Digital Discovery

PAPER

Check for updates

Multi-constraint molecular generation using sparsely labelled training data for localized high-concentration electrolyte diluent screening†

Jonathan P. Mailoa, [ID] *[a] Xin Li,[a] Jiezhong Qiu[a] and Shengyu Zhang*[b]

Recently, machine learning methods have been used to propose molecules with desired properties, which is especially useful for exploring large chemical spaces efficiently. However, these methods rely on fully labelled training data, and are not practical in situations where molecules with multiple property constraints are required. There is often insufficient training data for all those properties from publicly available databases, especially when *ab initio* simulation or experimental property data is also desired for training the conditional molecular generative model. In this work, we show how to modify a semi-supervised variational auto-encoder (SSVAE) model which only works with fully labelled and fully unlabelled molecular property training data into the ConGen model, which also works on training data that have sparsely populated labels. We evaluate ConGen's performance in generating molecules with multiple constraints when trained on a dataset combined from multiple publicly available molecule property databases, and demonstrate an example application of building the virtual chemical space for potential lithium-ion battery localized high-concentration electrolyte (LHCE) diluents.

Introduction

Conditional molecular generation capability is a topic of strong interest for the purpose of chemical space exploration in the material virtual screening effort. Efforts in the field of conditional molecular generative model either takes no conditional constraint on the generation approach[1–5] or fail to introduce a cost function based on the generated molecules' property accuracy, making the models' generated molecular properties vary over a large range far from the desired property range.[6] This difficulty arises because in a model, molecular properties are typically the output of some regression model using the molecular structure as input. This makes it more challenging to use molecular properties as the input to conditionally constrain the chemical space of the generated molecules. Recent work based on reinforcement learning has enabled a conditional molecule generator which generates good molecular candidates after thousands of training iterations, assuming that a molecule property evaluator (cheminformatics library or computational material simulation tool) can continuously be utilized on the generated molecules during training.[7] The effectiveness of this reinforcement learning approach has also been demonstrated

through several other molecular generation design workflow based on reinforcement learning over the past few years.[8–13] In this work, we are interested in a specific practical task more commonly encountered in the virtual screening of chemical space relevant to industry: given a limited and often incomplete set of molecular property training labels from multiple sources, develop a generative model to generate a molecular chemical space which satisfies multiple property constraints so that it can be used as the high-quality input for a virtual screening pipeline in a low-cost and relatively accurate manner, without requiring additional simulations or experiments to further refine the generative model.

Recent work such as the semi-supervised variational auto encoder (SSVAE) model developed by Kang, *et al.*[14,15] which is based on foundational work by Kingma, *et al.*[16] solves a part of this problem by employing a dual-track architecture where the molecular property y is simultaneously the output from a molecule regression predictor sub-model and the input to a molecule generation decoder sub-model, in addition to also being the input for a separate molecule encoder sub-model. Because y is an output of the predictor sub-model, it can still be used to construct a useful cost function for the entire model even though it is also being used as the input to control the decoder's generated molecule structures. The resulting combined model has a relatively good control over the generated molecules' property, making it attractive for efficiently generating conditionally constrained molecular chemical space of interest. In addition to that, the SSVAE model is capable of utilizing both fully labelled molecules and fully unlabelled molecules during

[a]Tencent Quantum Laboratory, Tencent, Shenzhen, Guangdong, China. E-mail: jpmailoa@alum.mit.edu

[b]Tencent Quantum Laboratory, Tencent, Hong Kong SAR, China. E-mail: shengyzhang@tencent.com

the training process, making it somewhat attractive for practical usage as there are many cases where we have no access to the molecule properties due to a lack of simulation or experimental data. Nevertheless, the model is still impractical because in practice there are a lot of molecules where the data is only partially labelled and the SSVAE model is not equipped to handle such cases. A practical example of this problem is a situation in battery electrolyte molecule screening where 'easy' molecular properties such as molecular weight (Mol. Wt) and the number of fluorine atoms (n_F) are easily obtainable from cheminformatics libraries, while simulation data such as ionization energy (IE) and experimental data such as the viscosity (Log.·Vis, or the logarithm of viscosity) are not widely available. If we are interested in generating a chemical space satisfying several of these constraints, many of the molecules found in publicly available databases cannot be used as the fully labelled training data for the SSVAE model. Removing the labels completely and turning them into fully unlabelled SSVAE training data is detrimental as we then lose significant valuable label information from our training dataset.

