Inverse Design of Organic Semiconductors for Organic LEDs and Solar Cells

Chavda Haarit Ravindrakumar
Roll: 23110077 Programme: CSE Btech 27
Indian Institute of Technology Gandhinagar
23110077@iitgn.ac.in

I. Problem Statement

Inverse molecular design is a method that creates molecules with specific desired properties that are very specific to their applications. This project aimed to design organic semiconductors for OLEDs and solar cells, molecules with unique properties such as low s1-t1 gap and high oscillator strength. S1-T1 gap is general sound 3ev, which falls in the visible spectrum of the light. Traditional methods, which involve manually designing molecules and running expensive calculations, take time, human force, and computational power to find the molecules. This method was no less than brute forcing all the possible molecules. We have to do the same task more efficiently using the modern machine learning models and concepts.

II. Proposed Solution

The task of designing molecules with desired properties can be modeled as an optimization problem. A genetic algorithm, JANUS (Parallel Tempered Genetic Algorithm Guided by Deep Neural Networks for Inverse Molecular Design), was used to automate this process. JANUS, developed by researchers from Stanford University and the University of Toronto, leverages a very complex and efficient architecture and deep neural networks (DNNs) to filter molecues which reduces the computational costs associated with property evaluation.JANUS explores chemical space efficiently and uses a deep neural network to select the molecules that have the best fitness, reducing the time needed for calculations. We also trained a transformer model to predict properties like S1-T1 energy gaps and oscillator strengths, improving speed and accuracy.

III. Architecture of JANUS

The architecture of JANUS integrates parallel tempered genetic algorithms with deep learning to enhance molecular optimization. It maintains two populations: an exploratory population for sampling diverse regions of chemical space and an exploitative population for refining promising molecules. These populations use tailored genetic operators such as mutations and crossovers, implemented through the STONED algorithm and SELFIES representation, ensuring efficient and valid molecular generation. The exploratory population leverages a DNN trained through active learning to create selection pressure on candidates and filter the best ones, while the exploitative population employs molecular similarity metrics for optimization. Molecules are exchanged between populations to balance exploration and refinement. This architecture provides a robust framework for navigating vast chemical spaces efficiently.

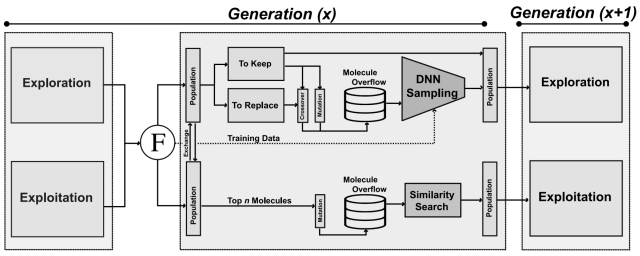

Fig. 1: JANUS architecture (Creds: The Matter Lab - aspuru-guzik-group)

IV. Workflow of JANUS

The workflow begins by initializing two populations with molecular structures. At each generation, molecules compete based on a fitness function tailored to the desired properties. The exploratory population generates new molecules using random string manipulations of SELFIES and filters them using a DNN. The exploitative population focuses on high-performing regions, selecting molecules based on molecular similarity. Excess molecules are filtered to retain only the most promising candidates, significantly reducing computational costs. Periodic exchanges between populations allow high-fitness molecules to enhance exploration and refinement simultaneously. This iterative process continues until optimal molecules are identified. For computational efficiency, the DNN filters potential candidates before property evaluation. For example, out of 1,000 generated molecules, the DNN may select 100 promising candidates, reducing the computational load by a factor of 10.

V. Challenges

Although JANUS, with its deep neural network (DNN), significantly reduces computation time compared to traditional algorithms, the fitness function—based on oscillator strength

and the S1-T1 energy gap—remains computationally expensive to evaluate. Directly calculating these properties for all generated molecules resulted in considerable delays.

To address this, several approaches were explored. Initially, I tried storing the calculated properties of molecules using PySCF and reused these values if the same molecule appeared again. However, JANUS does random molecular modifications made the probability of generating identical molecules extremely low, making this solution ineffective.

Another method I tried involved training a transformer model to predict the properties of molecules. The transformer was fine-tuned using filtered data to estimate S1-T1 energy gaps and oscillator strengths. This method is very efficient and accurate, reducing computation time significantly. Fine-tuning the transformer required significant effort, but once trained, it quickly and reliably predicted molecular properties, enabling faster optimization.

