Supplementary Information – Multi-Constraint Molecular Generation using Sparsely Labelled Training Data for Localized High-Concentration Electrolyte Diluent Screening

Jonathan P. Mailoa,[1*] Xin Li,[1] Jiezhong Qiu,[1] and Shengyu Zhang[2*]

1) Tencent Quantum Laboratory, Tencent, Shenzhen, Guangdong, China
2) Tencent Quantum Laboratory, Tencent, Hong Kong SAR, China

[*] corresponding author: jpmailoa@alum.mit.edu, shengyzhang@tencent.com

Baseline SSVAE Model Cost Function Derivation

Our work mostly utilizes the cost function originally developed by Kang, *et al* based on the work by Kingma, *et al*. The regression loss component $R_{SSVAE}(x,y)$ from Equation 3 in the main text is a straightforward regression square loss function and will not be discussed in more detail.

The VAE cost function for the labelled entries (**Equation 1**) originates from **Equation 1** in Kang paper and **Equation 6** in Kingma paper. It is described that the variational lower bound $-L(x,y)$ of the log-probability of a labelled instance (x,y) is:

$$\begin{aligned}
\ln p(x,y) &\geq E_{q_\phi(z|x,y)}[\ln p_\theta(x|y,z) + \ln p(y) + \ln p(z) - \ln q_\phi(z|x,y)] \\
&= E_{q_\phi(z|x,y)}[\ln p_\theta(x|y,z)] + \ln p(y) - D_{KL}(q_\phi(z|x,y)||p(z)) \\
&= -L(x,y) \\
L(x,y) &= -E_{q_\phi(z|x,y)}[\ln p_\theta(x|y,z)] - \ln p(y) + D_{KL}(q_\phi(z|x,y)||p(z))
\end{aligned}$$

The first term is simply the cross-entropy loss. When summed over all the n_L fully labelled samples in the minibatch and the n_x dimension corresponding to our molecule SMILES one-hot vector representation (see Kang paper for detail), the first term is converted to **(1)**:

$$-\sum_{i=1}^{n_L}\sum_{j=1}^{n_x}\left(x_{i,j}\ln x_{D,i,j} + (1 - x_{i,j})\ln(1 - x_{D,i,j})\right)$$

For the second term, we first remember that $p(y)$ is a multivariate normal distribution with mean matrix E ($n_y \times 1$) and covariance matrix C ($n_y \times n_y$) constructed from all the available training labels in the dataset, where n_y is the dimension of y. The multivariate probability distribution function is:

$$p(y) = \frac{1}{\sqrt{(2\pi)^{n_y} det(C)}} e^{-\frac{1}{2}(y-E)^T C^{-1}(y-E)}$$

When the term $(-\ln p(y))$ is expanded for all molecule labels y_L in the minibatch, we have **(2)**:

$$\sum_{i=1}^{n_L}\frac{1}{2}\left(n_y \ln 2\pi + \ln(det(C)) + \sum_{j=1}^{n_y}(y_{L,i,j} - E_j)\sum_{k=1}^{n_y}(y_{L,i,k} - E_k)C^{-1}_{k,j}\right)$$

For the third term, we recall that the definition of Kullback-Leibler divergence loss is:

$$D_{KL}(P||Q) = \sum_{x \in \chi} P(x) \ln\left(\frac{P(x)}{Q(x)}\right)$$

We also remember that the prior distribution $p(z) = N(z|0,I)$ and approximated posterior distribution $q_\phi(z|x,y) = N\left(z|\mu_\phi(x,y), diag\left(\sigma_\phi^2(x,y)\right)\right)$ are used in the Kang paper. Before we proceed with the derivation, recall the Gaussian function integral formulas $\int_{-\infty}^{\infty} e^{-\alpha x^2} dx = \sqrt{\pi/\alpha}$, $\int_{-\infty}^{\infty} x e^{-\alpha x^2} dx = 0$ and $\int_{-\infty}^{\infty} x^2 e^{-\alpha x^2} dx = \frac{1}{2}\sqrt{\pi/\alpha^3}$. Also recall that $\int_{-\infty}^{\infty} e^{-\alpha(x-\beta)^2} dx = \sqrt{\pi/\alpha}$, and $\int_{-\infty}^{\infty} (x-\beta) e^{-\alpha(x-\beta)^2} dx = 0$, and $\int_{-\infty}^{\infty} (x-\beta)^2 e^{-\alpha(x-\beta)^2} dx = \frac{1}{2}\sqrt{\pi/\alpha^3}$ hold for the constants α and β. For the sake of clarity, in the following derivation we will drop x, y, and the subscript ϕ from the functions and just use z, with n_z being its dimension. Furthermore, note that the prior and approximated posterior distributions are independent between different z dimensions, so we can process them individually. i and j are the labelled molecule and the z dimension indexes, and we will drop this subscript in the integral for clarity:

$$q(z_{i,j}) = \frac{1}{\sigma_{i,j}\sqrt{2\pi}} e^{-\frac{(z_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}$$

$$p(z_j) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z_j^2}{2}}$$

$$D_{KL}\big(q(z_{i,j})||p(z_j)\big) = \int_{-\infty}^{\infty} \frac{1}{\sigma_{i,j}\sqrt{2\pi}} e^{-\frac{(z-\mu_{i,j})^2}{2\sigma_{i,j}^2}} \ln\left(\frac{1}{\sigma_{i,j}} e^{\frac{z^2}{2} - \frac{(z-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right) dz$$

$$= \frac{1}{\sigma_{i,j}\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{(z-\mu_{i,j})^2}{2\sigma_{i,j}^2}} \left(-\ln\sigma_{i,j} + \frac{(z-\mu_{i,j})^2}{2} + (z-\mu_{i,j})\mu_{i,j} + \frac{\mu_{i,j}^2}{2} - \frac{(z-\mu_{i,j})^2}{2\sigma_{i,j}^2}\right) dz$$

$$=-\ln\sigma_{i,j} + \frac{\sigma_{i,j}^2}{2} + 0 + \frac{\mu_{i,j}^2}{2} - \frac{1}{2}$$

The readers can convince themselves that repeating the exercise above using the joint distribution $D_{KL}\big(q(z_i)||p(z)\big)$ will simply generate the summation of the term over j (the n_z dimensions of z). After we consider the summation over i and j and put back the z notation on μ and σ for clarity, we get the following term **(3)**, completing **Equation 1** of the main text:

$$-\sum_{i=1}^{n_L}\sum_{j=1}^{n_z}\frac{1}{2}\left(1 + \ln\sigma(z_{i,j})^2 - \mu(z_{i,j})^2 - \sigma(z_{i,j})^2\right)$$

The VAE cost function for the unlabelled molecule entries (**Equation 2**) originates from **Equation 2** in Kang paper and **Equation 7** in Kingma paper. It is described that the variational lower bound $-U(x,y)$ of the log-probability of an unlabelled instance (x) is:

$$\ln p(x) \geq E_{q_\phi(y,z|x)}\left[\ln p_\theta(x|y,z) + \ln p(y) + \ln p(z) - \ln q_\phi(y,z|x)\right]$$

$$= E_{q_\phi(y,z|x)}\left[\ln p_\theta(x|y,z) + \ln p(y) - \ln q_\phi(y|x) + \ln p(z) - \ln q_\phi(z|x,y)\right]$$

$$= E_{q_\phi(y,z|x)}\left[\ln p_\theta(x|y,z)\right] - D_{KL}\big(q_\phi(y|x)||p(y)\big) - E_{q_\phi(y|x)}\left[D_{KL}\big(q_\phi(z|x,y)||p(z)\big)\right]$$

$$=-U(x)$$

$$U(x) =-E_{q_\phi(y,z|x)}\left[\ln p_\theta(x|y,z)\right] + D_{KL}\big(q_\phi(y|x)||p(y)\big) + E_{q_\phi(y|x)}\left[D_{KL}\big(q_\phi(z|x,y)||p(z)\big)\right]$$