In this work, we show how to enable a generative model which fully utilizes molecules with incomplete labels as the training data for a generative model without having to request additional training data label during training. This model improvement is enabled by modifying the SSVAE model to stop differentiating between fully labelled or unlabelled molecules. The model now relies on a molecular property mask instead, which tells the model which property can be used for training from a given molecule and which cannot. We name this modified SSVAE approach as the ConGen model, and the major modifications needed to enable these practical capabilities will be outlined in the next section. When the supplied molecule training data is either fully labelled or fully unlabelled, the ConGen model's data workflow will look identical to that of the SSVAE model's fully labelled and fully unlabelled data workflow. However, when the ConGen model is supplied with molecules with sparsely populated property labels as the training data, its components and cost functions are appropriately modified such that it only uses the relevant property labels based on the property mask. We first benchmark the usage of this model on a training dataset used by the original SSVAE model, which contains just labelled and unlabelled molecules. We then demonstrate several use cases which cannot be done using the SSVAE model, including the generation of virtual screening chemical space for lithium-ion battery localized high concentration electrolyte diluent (LHCE) candidates. This is achieved

by combining five publicly available molecular property databases, comprising different properties such as Mol. Wt, number of fluorine and oxygen atoms (n_F and n_O), ionization energy and electron affinity (IE and EA), and Log·Vis. The availability of these properties are very different, with the first three being fully available ('easy'), the next two with availability of approximately 3% ('medium' property, obtainable from quantum chemistry simulations), and the last one with availability of approximately 0.03% ('hard' property, obtainable from experimental measurements).

Baseline SSVAE model

We first describe the inner workings of the baseline SSVAE model developed by S. Kang, *et al.*,[14] which forms the foundation of this work. The main idea of the SSVAE model is simple:

(1) Encode the input molecule structure x from the training dataset into a latent space representation z using an encoder sub-model.

(2) Predict the property label of the input molecule structure x from the training dataset into predicted property y_P using a predictor sub-model. If an actual molecule property label y_L exists in the training database, y_P is discarded and the model uses the internal molecule property label $y = y_L$. Otherwise, $y = y_P$ is used.

(3) Use the internal molecule property label y and the latent space representation z as input to the decoder sub-model to generate the output molecule structure x_D.

In order to handle different types of training data (labelled *vs.* unlabelled), the SSVAE model treats the two types of data differently. The training dataset in an epoch's minibatch is split into two minibatches (labelled *vs.* unlabelled). The SSVAE workflow is then run twice, in a slightly different manner depending on whether the molecule minibatch is fully labelled or fully unlabelled (Fig. 1).

In SSVAE approach, the molecule input representation SMILES is converted into input embedding x using one-hot encoding. A molecule entry's training cost function needs to be split into three parts (eqn (1)–(3)). The cost function is written in verbose detail below for clarity, as we need to subsequently explain in the following section how the modifications need to be done for the dirty (partially labelled) data in the ConGen model:

(a) VAE cost function for completely labelled entries in the minibatch (eqn (1)):

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = &-\sum_{i=1}^{n_L}\sum_{j=1}^{n_x}\left(x_{i,j}\ln x_{D,i,j} + (1-x_{i,j})\ln(1-x_{D,i,j})\right) \\
&+ \sum_{i=1}^{n_L}\frac{1}{2}\left(n_y\ln 2\pi + \ln(\det(\boldsymbol{C})) + \sum_{j=1}^{n_y}(y_{L,i,j}-E_j)\sum_{k=1}^{n_y}(y_{L,i,k}-E_k)C_{k,j}^{-1}\right) \\
&- \sum_{i=1}^{n_L}\sum_{j=1}^{n_z}\frac{1}{2}\left(1+\ln\sigma(z_{i,j})^2 - \mu(z_{i,j})^2 - \sigma(z_{i,j})^2\right)
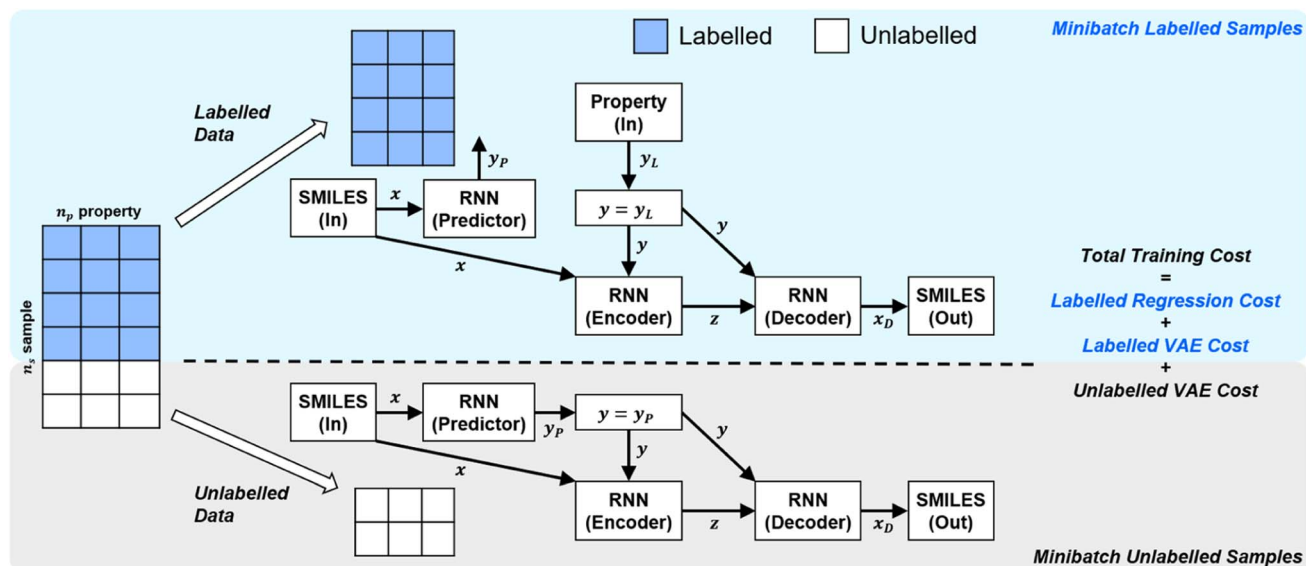\end{aligned}
\tag{1}
$$

Fig. 1 | High-level labelled/unlabelled data & model differentiation within Kang *et al.*'s original SSVAE model.[14] The variational auto-encoder (VAE) cost is calculated separately for the unlabelled and the labelled dataset, while regression cost is only calculated for the labelled dataset. The three costs are then summed up to calculate the total minibatch training cost.