VI. Results

Code Repository: Link to the code: https://github.com/haarit19058/janus-final.git

1) **Running JANUS with PySCF:** JANUS was run on a filtered dataset of 200 molecules for three generations, each with a size of 20 molecules. This process took three days due to the computational intensity of property evaluations. The best-performing molecule identified in this run is shown in Figure 2.

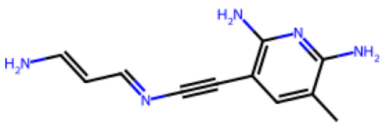

Fig. 2: Best-performing molecule identified using JANUS and pyscf.

2) **Transformer Predictions on Filtered Training Data:** The transformer was used on the filtered training dataset to predict values. The generation size is 500 and num exchanges are 10 and it took about 2-3 generation to obtain these molecules. Results are shown in Figures 3 and 4.
3) **Refined Transformer with Fitness Function and Filtered Data:** Using a more refined version of the fitness function and filtering data such that the `s1-t1` gap is between 0.2 and 0.8 and the oscillator strength is greater than 0.2. The generation size in this setup was 700 and num exchanges was 10 and the results are shown in Figure 7.

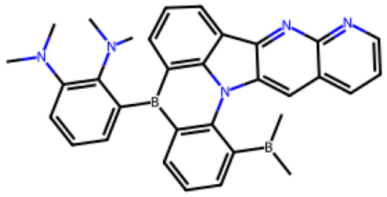

Fig. 3: Molecule obtained from result 2.

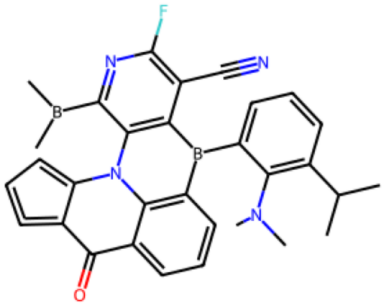

Fig. 4: Molecule obtained from result 2.

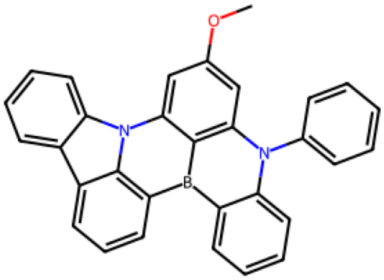

Fig. 5: Results using the refined transformer and filtered data.

VII. Future Work

Short-Term Targets:

1) We can try to come up with an alternative so that reduces the time it takes for pyscf to calculate the fitness function

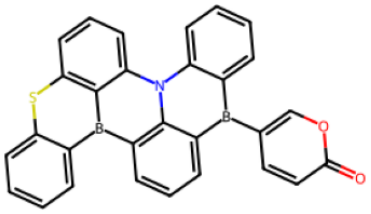

Fig. 6: Results using the refined transformer and filtered data.

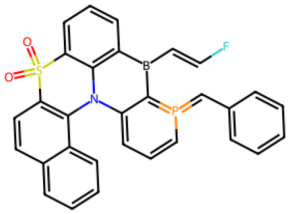

Fig. 7: Results using the refined transformer and filtered data.

values or explore alternative methods that can achieve the same result more quickly.

2) I updated the current transformer model. Although the transformer is trained on the zinc dataset, which is available in transformer library of python, I fine-tuned it for 10 epochs to meet our needs. However, this transformer doesn't perform well with organic molecules and using the predicted values for the properties may lead the resultant molecules to be confined in a close proximity of the molecules in the dataset in the chemical space. To improve results, we can train our own version of the transformer using our dataset of s1-t1 gaps and f1 energy. This would provide more accurate predictions, training on a custom dataset and creating a large enough dataset of molecules might take a considerable amount of time.

Long-Term Targets:

I aim to create our own genetic algorithm, specifically designed to optimize the f1 energy and s1-t1 energy gaps. This project requires in depth knowledge and experience in deep learning and the chemistry of the molecules.

REFERENCES

We are using the model made by the "The Matter Lab" and the description and architecture in this report follows from their papers which can be found at : https://github.com/aspuru-guzik-group/JANUS.git

The other resourcse that we referred to while making the model also belongs to the matter lab. The code which they had for LEDs guided me a lot. This can be found at https://github.com/aspuru-guzik-group/Artificial-Design-of-Organic-Emitters