The derivation for the first and the third term are identical to their counterparts for the fully labelled molecules. They will correspondingly generate **(4)**:

$$-\sum_{i=1}^{n_U}\sum_{j=1}^{n_x}\left(x_{i,j}\ln x_{D,i,j} + (1 - x_{i,j})\ln(1 - x_{D,i,j})\right)$$

and **(5)**:

$$-\sum_{i=1}^{n_U}\sum_{j=1}^{n_z}\frac{1}{2}\left(1+\ln\sigma(z_{i,j})^2-\mu(z_{i,j})^2-\sigma(z_{i,j})^2\right)$$

The second loss function term related to unlabelled y is significantly more complicated. Recall that we have the multivariate probability prior distribution function for y based on the fully labelled molecules:

$$p(y)=\frac{1}{\sqrt{(2\pi)^{n_y}det(C)}}e^{-\frac{1}{2}(y-E)^TC^{-1}(y-E)}$$

Also recall that we have taken the assumption of normal posterior distribution for the sampled y generated by the predictor $q_\phi(y|x)=N\left(y|\mu_\phi(x),diag\left(\sigma_\phi^2(x)\right)\right)$ below. i and j correspond to unlabelled molecule and y dimension indices, and we have dropped y from μ and σ for clarity:

$$q(y_{i,j})=\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}$$

The complex form for the joint distribution of $p(y)$ means we no longer have the luxury of separating the integrals based on the label space dimension the way we have done for $p(z)$.

$$D_{KL}(q(y_i)||p(y))=\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\ln\left(\frac{\sqrt{det(C)}}{\prod_j\sigma_{i,j}}e^{\left(\frac{1}{2}(y-E)^TC^{-1}(y-E)-\sum_j\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}\right)}\right)dy_i$$

$$=\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\left(\ln\frac{\sqrt{det(C)}}{\prod_j\sigma_{i,j}}+\frac{1}{2}(y-E)^TC^{-1}(y-E)-\sum_{j=1}^{n_y}\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}\right)dy_i$$

Note that this integration is performed over the n_y dimensions of y_i. We separate the terms for clarity. The first term in the parenthesis is simply a constant, and integrating over it generates **(6)**:

$$\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\ln\frac{\sqrt{det(C)}}{\prod_j\sigma_{i,j}}dy_i=\frac{1}{2}\ln\left(det(C)\right)-\frac{1}{2}\sum_{j=1}^{n_y}\ln\sigma_{i,j}^2$$

The third term in the parenthesis consists of n_y terms. For each of them (let's say $j = k$), the following situation applies:

$$-\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}dy_i$$

$$=-\int_{-\infty}^{\infty}\left(\prod_{j\neq k}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)dy_{i,j\neq k}\int_{-\infty}^{\infty}\left(\frac{1}{\sigma_{i,k}\sqrt{2\pi}}\frac{(y_{i,k}-\mu_{i,k})^2}{2\sigma_{i,k}^2}e^{-\frac{(y_{i,k}-\mu_{i,k})^2}{2\sigma_{i,k}^2}}\right)dy_{i,k}=-\left(1^{n_y-1}\right)\frac{1}{2}$$

After summation over j, we simply get $-\frac{n_y}{2}$, which is the summation term **(7)**.

Finally, for the third term in the parenthesis, we recognize that it is simply a double summation over n_y^2 terms.

$$(y-E)^TC^{-1}(y-E)$$
$$=\sum_{j=1}^{n_y}\sum_{k=1}^{n_y}C_{k,j}^{-1}(y_{i,j}-E_j)(y_{i,k}-E_k)=\sum_{j=1}^{n_y}\sum_{k=1}^{n_y}C_{k,j}^{-1}\left((y_{i,j}-\mu_{i,j})(y_{i,k}-\mu_{i,k})+(y_{i,j}-\mu_{i,j})(\mu_{i,k}-\right.$$

Based on the Gaussian function integral formula, remember that the terms $(y_{i,j}-\mu_{i,j})(\mu_{i,k}-E_k)$ and $(\mu_{i,j}-E_j)(y_{i,k}-\mu_{i,k})$ will contribute 0 to the integral, so we will drop them.

When $j = k$, we have the integral term **(8)**:

$$\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\left(\frac{1}{2}C_{j,j}^{-1}\left((y_{i,j}-\mu_{i,j})^2+(\mu_{i,j}-E_j)^2\right)\right)dy_i=\frac{1}{2}C_{j,j}^{-1}\sigma_{i,j}^2+\frac{1}{2}C_{j,j}^{-1}(\mu_{i,j}-E_j)^2$$

When $j \neq k$, the first term will additionally generate zero contribution to the integral, and we simply have the integral term **(9)**:

$$\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\left(\frac{1}{2}C_{k,j}^{-1}\left((\mu_{i,j}-E_j)(\mu_{i,k}-E_k)\right)\right)dy_i=\frac{1}{2}C_{k,j}^{-1}(\mu_{i,j}-E_j)(\mu_{i,k}-E_k)$$

Now that we have all the individual components within the double summation integrated, we can perform double summation of **(8)** and **(9)** to get the integral term **(10)**:

$$\int_{-\infty}^{\infty}\left(\prod_{j=1}^{n_y}\frac{1}{\sigma_{i,j}\sqrt{2\pi}}e^{-\frac{(y_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}\right)\left(\frac{1}{2}(y-E)^T C^{-1}(y-E)\right)dy_i = \frac{1}{2}\sum_{j=1}^{n_y} C^{-1}_{j,j}\sigma^2_{i,j} + \frac{1}{2}\sum_{j=1}^{n_y}(\mu_{i,j}-E_j)\sum_{k=1}^{n_y}(\mu_{i,k}-E_k)C^{-1}_{k,j}$$

Now that we have all the integral component terms **(6)**, **(7)**, **(10)** for y, after summation over all the unlabelled molecules n_U and putting back the $y_{P,i,j}$ into μ and σ for completeness, we have the following term **(11)**:

$$\sum_{i=1}^{n_U}\frac{1}{2}\left(\sum_{j=1}^{n_y}C^{-1}_{j,j}\sigma(y_{P,i,j})^2 + \sum_{j=1}^{n_y}\left(\mu(y_{P,i,j})-E_j\right)\sum_{k=1}^{n_y}\left(\mu(y_{P,i,k})-E_k\right)C^{-1}_{k,j} - n_y + \ln\left(det(C)\right) - \sum_{j=1}^{n_y}\ln\sigma(y_{P,i,j})^2\right)$$

This Kullback-Leibler divergence loss term for y completes our re-derivation of the original baseline SSVAE model cost function as implemented by Kang, *et al*, with **(4)**, **(11)**, and **(5)** forming the total VAE loss for the unlabelled molecules in **Equation 2** of the main text.