(b) VAE cost function for completely unlabelled entries in the minibatch (eqn (2)):

$$
\begin{aligned}
\mathcal{U}(x) = &-\sum_{i=1}^{n_U}\sum_{j=1}^{n_x}\left(x_{i,j}\ln x_{D,i,j} + (1-x_{i,j})\ln(1-x_{D,i,j})\right) \\
&+\sum_{i=1}^{n_U}\frac{1}{2}\left(\sum_{j=1}^{n_y}C_{j,j}^{-1}\sigma(y_{P,i,j})^2 + \sum_{j=1}^{n_y}\left(\mu(y_{P,i,j})-E_j\right)\sum_{k=1}^{n_y}\left(\mu(y_{P,i,k})-E_k\right)C_{k,j}^{-1} - n_y + \ln(\det(C)) - \sum_{j=1}^{n_y}\ln\sigma(y_{P,i,j})^2\right) \\
&-\sum_{i=1}^{n_U}\sum_{j=1}^{n_z}\frac{1}{2}\left(1+\ln\sigma(z_{i,j})^2 - \mu(z_{i,j})^2 - \sigma(z_{i,j})^2\right)
\end{aligned} \quad (2)
$$

(c) Regression cost function for completely labelled entries (eqn (3)):

$$
\mathcal{R}_{SSVAE}(x,y) = \beta\sum_{i=1}^{n_L}\sum_{j=1}^{n_y}\left(y_{L,i,j}-\mu(y_{P,i,j})\right)^2 \quad (3)
$$

where $C = \text{Cov}(y_L)$ and $E = E(y_L)$ are the label covariance matrix and mean values constructed from the entire fully labelled training set, μ is the mean function, σ is the standard deviation function, β is the tradeoff hyperparameter between generative and supervised learning, while n_L, n_U, n_x, n_y, and n_z represent the number of minibatch's completely labelled entries, completely unlabelled entries, and dimensions of x, y, and z respectively. Finally, the total minibatch cost function is simply $\text{Cost}_{SSVAE} = \mathcal{L} + \mathcal{U} + \mathcal{R}_{SSVAE}$. Note that we use $\mu(y_{P,i,j})$ instead of $y_{P,i,j}$ in eqn (3), because the SSVAE predictor sub-model output $y_{P,i,j}$ is assumed to have a normal distribution.[14]

Finally, once the training is finished, the decoder sub-model can be extracted and be run independently by specifying the conditional property input y and the randomly sampled latent space input z to conditionally generate the desired molecule outputs. The beam search algorithm is used to efficiently convert the probabilities into the most likely output sequence x_D (based on breadth-first tree search mechanism), which is then easily converted to the output molecule SMILES.[14] The primary disadvantage of this approach is that the training dataset must be either fully labelled or fully unlabelled. The reason the SSVAE model splits the problem as specified in Fig. 1 above is because it simplifies the model dataflow, math, and behaviour tremendously. In practice, training datasets of interest likely consist of molecules with incomplete labels, in addition to the completely labelled or unlabelled molecules. This is especially so, if the training molecule database is either taken from a publicly available database (like PubChem experimental data[17]) or combined from several different databases. Neither of these practical types of "dirty" datasets will work for training the baseline SSVAE model, thereby severely limiting the type of conditional molecule generation which can be done, especially when multi-property conditional molecule generation is desired. This is typically the case for battery electrolyte or pharmaceutical drug molecule virtual screening.

Enabling sparse labelled data utilization using ConGen model

We modify the SSVAE model into the ConGen model, which is explicitly designed to work with "dirty" training data, thereby enabling the usage of significantly larger number of training data sources including those merged from different public and private sources. This enables us to perform conditional molecule generation tasks which are previously not possible using the SSVAE model. For example, given a large labelled molecule dataset from ZINC[18] (containing Mol. Wt, hydrophobicity LogP, and drug-likeness QED) and another similarly large molecule dataset from Materials Project Electrolyte Genome[19] (containing Mol. Wt, EA, and IP), we can train a conditional generative model which can generate molecules with multiple simultaneous constraints on the Mol. Wt, LogP, and EA values (known useful properties for screening lithium battery electrolytes). Given these diverse sources of training data, the original SSVAE model cannot be trained on the combined database of Mol. Wt, LogP, and EA labels because the training data label is sparse. ConGen on the other hand has no such limitation, allowing users to mix non-ideal practical data from multiple sources as desired.

