctly adding the assets to the yaml file and within the job submission. Other pitfalls could include issues with the computational resources on Azure, which may crash occasionally. Ensure if using a JTVAE model that the reconstruction accuracy is accurate. The original GGMD implementation has provided code to test this.
* **Requirements**:This use case requires dedicated cpu resources. The jobs can run on non-dedicated resources but could result in jobs failing due to time depending on parameters set.

Results

* **Outputs**: The final output will result in a .csv file that shows the generated compounds. This output shows the compounds generated after 8 epochs, with information on how they were generated such as initial, mutation, crossover, and information from the scorer provided.
* **Results**: Below is a figure of the head of the output file. The columns are clearly labeled with teir contents. The fitness score is the output of the scoring function that the user has decided to use. The compounds generated will have a fitness score given, and how the compound was produced will be provided. Also provided will be the number of times a specific molecule was generated.



Discussion

* **Results interpretation:** The results for this example provide a list of compounds generated, or from the initial library that are favorable to the scorer used. A pitfall the user needs to be aware of is the generated compounds over fixating on certain types of compounds, and not fully exploring the desired compound space. The user also needs to be careful with the model over exploring a compound space, predicting compounds that are far outside of the initial library.
* **Objective(s) met**:The original objective was met, and users are able to access these three specific examples on the github link provided.

Conclusion

* **Limitations:** The limitations with this specific use case include compounds being generated that are not synthetically feasible. Other limitations include use of non-dedicated computing resources, which may result in failed jobs on AML. This use case is modifiable to any scorer or model type that is used within the GGMD pipeline on AML. This use case could also be used as a template with other cloud services if they have similar infrastructure as AML.
* **Goal**: *This work highlights an example use of the GGMD pipeline on cloud services. This will give researchers the ability to use this pipeline on services if they do not have access to their own HPC as well as if they are using a larger dataset.*

**Key Links to Related Portal Pages:**

*Related resources or activities relevant to this use case may be found at the links below.*

*NOTE: these are intentionally not hyperlinked because the site is under development.*

* Publications

Wengong Jin, R. B., Tommi Jaakkola (2019). "Junction Tree Variational Autoencoder for Molecular Graph Generation." arxiv.

* Computational Resources: Models, Software, Datasets

https://github.com/CBIIT/GGMD

https://github.com/joverhul/GGMD\_Azure

* Educational Materials