Impact of ConGen Missing Label Imputation

We use the same training dataset which we have used in the main text **Table 2**, containing two databases: 1) ZINC database with 310k molecules containing $Mol.Wt$, $LogP$, and QED, and 2) Materials Project electrolyte molecule database containing $Mol.Wt$, IE, and EA. We trained two different RNN-based ConGen models, one utilizing the mean and covariance table (E and C) imputation (the one used in the main text), and a second model where such imputation throughout the training is not performed. Afterward, we perform three different queries: **(1)** $Mol.Wt$ = 250 Da, $LogP$ = 2.5, IE = 5.0 eV, **(2)** $LogP$ = 2.5, IE = 5.0 eV, and **(3)** $Mol.Wt$ = 250 Da, $LogP$ = 2.5, EA = 4.0 eV. For each query task, we obtain 1000 molecules (3-5% of generated molecules are invalid) and validate the generated molecules' $Mol.Wt$ and $LogP$ using RDKit. IE and EA are not validated due to the computational costs, but these conditions are used for the conditional generation tasks in this section because the impact of imputation only manifests when there are insufficient valid samples for C calculation (such as $LogP$-IE pairs). From **Supplementary Table 1** below, we can see that imputation has relatively no impact on task **(1)**, positive impact on task **(2)**, and negative impact on task **(3)**. It is expected that imputation helps on task **(2)**, as there is almost no entry overlap to calculate the correlation matrix entry between $LogP$ and IE (the only two query constraints), making the C matrix with imputation statistically more meaningful. However, we are also able to find a situation where imputation is harmful, such as task **(3)** where the query condition EA = 4.0 eV is near the extreme end of the property label distribution. Future study should be conducted to determine the circumstances where imputation usage will be beneficial more systematically.

Query	RNN Model Type	$Mol.Wt$ (Da)	$LogP$
(1) $Mol.Wt$ = 250, $LogP$ = 2.5, IE = 5.0	Imputation	250 ± 4	2.54 ± 0.31
	No imputation	250 ± 4	2.53 ± 0.31
(2) $LogP$ = 2.5, IE = 5.0	**Imputation**	314 ± 90	**2.54 ± 0.40**
	No imputation	318 ± 94	2.44 ± 0.43
(3) $Mol.Wt$ = 250, $LogP$ = 2.5, EA = 4.0	Imputation	251 ± 8	2.97 ± 0.53
	No imputation	249 ± 8	**2.78 ± 0.50**

Supplementary Table 1 | Impact of missing label imputation on different RNN ConGen multi-constraint generation queries. Best model for each task query is bolded. For task **(1)**, the two models are equivalent.

Impact of ConGen BERT β Variations and Training from Scratch

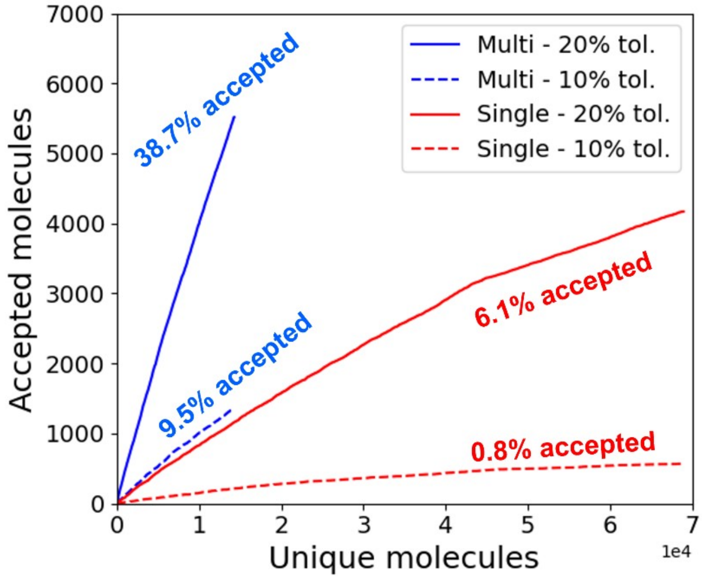
We use the same training dataset which we have used in the main text **Table 2**, to train BERT-type ConGen model (transferred ChemBERTa parameters used just like in the main text, with just the last layer being unfrozen). In the main text, we use β = 10000 for both RNN and BERT-based ConGen, the same setting used to train the original RNN-based SSVAE model but fail to see clear model improvement when BERT-based ConGen is used. Correspondingly, we no longer use BERT-based ConGen in the subsequent sections as it is computationally more expensive than RNN-based ConGen. In this supplementary section, we attempt different β settings (1000, 3000, 10000, 30000, and 100000) to see if we can get transfer learning BERT-based ConGen to clearly perform better than RNN-based ConGen. We also train a BERT-based ConGen from scratch with the original β = 10000 to see its impact on performance. From **Supplementary Table 2** below, we can see that β = 10000 seems to have the best ConGen BERT performance, while training the ConGen BERT from scratch instead of utilizing transfer learning produces a model with poor regression and conditional generation capabilities.

Model	β	$Mol.Wt$ (Da)	$LogP$	QED	EA (eV)	IE (eV)
Predictor MAE						
RNN	10000	2.70	0.05	0.009	0.20	0.16
BERT	1000	7.09	0.16	0.020	0.25	0.24
	3000	6.24	0.15	0.017	0.22	0.19
	10000	6.07	0.15	0.017	0.22	0.19
	10000 – fresh start	82.6	0.95	0.129	0.85	1.17
	30000	6.33	0.16	0.017	0.23	0.20
	100000	6.17	0.15	0.017	0.22	0.19
Conditional Generation						
RNN	10000	248 ± 4	2.55 ± 0.23	0.672 ± 0.082	2.06 ± 0.55	6.53 ± 0.62
BERT	1000	229 ± 36	2.55 ± 0.61	0.716 ± 0.071	2.26 ± 0.93	6.35 ± 0.45
	3000	209 ± 34	2.75 ± 0.64	0.751 ± 0.029	1.36 ± 0.50	6.48 ± 0.24
	10000	252 ± 3	2.45 ± 0.36	0.756 ± 0.127	1.80 ± 0.64	6.36 ± 0.41
	10000 – fresh start	249 ± 6	2.45 ± 0.64	0.659 ± 0.125	1.20 ± 1.15	6.64 ± 0.47
	30000	253 ± 9	3.01 ± 0.26	0.790 ± 0.075	1.62 ± 0.75	6.42 ± 0.36
	100000	252 ± 5	2.44 ± 0.46	0.668 ± 0.131	2.39 ± 0.98	6.54 ± 0.74

Supplementary Table 2 | Impact of β variation on BERT-based ConGen performance. We use the same conditional molecule generation with multiple constraints we use in **Table 2** of the main text: $Mol.Wt$ = 250 Da, $LogP$ = 2.5, and IE = 5.e eV.