The primary idea of the ConGen model is to take the general high-level architecture of the SSVAE model, but then modify all its components as needed in order to enable the usage of dirty training data. We have re-written the entire SSVAE model from the original TensorFlow 1.0 version into a PyTorch version to enable better model flexibility, before further implementing the necessary modifications to enable the usage of sparse training data labels. When this PyTorch version is trained on the original SSVAE training data (only fully labelled and fully unlabelled molecules) using the same hyperparameter training settings (n_{trn} = 285k training molecules with 50 : 50 labelled/unlabelled molecule split, n_{val} = 15k validation molecules, n_{tst} = 10 k test molecules, $\beta = 10^4$, Adam optimizer learning rate LR = 10^{-4}), we obtain accuracy metrics for property prediction, unconditional and single-property conditional molecule generation tasks (only Mol. Wt = 250 Da constraint is used, because the original SSVAE code only allows single-property constraint) equivalent to the

TensorFlow version (Table 1). 100 molecules are generated on both unconditional & conditional generation tasks.

Once we have confirmed that the two models are equivalent, the input data preprocessing and molecule data workflow inside the ConGen sub-models are modified (Fig. 2). First, we enable the ability to merge molecule training data labels with different types of property labels into a new property label matrix $\boldsymbol{y}_{\text{L}}$. This will cause a significant fraction of the merged database to contain missing [molecule, property] entry labels. For entries with no label available from all the databases, we designate the property label as invalid. This can be done by generating a mask matrix $\boldsymbol{M}$ containing '0' for invalid entries and '1' for entries with available property values. For entries where multiple property labels are available from different databases, we choose the available label from the latest database being merged. Both $\boldsymbol{y}_{\text{L}}$ and $\boldsymbol{M}$ matrices are now required as inputs into the ConGen model. ConGen no longer differentiates data workflow based on whether the molecule is fully labelled or fully unlabelled. ConGen instead implements a selector for the intermediate label $\boldsymbol{y}$ which choose whether to utilize existing label $\boldsymbol{y}_{\text{L}}$ or the predicted property label $\boldsymbol{y}_{\text{P}}$ generated by the predictor sub-model depending on the value of the mask $\boldsymbol{M}$ (eqn (4)):

$$\boldsymbol{y}(i,j) = \begin{cases} \boldsymbol{y}_{\text{L}}(i,j) & \text{if } \boldsymbol{M}(i,j) = 1 \\ \boldsymbol{y}_{\text{P}}(i,j) & \text{if } \boldsymbol{M}(i,j) = 0 \end{cases} \qquad (4)$$

where i and j denote the molecule and property type indices, respectively. With this modification, a unified data workflow can be utilized for fully labelled, fully unlabelled, and partially unlabelled molecules. Furthermore, when the molecule in the mini-batch is either fully labelled or fully unlabelled the mathematical operations performed on them within the ConGen model will be identical to those performed in the SSVAE model.

However, it is not as straightforward with respect to the training cost function and subsequent molecule generation. It is important to recognize that the implementation of the training cost function within the SSVAE model is heavily dependent on whether the molecule is fully labelled or fully unlabelled. The SSVAE cost function consists of three major elements, designed to ensure that the predictor, encoder, and decoder are all

Table 1 Comparison between SSVAE (TensorFlow 1.0) and baseline ConGen (PyTorch) model on the original SSVAE model tasks. The baseline ConGen is equivalent to SSVAE, except that it is implemented in PyTorch. This comparison is performed on SSVAE 'clean' original training dataset, which only contains fully labelled and fully unlabelled molecules. Identical training hyperparameter settings are used, and relatively equivalent performance metrics are obtained. The slight differences can be attributed to the high aggressivity of the original model's training hyper-parameter settings. For the property prediction task, predictor sub-model is utilized to calculate mean absolute error (MAE) with respect to the training labels. For the unconditional/conditional generation tasks, the decoder sub-model is used to generate the molecules and the molecules property labels are calculated using RDKit cheminformatics library