ConGen Advantage on Multi-Condition Generative Design

It is instructive to contemplate whether enabling multi-condition generative model such as ConGen is useful or not, compared to single-condition generative models. We use the RNN-based ConGen model we have trained and utilized in the main text **Table 2**, with the intent of generating molecules which have the following simultaneous properties: $Mol.Wt$ = 250, $LogP$ = 2.5, and QED = 0.55. In the multi-constraint approach, we query the model 90,000 times with these simultaneous constraints. Out of these 90,000 molecules, 4,257 are within the training dataset and we generate a total of 14,628 unique molecules outside of the training dataset. In the single-constraint approach, we query the model 30,000 times for each of the single constraint. Out of these 90,000 molecules, 3,928 are within the training dataset and we generate a total of 68,925 unique molecules outside of the training dataset. Note that multi-constraint ConGen here generates less unique molecules compared to single-constraint version, but the generated molecule properties are more accurate as we will show below. Because it is impossible for the model to generate molecule with the same exact properties as our reference values, we need to decide on an acceptance tolerance criterion. If our tolerance criterion is 20% relative error, we obtain 5,518 and 4,170 acceptable molecules from the multi-constraint and single-constraint approach respectively, corresponding to 38.7**%** and 6.1**%** acceptance rate for the two approaches. If our tolerance criterion is 10% relative error (stricter), we only obtain 1,354 and 568 acceptable molecules from the multi-constraint and single-constraint approach respectively, corresponding to 9.5**%** and 0.8**%** acceptance rate for the two approaches. It is straightforward to see the user's tolerance on the generated molecule properties determine which approach should be used. If the user has very large tolerance (in extreme case, any molecule property error is accepted), then the single-constraint approach should be used because the model generates more diverse and unique molecules. However, if the user has strict requirements on the generated molecule properties, a multi-constraint approach will generate acceptable molecules with significantly higher acceptance rate. We demonstrate this result in **Supplementary Figure 1** below. It is important to note that the 10% and 20% relative error acceptance criterion we use here is simply for convenience purposes. In general, the user should specify the acceptance criterion which makes more sense for each molecule property in the user application.



Supplementary Figure 1 | Acceptance rate of multi-constraint ConGen vs single-constraint ConGen. The same generative model is used for both approaches, but the multi-constraint version is queried 90000 times using the 3 simultaneous constraints while the single-constraint version is queried 30000 times for each of the 3 single constraints. The single-constraint approach can generate more diverse and unique molecules because it is less restricted, but the acceptance rate of the generated molecule properties is much lower than the multi-constraint generative model approach. Both acceptance under the 20% tolerance and the stricter 10% tolerance criterions are shown.

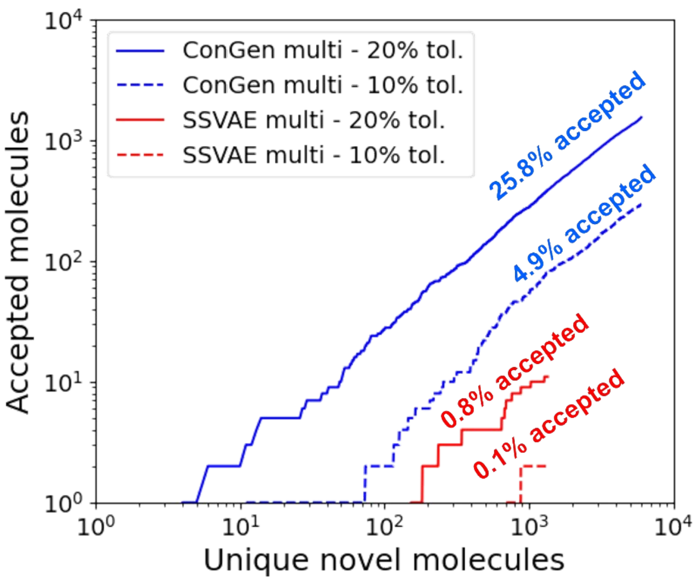
ConGen Advantage Compared to SSVAE on Incomplete-Labelled Dataset

In the main text, we have mentioned that the primary advantage of ConGen is that it can work with molecule databases with incomplete labels, which is not suitable for a baseline SSVAE model. Suppose we would like to utilize the SSVAE model for multi-condition generative modelling tasks using training molecule databases with incomplete labels regardless, for comparison purposes. We can perform this task with two different approaches on the SSVAE model before comparing its performance to a ConGen model trained on the same datasets:

1. Use all molecules with full labels as fully labelled training dataset and designate any molecules with incomplete labels as fully unlabelled training dataset.
2. Train individual SSVAE models with only single-property label each, ensuring that each SSVAE model can utilize all the training property labels.

In the first approach, we allow the SSVAE model to perform multi-constraint molecule generation, in exchange for a significant loss of property training data label information. It is impossible to test this approach on the main text's training datasets because of the extremely low availability of fully labelled molecules in our dataset. We have instead utilized the full ZINC database from the main text (containing 310,000 unique molecules) with fully labelled $Mol.Wt$, $LogP$, and QED properties (obtained using RDKit). 300,000 of these molecules are designated as the training dataset, while the remaining 10,000 molecules are designated as the test dataset. For each property label column in the training dataset, we randomly de-label 70% of the properties. Consequently, only ~2.7% of the molecules in our training dataset are fully labelled (8,117 fully labelled + 291,883 fully unlabelled molecules). This reflects the severe consequence of randomness we typically encounter from available experimental databases when multiple properties are needed. We train the SSVAE model of Kang, *et al* on this dataset, and perform a query to generate 10,000 molecules in multi-constraint mode ($Mol.Wt$ = 250, $LogP$ = 2.5, and QED = 0.55). Note that the original SSVAE model as published by Kang, *et al* only supports single-constraint molecule generation mode, and we have slightly modified the

SSVAE model's molecule generation function to enable multi-constraint generation mode the exact same way it is being done in ConGen. We also train the ConGen model on the same partially de-labelled training dataset (ConGen can utilize all ~70% remaining partial labels) and perform a query to generate 10,000 molecules with same multi-property constraints as specified above. Out of the 10,000 SSVAE molecules, 488 molecules are within the training dataset and we obtain 1,347 unique molecules outside of the training dataset. On the other hand, ConGen generates 239 molecules which are within the training dataset and 5,988 unique molecules outside of the training dataset. If our tolerance criterion is 20% relative error, we obtain 11 and 1,545 acceptable molecules from the SSVAE and ConGen approaches respectively, corresponding to 0.8**%** and 25.8**%** acceptance rate for the two approaches. If our tolerance criterion is 10% relative error (stricter), we only obtain 2 and 295 acceptable molecules from the SSVAE and ConGen approaches respectively, corresponding to 0.1**%** and 4.9**%** acceptance rate for the two approaches. See Supplementary **Figure 2** below for more details.



Supplementary Figure 2 | Acceptance rate of multi-constraint ConGen vs multi-constraint SSVAE. The same training dataset (ZINC database with $Mol.Wt$, $LogP$, and QED labels) is used for both approaches, and a query for generating 10,000 molecules with multi-property constraints ($Mol.Wt$ = 250 Da, $LogP$ = 2.5, and QED = 0.55) is performed using both models. However, the multi-constraint SSVAE cannot utilize labels from molecules which are only partially labelled. Because of the lack of fully labelled molecules, the multi-constraint SSVAE model suffers on both its ability to generate diverse molecules under multi-property constraints, and on its ability to generate molecules with the correct desired multi-property constraints. Both acceptance under the 20% tolerance and the stricter 10% tolerance criterions are shown.

In the second approach, we restrict the individual SSVAE model training to be done on only one property each. In this example, we can utilize the training dataset we have previously used in the main text **Table 2** and perform direct comparison with the molecules generated by the ConGen approach. We train three individual SSVAE models for each of the three property constraints (*Mol.Wt* = 250 Da, *LogP* = 2.5, and *IE* = 5 eV), and then generate 10 molecules for each SSVAE model's validation using the corresponding single-property constraint. Note that the original SSVAE model as published by Kang, *et al* will generate programming errors when trained on single properties (the code as published was trained on multiple properties and can only be used for single-property generation tasks), and some minor code reprogramming is needed to enable the model to work on single-property training and molecule generation tasks. The result is show in **Supplementary Table 3**, where we show that the individual SSVAE models separately trained on $Mol.Wt$ and IE have good control over the individual property of the molecule it generates, but surprisingly the model separately trained on the single $LogP$ property has bad performance on $LogP$ molecule generation task ($LogP$ = 2.35 ± 1.12). While the individual SSVAE models have good performance on the single property constraint it was trained on, they are not constrained on the other two properties they are not trained on. Consequently, these molecules generated by these single-constraint SSVAE's are not suitable for satisfying the requirement of multi-constraint generation queries, compared to the ConGen model.

RNN-based Model	$Mol.Wt$ (Da)	$LogP$	QED	EA (eV)	IE (eV)
ConGen	**248 ± 4**	**2.55 ± 0.23**	0.672 ± 0.082	2.06 ± 0.55	**6.53 ± 0.62**
SSVAE (MolWt = 250 Da)	**251 ± 7**	2.68 ± 0.82	0.780 ± 0.091	1.76 ± 0.62	6.14 ± 1.39
SSVAE (LogP = 2.5)	300 ± 82	**2.35 ± 1.12**	0.745 ± 0.149	2.04 ± 0.79	5.94 ± 0.82
SSVAE (IE = 5.0 eV)	322 ± 57	2.81 ± 1.07	0.716 ± 0.154	2.54 ± 0.78	**6.27 ± 0.50**
SSVAE (combined)	**291 ± 65**	**2.61 ± 1.03**	0.747 ± 0.137	2.11 ± 0.78	**6.12 ± 0.95**

Supplementary Table 3 | ConGen comparison with baseline SSVAE models trained using single properties on multi-constraint decoder conditional generation task. We use the same conditional molecule generation with multiple constraints we use in **Table 2** of the main text: $Mol.Wt$ = 250 Da, $LogP$ = 2.5, and IE = 5.e eV. The ConGen model is trained on these 3 property types simultaneously, while the SSVAE models are trained on individual property types. Note that while the generated molecules' IE magnitudes do not perfectly match the IE constraint (likely due to the difference in quantum chemistry workflow between ours and the online database), ConGen and SSVAE (IE) models both demonstrate tight IE control for the generated molecules.

Complete List of Molecules Generated in Main Text Query 1 (Figure 4b)

Supplementary Table 4 | Candidate Li-ion battery LHCE diluent molecules generated with multi-constraint ConGen model (Query 1).

['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 250, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 250, 4.0, 1.0]
Cc1ccc(OCC(F)(F)C(F)(F)Cl)cc1	CN(C)C(=O)Nc1cc(F)cc(C(F)(F)F)c1
Nc1c(F)cc(OC(F)(F)F)cc1CCl	CC(C)(C)OC(c1cc(F)cc(F)c1)C(F)F
CC(C)C(=O)Nc1ccc(F)c(C(F)(F)F)c1	CCOC(=N)c1c(F)cccc1CC(F)(F)F
CN(C)C(=O)Nc1c(F)c(F)cc(F)c1CF	Nc1cc(C(F)(F)F)ccc1OCCC(F)C
OCc1nc(C(F)(F)F)nc2c(F)cccc12	Fc1cc(OC(F)(F)F)cc(C2CCNC2)c1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 250, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 250, 4.0, 2.0]
CN(C)C(=O)OC1(C(F)(F)F)CCC(F)C1	Oc1cc(OC(F)(F)F)cc(F)c1CCl
CCOC(=O)c1cc(C(F)(F)F)nc(F)c1C	OC(OCC(F)(F)C(F)F)c1ccsc1
CC(=O)NCC(O)c1c(F)c(F)cc(F)c1F	COC(=O)CC(CC(F)(F)F)c1ccccc1
OCc1c(OC(F)(F)F)ccc(F)c1C1CC1	Cc1cc(OCC(F)(F)C(F)(F)CO)ccn1
COc1ccc(OCC(F)(F)C(F)F)c(C)c1	COC(=O)Cc1c(F)cnc(C(F)F)c1CF
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 250, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 250, 6.0, 1.0]
Oc1nc(F)cc(CC(F)(F)C(F)(F)F)n1	CC(OC(F)(F)F)c1ccccc1C(F)(F)F
FC(C(F)(F)F)C(F)(F)COC1CCCC1	Cc1ncc(OC(F)(F)F)nc1C(F)(F)F
FC(F)(F)C(F)(F)COc1ccccc1F	OC(CCC(F)(F)C(F)(F)C(F)F)C1CC1
OC(CCC(F)(C(F)(F)F)C(F)F)C1CC1	Fc1ccc(OCC(F)(F)F)cc1C(F)F
Cn1nc(C(F)(F)F)c(C(F)(F)F)c1CO	NC(c1ccoc1)C(C(F)(F)F)C(F)(F)F
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 250, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 250, 6.0, 2.0]
COC(=O)CCCC(F)(F)C(F)(F)C(F)F	OC(O)(CCCC(F)(F)F)CC(F)(F)F
OC(O)c1cc(C(F)(F)F)c(F)c(F)c1F	CC(=O)NCC(O)(C(F)(F)F)C(F)(F)F
Oc1cc(C(F)(F)F)cc(C(F)(F)F)c1O	CCOc1ccc(C(F)(F)F)c(C(F)(F)F)c1
COC(=O)CCC(F)(C(F)(F)F)C(F)(F)F	OC(OCC(F)(F)F)c1cccc(F)c1F
-- *invalid*--	Oc1c(OC(F)(F)F)cccc1C(F)(F)F
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 300, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 300, 4.0, 1.0]
Cc1noc(-c2ccc(C(F)(F)F)cc2Cl)c1C(F)F	Fc1cnc(OC(F)(F)F)c(I)c1
Cc1cc(OC(F)(F)F)cc(F)c1I	O=C(Nc1ncc(C(F)(F)F)cc1Cl)c1ccccc1
COc1cnc(C(F)(F)F)c(F)c1CBr	Cn1cc(-c2noc(-c3cc(F)c(F)c(F)c3F)n2)s1
Nc1c(F)cc(Oc2ccc(C(F)(F)F)cc2)c(Cl)c1	NC(=O)c1c(F)cccc1Nc1cc(C(F)(F)F)ccn1
COc1cnc(-c2ccc(C(F)(F)F)c(F)c2)c(Cl)c1	OC(Cc1ccc(C(F)(F)F)c(F)c1)c1cccs1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 300, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 300, 4.0, 2.0]
OC(Cc1cc(F)c(Br)cc1F)C(F)(F)O	NC(=O)COc1ccccc1-c1c(F)c(F)cc(F)c1F
COC(=O)CCc1ccc(Cl)cc1C(F)(F)C(F)F	OC(O)(c1cc(F)c(F)c(F)c1)c1ccc(F)cc1Cl
CCOC(=O)Nc1c(C(F)(F)F)cc(F)nc1CCl	NCc1cnc(OC(F)(F)F)nc1Oc1c(F)cccc1
N[C@@H](Cc1cc(F)c(F)c(F)c1F)c1ccc(O)cc1O	COc1ccc(CNc2cc(F)c(F)c(F)c2F)cc1O
CCC(=O)NCC(=O)Nc1cc(C(F)(F)F)cc(F)c1C	FC(F)(F)Oc1cccc(Oc2cc(F)cc(Cl)c2)c1