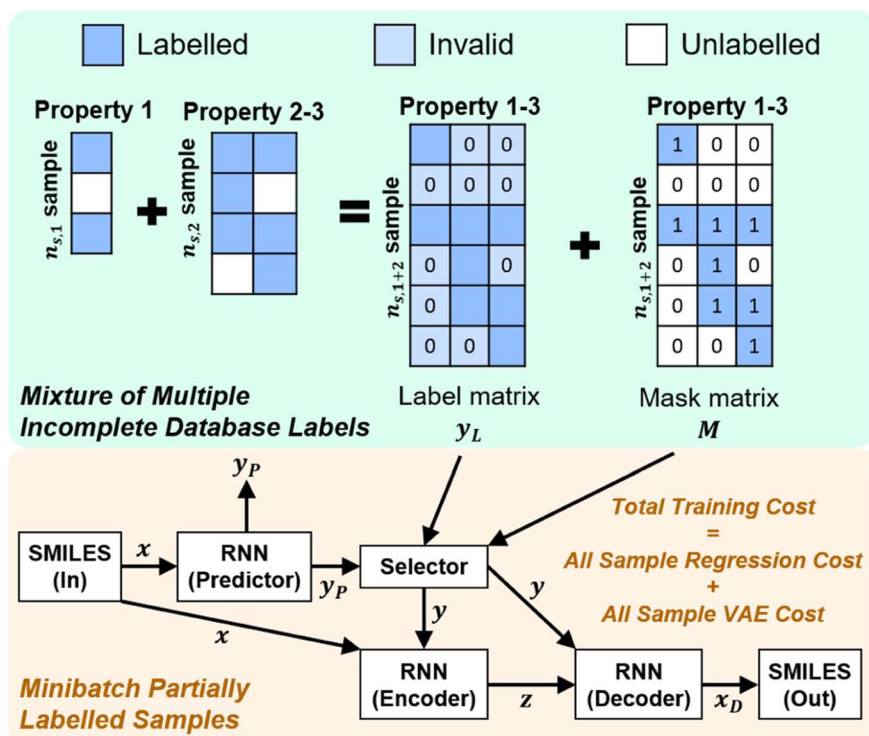| Task	Property	SSVAE	ConGen
Predictor regression MAE	Mol. Wt (Da)	0.95	1.22
	LogP	0.06	0.08
	QED	0.013	0.014
Decoder unconditional generation	Mol. Wt (Da)	360 ± 65	363 ± 64
	LogP	2.95 ± 1.06	3.01 ± 1.07
	QED	0.723 ± 0.142	0.713 ± 0.154
Decoder conditional generation	Mol. Wt (Da)	249 ± 6	251 ± 5
	LogP	2.38 ± 0.89	2.13 ± 0.91
	QED	0.810 ± 0.072	0.816 ± 0.095

Fig. 2 Dirty training label data merging and high-level dirty data workflow within the ConGen model. ConGen model no longer differentiates between fully labelled, fully unlabelled, and partially labelled molecule inputs. The unified data workflow is controlled by the mask matrix **M**. $n_{s,1}$, $n_{s,2}$, and $n_{s,1+2}$ denote the number of samples within the first, the second, and the merged property databases respectively.

accurate (eqn (1)–(3)) and we need to design the dirty data VAE cost function substitute for $\mathscr{L}$ and $\mathscr{U}$ because we no longer have fully labelled and fully unlabelled molecules. It is worth noting that during the execution of the original SSVAE model, there is no interaction between molecule inputs within a minibatch (*e.g.* if molecule A and B are processed simultaneously, the model output x_D for both molecules are not influenced by the fact that the other molecule is also simultaneously processed. This ensures that any intermediate values for a molecule (y_L, y_P, z, x_D, *etc.*) are solely determined by that molecule input x. Because of this, the implementation of a new cost function for the ConGen model becomes less complicated. There is a significant overlap of terms between $\mathscr{L}$ and $\mathscr{U}$, enabling us to design a new VAE cost function $\mathscr{G}$ for the ConGen model which takes partially labelled entries utilizing our mask matrix **M** (eqn (5)). When the entries are all completely labelled, the entries of **M** will all be 1, and $\mathscr{G}$

should be converted to $\mathscr{L}$, except for some constant terms that do not affect the training. When the entries are all completely unlabelled, the entries of **M** will all be 0, and $\mathscr{G}$ should be converted to $\mathscr{U}$, again, except for some constant terms. Similarly, our new regression cost function $\mathcal{R}_{\text{ConGen}}$ should only sum over labelled entries in the minibatch. By ensuring this behavior, the subsequent ConGen cost function differentiation and model parameter optimization will work exactly like the SSVAE versions when completely labelled/unlabelled data are supplied. However, it will also now work for dirty sparsely labelled training data. Henceforth, we define new cost functions for the ConGen minibatch, especially meant for dirty data:

(a) VAE cost function for dirty labelled entries in the minibatch (eqn (5)):

$$
\begin{aligned}
\mathcal{G}(\boldsymbol{x}, \boldsymbol{y}) = & -\sum_{i=1}^{n_S} \sum_{j=1}^{n_x} \left(x_{i,j} \ln x_{\mathrm{D},i,j} + (1 - x_{i,j}) \ln(1 - x_{\mathrm{D},i,j}) \right) \\
& + \sum_{i=1}^{n_S} \frac{1}{2} \left(n_y \ln 2\pi + \sum_{j=1}^{n_y} M_{i,j} (y_{i,j} - E_j) \sum_{k=1}^{n_y} M_{i,k} (y_{i,k} - E_k) C_{k,j}^{-1} \right) \\
& + \sum_{i=1}^{n_S} \frac{1}{2} \left(\sum_{j=1}^{n_y} C_{k,j}^{-1} (1 - M_{i,j}) \sigma(y_{i,j})^2 + \sum_{j=1}^{n_y} (1 - M_{i,j})(\mu(y_{i,j}) - E_j) \sum_{k=1}^{n_y} (1 - M_{i,k})(\mu(y_{i,k}) - E_k) C_{k,j}^{-1} - n_y - \sum_{j=1}^{n_y} (1 - M_{i,j}) \ln \sigma(y_{i,j})^2 \right) \\
& - \sum_{i=1}^{n_S} \sum_{j=1}^{n_z} \frac{1}{2} \left(1 + \ln \sigma(z_{i,j})^2 - \mu(z_{i,j})^2 - \sigma(z_{i,j})^2 \right)
\end{aligned}
$$