['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 300, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 300, 6.0, 1.0]
COC(c1cc(C(F)(F)F)nc(C(F)(F)F)c1)C1CC1 Nc1ccc(C(=O)NCC(F)(F)C(F)(F)C(F)F)cc1 NCc1cc(OC(F)(F)F)c(Cl)cc1C(F)(F)F NC(=O)c1cc(C(F)(F)F)cc(C(F)(F)F)c1CCl CC(C)C(=O)Nc1cc(C(F)(F)F)cc(C(F)(F)F)c1	COc1c(C(F)(F)F)ncc(C(F)(F)F)c1CCl OC(c1ccc(F)cc1)c1c(F)c(F)c(F)c(F)c1F Nc1cc(C(F)(F)F)cc(OC(F)(F)F)c1CCl FC(F)(F)c1cccc(-c2ccc(OC(F)(F)F)cc2)c1 Nc1c(F)cccc1Oc1cc(F)c(F)c(C(F)(F)F)c1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, 0.0, 300, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, 0.0, 300, 6.0, 2.0]
OCc1ccc(C(F)(F)F)cc1OCCC(F)(F)CF OB(O)c1c(C(F)(F)F)ccc(Cl)c1C(F)(F)F COc1ccc(OC(C(F)(F)F)C(F)(F)C(F)F)nc1 OCc1c(OC(F)(F)F)ncc(C(F)(F)F)c1C1CC1 CS(=O)(=O)Nc1cc(C(F)(F)F)cc(C(F)(F)F)c1	COc1cc(C(F)(F)F)c(C(F)(F)F)c(CC(N)=O)c1 Cc1cc(OC(F)(F)F)nc(OC(F)(F)F)c1CC#N Cc1ccc(COCC(F)(F)C(F)(F)C(F)F)cc1O O=C(O)Cc1cc(C(F)(F)F)nc(C(F)(F)F)c1CN OC(c1cccc(OC(F)(F)F)c1)c1ccc(F)cc1F
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 250, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 250, 4.0, 1.0]
Oc1ccc(-c2ccc(F)c(F)c2)c(F)c1F COC(c1c(F)c(F)nc(F)c1F)C1CCC1 NCc1cn(CC(F)(F)F)nc1CC(=O)F NCc1cc(OC(F)(F)F)cc(Cl)c1F Cn1cnc(OC(F)(F)F)c1-c1ccc(F)cc1	Cc1nc(-c2cccc(C(F)(F)F)c2F)c(C)o1 CC(NC(=O)C(F)(F)C(F)F)c1ccccc1 CC(CO)Nc1ccccc1C(F)(F)C(F)F CC(C)Oc1nc(C(F)(F)F)c(F)cc1CN OCC1Cc2cc(C(F)(F)F)cc(F)c2S1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 250, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 250, 4.0, 2.0]
COc1cc(C(F)(F)F)nc(OC)c1CCF CC(O)c1c(OC(F)(F)F)cc(F)cc1CN OCCC(=O)Nc1ccc(F)c(C(F)(F)F)c1 CC(N)(C(=O)O)c1nc(C(F)(F)F)ccc1F O=C(O)c1ccn(CCC(F)(F)C(F)F)c1F	OCC(O)Cc1ncc(C(F)(F)F)cc1CF O=C(O)CC(CC(F)(F)C(F)F)c1ccc[nH]1 CCC(NCC(F)(F)C(F)F)C(=O)OCC COc1ccc2c(F)c(F)c(F)c(F)c2c1O NCc1cc(OC(F)(F)F)c(O)c(CF)c1C
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 250, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 250, 6.0, 1.0]
OCc1nc(C(F)(F)F)c(C(F)(F)F)s1 OCCc1c(F)c(F)c(C(F)(F)F)c(F)c1 C[C@@H](O)c1c(F)c(F)c(C(F)(F)F)c(F)c1 OC(c1cc(F)cc(F)c1)C(F)(F)C(F)F CCc1c(OC(F)(F)F)n[nH]c1C(F)(F)F	N[C@@H](CO)c1c(F)c(F)c(F)c(F)c1CF CCc1ccc(OC(F)(F)F)c(C(F)(F)F)c1 Nc1cnc(OC(F)(F)F)c(C(F)(F)F)c1 OC[C@@H](c1cc(F)c(F)c(F)c1)C(F)(F)F COc1ncc(C(F)(F)F)c(C(F)(F)F)n1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 250, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 250, 6.0, 2.0]
C=C(O)CC(=O)C(C(F)(F)F)C(F)(F)CF OC(F)(F)C(F)(F)Oc1ccc(F)c(F)c1 OC[C@@H](O)CCC(F)(F)C(F)(F)C(F)(F)F OCCOCCC(F)(F)C(F)(F)C(F)(F)F OCc1c(O)c(F)c(F)c(F)c1C(F)(F)F	C[Si](C)(O)OC(F)(F)C(F)(F)C(F)(F)F COCC(O)CN(C(F)(F)F)C(F)(F)F CCOC(C)C(O)(C(F)(F)F)C(F)(F)CF O=C(O)CCCC(F)(F)C(F)CC(F)(F)F *--invalid--*

['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 300, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 300, 4.0, 1.0]
O=C(Cn1nc(C(F)(F)F)cc1Cl)c1ccccc1F CC1CCN(C(=O)Nc2cc(F)cc(C(F)(F)F)c2)CC1 OC(c1cc(F)cc(F)c1)c1ccc(C(F)(F)Cl)cc1 OC(c1cc(F)cc(F)c1)c1cnc(C(F)(F)Cl)cc1 CCc1nc(-c2ccc(OC(F)(F)F)cc2)nc(C)c1F	Cc1ccc(CC(=O)Nc2cc(F)cc(F)c2)c(F)c1F Cc1cc(C(F)(F)F)nc(Oc2cc(F)cc(Cl)c2)n1 CCCc1ncc(C(F)(F)F)c(Oc2ccc(F)cc2)n1 CC(OCC(F)(F)C(F)F)c1ccc(Cl)cc1Cl N#Cc1ccc(OCC(F)(F)C(F)F)cc1Br
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 300, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 300, 4.0, 2.0]
Cc1cc(OC(F)(F)C(F)(F)C(=O)NC2CC2)cs1 COC(=O)c1ncc(C(F)(F)F)c(F)c1Br COC(=O)Cc1nc(C(F)(F)F)c(F)cc1CCl CCOC(=O)Cc1cc(C(F)(F)F)cc(F)c1CCl FC(F)(F)Oc1cc(OC(F)F)cc(Br)n1	NC(=O)COc1ccc(C(F)(F)F)c(F)c1Br COC(=O)c1ccc(C(F)(F)F)c(-c2ccc(F)cc2)c1 FC(F)(F)Oc1ccc(OCc2ccncc2)c(F)c1C CCOc1cc(OC(F)(F)F)c(F)cc1Br N[C@@H](CC(=O)O)c1c(C(F)(F)F)cc(F)cc1CCl
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 300, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 300, 6.0, 1.0]
FC(F)C(F)(F)Oc1cc(C(F)(F)F)cnc1CCl OCc1c(C(F)(F)F)ccc(C(F)(F)F)c1CCl CCc1nc(OC(F)(F)F)c(C(F)(F)F)cc1CC#N OC(c1nc2ccccc2s1)C(F)(F)C(F)(F)CF Fc1ccc(-c2ccc(OC(F)(F)F)cc2)c(F)c1F	Nc1ccc(OCC(F)(F)C(F)(F)C(F)F)cc1C#N Oc1cc(F)c(-c2ccc(C(F)(F)F)cc2)c(F)c1F CCN(CC(F)(F)C(F)(F)C(F)(F)F)C(=O)NC1CC1 Fc1ccc(C(F)(F)F)c(Oc2cccc(F)c2)c1F Fc1cc(OC(F)(F)F)ccc1-c1ccc(F)c(F)c1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.0, -0.1, 300, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.0, -0.1, 300, 6.0, 2.0]
Cc1c(OC(F)(F)F)cnc(C(F)(F)F)c1CC(N)=O O=C(O)C(CC(F)(F)C(F)(F)C(F)F)c1ccccc1 OCc1cc(OC(F)(F)F)c(Cl)cc1C(F)(F)F C[C@@](N)(C(=O)O)c1nc(C(F)(F)F)c(C(F)(F)F)n1C CCOC(=O)c1cc(C(F)(F)F)ccc1C(F)(F)CF	Oc1ccc(COCC(F)(F)C(F)(F)C(F)F)cc1F OC(O)(Cc1ccc(C(F)(F)F)cc1)CC(F)(F)CF OC(O)(c1cc(C(F)(F)F)cc(C(F)(F)F)c1)C1CC1 O=C(NCC(F)(F)C(F)(F)C(F)F)c1ccc(O)cc1 FC(F)(F)COCCOc1c(F)cc(C(F)F)cc1N
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 250, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 250, 4.0, 1.0]
Cc1ccc(CNC(=O)C(F)(F)C(F)F)cc1 Cc1cc(CC(=O)NCC(F)(F)F)ccc1F O=C(c1cccnc1)c1cc(F)c(F)c(F)c1F FC(F)(F)c1cccc(Oc2ccccc2)c1F CCNC(=O)Nc1ccc(F)c(C(F)(F)F)c1	Cc1c(OC(F)(F)F)cc(F)cc1CCl OC(CC(F)(F)F)c1ccc(F)c(Cl)c1 COc1cc(C(F)(F)F)c(F)cc1CCl OCc1cnc(C(F)F)c(Cl)c1C(F)(F)F Nc1cc(OCC(F)(F)C(F)F)ccc1C#N
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 250, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 250, 4.0, 2.0]
OCCC(=O)Nc1c(F)cccc1C(F)(F)F O=C(Cc1ccc(OC(F)(F)F)cc1)C1CC1 CCOc1c(OC(F)(F)F)cc(F)cc1C#N CCOC1(C(F)(F)F)Oc2ccc(F)cc2C1 CC(C)CC(=O)NCC(O)CC(F)(F)C(F)F	OCc1cc(OCCC(F)(F)F)cc(F)c1C OCc1c(C(F)F)ncc(OC(F)F)c1CC COc1nc(OC(F)(F)F)c(F)cc1CC#N Cc1ccc(C(F)(F)C(F)(F)C(=O)O)cc1C Cc1nc(C(F)(F)F)c(CC(=O)O)cc1CF

['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 250, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 250, 6.0, 1.0]
O=C(NCC(F)(F)C(F)CF)CC(F)(F)F Cc1cnc(OC(F)(F)F)c(C(F)(F)F)c1 O=C(Nc1cc(F)cc(F)c1F)C(F)(F)F NCC(O)CCC(C(F)(F)F)C(F)(F)F Fc1cc(OCC(F)(F)F)ccc1C(F)F	OC(c1cc(F)c(F)c(F)c1)CC(F)(F)F FCOc1cc(C(F)(F)F)ccc1C(F)(F)F Cc1ncc(C(F)(F)F)c(OC(F)(F)F)n1 OC(c1c(F)c(F)c(F)c(F)c1F)C(F)F OCCc1c(F)c(F)c(C(F)(F)F)c(F)c1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 250, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 250, 6.0, 2.0]
O=C(O)CCCCC(F)(F)C(F)(F)C(F)F CCOC(=O)CC(C(F)(F)F)C(F)(F)CF Fc1cccc(OC(F)(F)OC(F)(F)F)c1 Oc1cc(OC(F)(F)F)cc(C(F)(F)F)c1 CCC(=O)OCC(C(F)(F)F)C(F)(F)CF	FC(F)(F)C1(C(F)(F)F)OCCCCOC1 COC(=O)CCC(C(F)(F)F)C(F)(F)CF Cc1c(O)cc(C(F)(F)F)cc1OC(F)(F)F O=C(OCCCCC(F)(F)F)CC(F)(F)F CC(O)CCC(O)(C(F)(F)F)C(F)(F)F
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 300, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 300, 4.0, 1.0]
O=C(NCC(F)(F)C(F)F)c1ccc(Cl)cc1Cl CCc1ncc(OC(F)(F)F)c(F)c1CBr FC(F)(F)c1ccc(Oc2ncccc2Cl)c(F)c1 Nc1ncc(F)cc1C(=O)Nc1ccc(C(F)(F)F)cc1 CCCc1nc(OC(F)(F)F)nc(F)c1Br	CN(C)c1cc(C(F)(F)F)nc(Oc2ccc(F)cc2)n1 OC(c1cccc(C(F)(F)F)c1)c1c(F)cccc1Cl Cc1cnc(C(F)(F)F)c(Oc2ccc(F)c(Cl)c2)n1 N#Cc1ccc(COc2ccccc2C(F)(F)F)c(F)c1 CNc1ccc(Oc2ccc(C(F)(F)F)nc2)c(F)c1C
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 300, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 300, 4.0, 2.0]
CC(Oc1cccc(C(F)(F)F)c1)C(=O)c1ccccc1 COCc1nc(OC(F)(F)F)c(F)cc1Br O=C(NCC(F)(F)CO)c1c(F)cc(F)cc1CCl FC(F)(F)Oc1cc(F)c(OCc2cccnc2)c(C)c1 CS(=O)(=O)Nc1ccc(C(F)(F)C(F)F)c(Cl)c1	CS(=O)(=O)N1CCN(c2c(F)c(F)cc(F)c2F)C1 Oc1cc(OC(F)(F)F)ccc1CBr CS(=O)(=O)Nc1ccc(SC(F)(F)C(F)F)cc1 O=S(=O)(c1cccc(C(F)(F)F)c1)c1ccc(F)cc1 CC(Nc1cc(C(F)(F)F)cc(F)c1)C(=O)OC(C)(C)C
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 300, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 300, 6.0, 1.0]
Cc1c(C(F)(F)F)cc(OC(F)(F)F)nc1CCl Nc1cc(OC(F)(F)F)cc(C(F)(F)F)c1CCl FC(F)(F)Oc1ccc(-c2ccc(C(F)(F)F)cc2)cc1 O=Cc1ccc(C(F)(F)F)c(C(F)(F)F)c1CCl --*invalid*--	Oc1c(F)cccc1-c1cc(C(F)(F)F)cc(F)c1F OC(c1cccc(C(F)(F)F)c1)c1cccc(F)c1F CC(=O)Nc1c(C(F)(F)F)cnc(C(F)(F)F)c1CN OCc1cc(C(F)(F)F)cc(C(F)(F)F)c1CCl OC(c1ccc(C(F)(F)F)nc1)c1c(F)cc(F)cc1F
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, 0.0, 300, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, 0.0, 300, 6.0, 2.0]
OCC(Oc1cccc(C(F)(F)F)c1)CC(F)(F)CF O=Cc1cc(OCC(F)(F)C(F)(F)C(F)F)ccc1N OCc1c(OC(F)(F)F)ncc(C(F)(F)F)c1Cl NC(COCC(F)(F)F)c1cccc(OC(F)(F)F)c1 OCc1ccccc1OCCC(F)(F)C(F)(F)C(F)F	C[C@@H](NC(=O)O)c1cc(C(F)(F)F)cc(C(F)(F)F)c1 CCOC(=O)Cc1c(F)cc(C(F)(F)F)cc1C(F)F NC(=O)c1c(OC(F)(F)F)cnc(C(F)(F)F)c1CN O=C(NCC(F)(F)F)N1CCC(O)(C(F)(F)F)CC1 FC(F)(F)Oc1ccc(OC(F)(F)C(F)Cl)cc1