(5)

(b) Regression cost function for dirty labelled entries in the minibatch (eqn (6)):

$$\mathcal{R}_{\text{ConGen}}(\boldsymbol{x}, \boldsymbol{y}) = \beta \sum_{i=1}^{n_{\text{S}}} \sum_{j=1}^{n_{\text{y}}} M_{i,j} \left(y_{i,j} - \mu\left(y_{\text{P},i,j}\right) \right)^2 \qquad (6)$$

where n_s refers to the number of all samples in the dirty data minibatch. It is straightforward to prove that under this scheme, $\mathcal{G}$ is converted to either $\mathcal{L}$ or $\mathcal{U}$ depending on the values of $\boldsymbol{M}$, except for constant terms which do not have any impact on the model parameter optimization process. Note that, compared to the SSVAE cost functions, we have intentionally removed the constant terms $\ln(\det(\boldsymbol{C}))$ from the cost function above for numerical reasons we will describe in the following paragraph related to the dirty data covariance matrix $\boldsymbol{C}$. Crucially, under this new cost function only the corresponding labelled/unlabelled matrix elements from $\mathcal{L}$ and $\mathcal{U}$ contributes to the summation over n_s and n_y forming $\mathcal{G}$. The total minibatch cost function is now simply $\text{Cost}_{\text{ConGen}} = \mathcal{G} + \mathcal{R}_{\text{ConGen}}$.

It is important to note that because we only have partially labelled entries, we do not have complete entries for $\boldsymbol{y}_{\text{L}}$ and correspondingly $\boldsymbol{C} = \text{Cov}(\boldsymbol{y}_{\text{L}})$ and $\boldsymbol{E} = E(\boldsymbol{y}_{\text{L}})$ can only be calculated using the available labels from the partially-labelled entries, making these matrices ill-defined especially $\text{Cov}(\boldsymbol{y}_{\text{L}})$. For an SSVAE model, $\boldsymbol{C}$ is well-defined because it is straightforward to completely discard the unlabelled molecule entries from the training set and calculate $\boldsymbol{C}$ and $\boldsymbol{E}$ directly from the completely labelled molecule entries (this will be a good approximation as long as there is a large number of fully labelled molecules which is a good chemical representation of the full training dataset). This can be done once during the model construction and be set at those values throughout the entire model training. However, this strategy does not work for ConGen because the training data is dirty. In this case, it only makes sense to calculate the label mean $\boldsymbol{E}$ from the valid entries and ignore the invalid values in the $\boldsymbol{y}_{\text{L}}$ matrix. Similarly, it makes more sense to calculate covariance matrix $\boldsymbol{C}$ entries from the available $\boldsymbol{y}_{\text{L}}$ matrix entries while ignoring the invalid entries. In other words, we have the following situation for $\boldsymbol{E}$ and $\boldsymbol{C}$ calculation (eqn (7) and (8)):

$$E_j = E(\boldsymbol{y}_{\text{L}})_j = \frac{\sum_{i=1}^{n_{\text{s}}} y_{\text{L},i,j} M_{i,j}}{\sum_{i=1}^{n_{\text{s}}} M_{i,j}} \qquad (7)$$

$$C_{j,k} = \text{Cov}(\boldsymbol{y}_{\text{L}})_{j,k} = \frac{\sum_{i=1}^{n_{\text{s}}} \left(y_{\text{L},i,j} - E_j \right) \left(y_{\text{L},i,k} - E_k \right) M_{i,j} M_{i,k}}{\left(\sum_{i=1}^{n_{\text{s}}} M_{i,j} M_{i,k} \right) - 1} \qquad (8)$$

In a clean training data like the ones being used in the SSVAE model, all entries of the mask matrix $\boldsymbol{M}$ are all 1's, and it can then mathematically be proven that the covariance matrix $\boldsymbol{C}$ will always be a positive semi-definite (PSD) matrix. Correspondingly, in SSVAE the log-determinant term $\ln(\det(\boldsymbol{C}))$ in the cost function above will always be well-defined. The mathematical guarantee

breaks down when the entries of mask matrix $\boldsymbol{M}$ are no longer all 1's, however.[20] Consequently, we can get training errors due to attempting log operations on negative numbers. Nevertheless, because the term $\ln(\det(\boldsymbol{C}))$ is just a constant, we can remove it from the ConGen cost function without any mathematical training consequences as we have done in eqn (5).