['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 250, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 250, 4.0, 1.0]
CNCC(=O)Nc1cc(F)c(C(F)(F)F)cc1 OC(CC(F)(F)F)c1ccc(F)c(Cl)c1 Fc1ccc(COc2ccccc2F)c(F)c1F OCc1cc(C(F)(F)F)c(F)cc1CCl COc1c(C(F)(F)F)ccc(F)c1CCl	OC(c1cc(F)cc(C(F)(F)F)c1)C1CCC1 OCCc1cc(C(F)(F)F)c(Cl)cc1F OCCc1nc(C(F)(F)F)c(F)cc1Cl Nc1cnc(OCCCC(F)(F)F)c(F)c1F Nc1ccc(OCCC(F)(F)C(F)F)c(C)c1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 250, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 250, 4.0, 2.0]
CN(C(=O)O)c1cc(C(F)(F)F)cc(F)c1C CS(=O)(=O)CCSCC(F)(F)C(F)F CCC(=O)NCC(=O)NCC(F)(F)C(F)F CNC1CC(C(F)(F)C(F)(F)C(=O)O)CC1 COC(=O)Nc1cccc(C(F)(F)C(F)F)c1	CCCC(=O)OCCCCC(F)(F)C(F)(F)F O=Cc1cc(OCC(F)(F)C(F)F)cs1 OC(COCC(F)(F)F)c1ccc(F)c(C)c1 CCCOC(=O)Nc1c(F)c(F)cc(F)c1F CCOC(=O)Nc1nc(C(F)(F)F)ccc1F
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 250, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 250, 6.0, 1.0]
C[C@H](O)c1cc(C(F)(F)F)cc(C(F)(F)F)c1 OCC(F)(F)C(F)(F)c1cc(F)cc(F)c1 C[C@H](N)CC(=O)NC(C(F)(F)F)C(F)(F)F Cc1cc(OC(F)(F)F)nc(C(F)(F)F)c1 *--invalid--*	Cc1c(OC(F)(F)F)cccc1C(F)(F)F Nc1ncc(F)c(OC(F)(F)F)c1C(F)F CNCC(=O)NCC(F)(F)C(F)(F)C(F)F Fc1cc(F)c(OCCC(F)(F)F)c(F)c1 FC(F)(F)CCOc1cc(F)c(F)c(F)c1
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 250, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 250, 6.0, 2.0]
OCc1c(F)c(F)c(OC(F)(F)F)c(F)c1 C=CCOC(=O)CC(F)(F)C(F)(F)C(F)F OCc1cc(C(F)(F)F)cc(C(F)(F)F)c1O Cc1c(OC(F)(F)F)[nH]c(C(F)(F)F)c1O C=CCOC(=O)C(C(F)(F)F)C(F)(F)CF	COC(=O)CCC(C(F)(F)F)C(F)(F)CF CC(CC(F)(F)C(F)(F)C(F)F)CC(=O)O CC(C)(CC(F)(F)C(F)(F)C(F)F)C(=O)O Cc1ccc(OC(F)(F)F)c(OC(F)(F)F)c1 FCOc1c(C(F)(F)F)[nH]c(C(F)F)c1O
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 300, 4.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 300, 4.0, 1.0]
OC(c1c(F)c(F)c(F)c(F)c1Br)C1CC1 Oc1nc(F)c(C(F)(F)F)cc1I OC(Cc1ccc(F)c(Br)c1)CC(F)(F)F COc1nc(C(F)(F)F)ccc1CBr O=C(Cc1nc(C(F)(F)F)ns1)c1ccc(F)cc1C	OC(c1cccc(C(F)(F)F)c1)c1ccc(F)cc1Cl CC1CC(c2ccc(C(F)(F)F)cc2)NC(=O)NC1F CC(NC(=O)CC(F)(F)C(F)F)c1ccc(Cl)cc1 COc1cc(Br)cc(CC(F)(F)C(F)F)c1 Cn1nc(C(F)(F)F)cc1C(=O)Nc1ccc(F)cc1C
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 300, 4.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 300, 4.0, 2.0]
CC(=O)Nc1nc(-c2ccc(C(F)(F)F)c(F)c2)c(C)o1 Oc1cccc(Oc2c(F)c(F)c(F)c(F)c2Cl)c1C O=S(=O)(Cc1ccc(F)cc1)c1c(F)cc(F)cc1F OC(O)(c1c(F)cccc1F)c1cc(F)c(Cl)cc1F *--invalid--*	CC(Oc1cccc(C(F)(F)F)c1)c1cccc(F)c1O COc1nc(C(F)(F)F)ccc1OCc1ccc(F)cc1 OC(COCC(F)(F)C(F)F)Cc1ncccc1Cl FC(F)C(F)(F)Oc1ccccc1COc1ccccc1 COC(=O)c1ncc(C(F)(F)F)cc1-c1ccc(F)cc1

['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 300, 6.0, 1.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 300, 6.0, 1.0]
OC(C(F)(F)C(F)(F)C(F)(F)F)C(Cl)(Cl)Cl OCC(c1c(F)cc(F)cc1Cl)C(F)(F)C(F)(F)F FC(F)(F)Oc1ccnc(-c2ccc(C(F)(F)F)cc2)c1 CS(=O)c1cc(C(F)(F)F)cc(C(F)(F)F)c1C#N OC(F)(C(F)(F)F)C(F)(F)I	OCC(c1ccc(C(F)(F)F)c(C(F)(F)F)c1)C1CC1 FC(F)(F)Oc1cc(C(F)(F)F)c(Cl)cc1Cl Cc1ccc(OCC(F)(F)C(F)(F)C(F)(F)Cl)cc1 Fc1cc(OC(F)(F)F)ccc1-c1cc(F)cc(F)c1 *--invalid--*
['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.5, 7.5, -0.1, 300, 6.0, 2.0]	['EA', 'IE', 'LogVis', 'MolWt', 'n_F', 'n_O'] : [0.0, 7.5, -0.1, 300, 6.0, 2.0]
FC(F)(F)c1cc2c(cc1C(F)(F)F)OCCCCO2 CCCC(=O)OCCCCC(F)(F)C(F)(F)C(F)(F)F COC(=O)Cc1cc(C(F)(F)F)cc(C(F)(F)F)c1C O=C(Cc1cc(C(F)(F)F)cc(C(F)(F)F)c1)C(N)=O CCc1cc(C(F)(F)F)cc(C(F)(F)F)c1CC(=O)O	OC(Cc1ccc(OC(F)(F)F)cc1)CC(F)(F)CF OCc1ccc(COCC(F)(F)C(F)(F)C(F)F)cc1 Oc1cc(OC(F)(F)F)ccc1SCC(F)(F)F NC(=O)Cc1cc(C(F)(F)F)nc(OC(F)(F)F)c1C OC(c1c(F)c(F)c(F)c(F)c1F)c1ccc(F)cc1O