The real physical issue arises from the quality of $\boldsymbol{E}$ and $\boldsymbol{C}$ themselves. When we have low availability of training data label (a lot of 0 entries in the mask matrix $\boldsymbol{M}$), we will have significant problems because the $\boldsymbol{E}$ and $\boldsymbol{C}$ matrices do not accurately represent the real molecule property labels, especially when we have many invalid labels in the training dataset. Keeping the values of $\boldsymbol{E}$ and $\boldsymbol{C}$ the same throughout the training iterations mean we will have poor control on the conditionally generated molecules' properties after subsequent model training and conditional generation processes. We can mitigate this problem by using imputation technique[20] to re-calculate $\boldsymbol{E}$ and $\boldsymbol{C}$ using predicted molecule property labels from the predictor sub-model when there is no valid label in $\boldsymbol{y}_{\text{L}}$. In other words, we track minibatch $\boldsymbol{y}$ from the selector (eqn (4), Fig. 2) throughout a training epoch, and re-calculate $\boldsymbol{E}$ and $\boldsymbol{C}$ using $\boldsymbol{y}$ instead of using $\boldsymbol{y}_{\text{L}}$ after each training and validation cycle in the epoch has been completed. This update is performed iteratively throughout the training, and it is important to store the final $\boldsymbol{E}$ and $\boldsymbol{C}$ as part of the ConGen model parameter because subsequent molecule generation tasks need to be performed using these higher quality $\boldsymbol{E}$ and $\boldsymbol{C}$ parameters (eqn (9) and (10)):

$$E_j = \boldsymbol{E}(\boldsymbol{y})_j = \frac{\sum_{i=1}^{n_{\text{a}}} y_{i,j}}{n_{\text{a}} - 1} \qquad (9)$$

$$C_{j,k} = \text{Cov}(y)_{j,k} = \frac{\sum_{i=1}^{n_{\text{a}}} \left(y_{i,j} - E_j \right) \left(y_{i,k} - E_k \right)}{n_{\text{a}} - 1} \qquad (10)$$

where n_{a} is the number of all molecules in the training dataset. The quality of $\boldsymbol{E}$ and $\boldsymbol{C}$ are not very good in the beginning of the training. However, as the predictor sub-model gets more accurate during subsequent training iterations, $\boldsymbol{E}$ and $\boldsymbol{C}$ will represent the real sample population better and we correspondingly achieve better molecule property prediction and conditional generation accuracy in the end. Further discussion on imputation impact can be found in ESI.†

We also take advantage of the modular nature of the ConGen model (inherited from the modularity of SSVAE) to further improve model performance on dataset with rare training property labels (such as *ab initio* simulation or experimental properties). It is straightforward to implement transfer learning in ConGen by replacing the recurrent neural networks (RNN) in the predictor and encoder sub-models with a bidirectional encoder representations from transformer (BERT) model pre-trained on a much larger (but 'cheaper') molecule property dataset. Here we use the ChemBERTa model, which is a large-scale self-supervised transformer-based pretraining model which only requires molecule SMILES as input and has been thoroughly evaluated.[21] During the sub-model construction, we add a fully connected network linear layer on top of the

Table 2 ConGen model performance comparison on 'dirty data' tasks, including both RNN-based ConGen and BERT-based ConGen. This 'dirty data' task cannot be done with the original SSVAE model but is useful in practice for conditional generative model training because molecule property labels are often unavailable or incomplete. Including a pre-trained BERT can increase the predictor sub-model's ability on 'rare' properties such as EA and IE, even though in some cases it may reduce the predictor sub-model's performance on 'common' properties (Mol. Wt, LogP, and QED in this case). The conditional generation co-constraints are Mol. Wt = 250 Da, LogP = 2.5, and IE = 5 eV. Regression MAE is calculated using property labels from the database, while generated molecules' properties are validated using either RDKit library or *ab initio* simulation. Out of the 372 210 unique molecules in the full combined database, we have 372 210 Mol. Wt, 310 000 LogP, 310 000 QED, 55 748 EA, and 52 346 IE labels available. However, only 11 unique molecules have all 5 property labels fully available

Task	Model	Mol. Wt (Da)	LogP	QED	EA (eV)	IE (eV)
Predictor regression test set MAE	RNN	2.70	0.05	0.009	0.20	0.16
	BERT	6.07	0.15	0.017	0.22	0.19
Decoder unconditional generation	RNN	312 ± 95	2.07 ± 1.28	0.677 ± 0.124	1.79 ± 0.84	5.99 ± 0.44
	BERT	271 ± 145	2.15 ± 1.11	0.583 ± 0.138	1.72 ± 0.82	6.40 ± 0.34
Decoder conditional generation	RNN	248 ± 4	2.55 ± 0.23	0.672 ± 0.082	2.06 ± 0.55	6.53 ± 0.62
	BERT	252 ± 3	2.45 ± 0.36	0.756 ± 0.127	1.80 ± 0.64	6.36 ± 0.41

transferred ChemBERTa model (in case of the decoder, to match its hidden representation vector size of x with its hidden representation vector size of y). We nickname this type of transferred model 'BERT' from here onward. When BERT is used to substitute the RNN encoder, the entire ChemBERTa layers' parameters are frozen. However, when BERT is used to substitute the RNN predictor, the last ChemBERTa layer's parameters can be fine-tuned by the PyTorch optimizer. While we do not substitute the RNN decoder with other types of decoder sub-model, in principle it is straightforward to do so as well if desired. For the standard ConGen model training with just RNN sub-models, we set the Adam optimizer $LR = 10^{-4}$ and clip the gradients absolute value to a maximum of 10^2. For the ConGen model training with BERT predictor and decoder sub-model substitutions, we have significantly lower Adam optimizer $LR = 3 \times 10^{-5}$ for the BERT-based sub-models, and $LR = 10^{-3}$ is used for optimizing the decoder sub-model parameters.

Finally, we demonstrate the resulting capability of the ConGen model on dirty dataset in Table 2. The training data labels are mixed from two different databases: (1) ZINC database containing properties such as Mol. Wt, LogP, and QED[18] used in the SSVAE publication,[14] (2) Materials Project Electrolyte Genome database containing properties such as Mol. Wt, IE and EA.[19] The ConGen model is trained on all 5 of these properties, which cannot be done by the SSVAE model. As an example of multi-property conditional generation, we query the models to generate molecules with 3 simultaneous properties: Mol. Wt = 250 Da, LogP = 2.5, and IE = 5 eV. The corresponding regression and conditional generation results are given below in Table 2. We validate the properties of the generated molecules (10 molecules for each model and generation task) using RDKit[22] (for Mol. Wt and LogP) and quantum chemistry (for IE, see Methods). Further discussion on the benefit of multi-constraint conditional generative model over single-constraint conditional generative model, as well as additional comparisons with the baseline SSVAE model, can be found in the ESI.† We see that overall, the BERT-based ConGen has worse performance than the RNN-based ConGen model on property prediction tasks but is relatively equivalent to the RNN-based ConGen on conditional generation tasks (good on Mol. Wt

and LogP, but less accurate on IE). We have expected the transferred BERT-based ConGen to perform worse than the RNN-based ConGen on abundant property label such as Mol. Wt and LogP (simple properties to learn) and better than RNN-based ConGen on rare property label such as IE (complex property to learn). We expected this outcome because in general, training a model from scratch is advantageous when enough training data is available while a pre-trained model performs better when there is insufficient training data. The fact that both RNN and BERT-based ConGen shows relatively equivalent performance for molecular conditional generation tasks merits further future investigation. We note that re-training the BERT-based ConGen from scratch significantly hurts its property prediction performance, although its conditional generation capability on LogP is still better than one of our less-optimized RNN-based ConGen (see ESI†). We hypothesize that we still have insufficient number of quantum chemistry property training labels from just the Materials Project Electrolyte Genome database,[19] and that a more accurate and data-efficient predictor sub-model is still needed. Currently the BERT-based ConGen is computationally more expensive while offering no significant improvement over the RNN-based ConGen, so we focus solely on using RNN-based ConGen in the following large-scale electrolyte diluent screening section. We note however, that the molecules generated by BERT-based ConGen has slightly better Synthetic Accessibility Score[23] (SA score = 2.42 ± 0.63) compared to the ones generated by RNN-based ConGen (SA score = 2.52 ± 0.70). This is likely because the ChemBERTa model is pre-trained on molecules which have previously been shown in literature, making it more likely that these molecules are more synthesizable.

Use case example: lithium–ion battery localized high concentration electrolyte diluent screening

Finally, we demonstrate the usage of the ConGen model on a practical example: generating the chemical space for further virtual screening of Li-ion battery localized high concentration electrolyte (LHCE) diluent molecules. Recent progress in the

development of Li-ion battery electrolytes have led to the discovery of LHCE-type of electrolytes, which microscopically look like that of high salt concentration electrolyte (HCE), but macroscopically look more like a conventional electrolyte.[24] The LHCE is useful because it is stable over a wide electrochemical window, in addition to forming stable solid electrolyte interphase (SEI) layer which is important for the long-term stability of the battery.[25,26] From a cost perspective, the LHCE is also important because it can reduce the required amount of Li-salt used, *versus* that of HCE which requires a large amount of expensive Li-salt.[27] Finally, LHCE can have significantly lower solution viscosity than HCE, which is useful not just for improving the electrolyte's lithium ion transport properties, but also for enabling better electrode wetting which helps to better optimize the energy capacity of Li-ion battery cells.[28] Chemically, what differentiates LHCE from HCE and conventional electrolytes is the addition of small molecules which act as a diluent in the electrolyte.[24] These diluent molecules are typically hydrofluoroether (HFE) such as bis(2,2,2-trifluoroethyl) ether (BTFE) and 1,1,2,2-tetrafluoroethyl-2,2,3,3-tetrafluoropropyl ether (TTE), or fluorinated orthoformate such as tris(2,2,2-trifluoroethyl) orthoformate (TFEO).[24–26] The unique trait of these types of compound is that while they are sufficiently polar, they are less polar than the Li-salt anions being used in the LHCE. Consequently, at the right concentration range the Li$^+$ cations will primarily coordinate with the polar salt anions in the first Li$^+$ solvation shell. The diluent molecules will then mostly coordinate with these salt clusters from the second shell onward using their polar oxygen atoms. Furthermore, the fluorinated components of the diluent molecules will tend to form their own non-polar network in the LHCE. Consequently, the addition of diluents into LHCE ensures that locally the salt cluster looks like that of HCE (more stable), while macroscopically the diluents separate these salt clusters and ensure that the solution is less viscous, ionically conductive, and ideally inflammable (due to the proportion reduction of flammable solvent molecules in LHCE).

Many criteria need to be satisfied by these LHCE diluent molecules such as electrochemical stability, inflammability, and low viscosity. While there are several known working LHCE diluents, it is important to find more relevant compounds in this field to enrich the diluent chemical space suitable for the optimization of specific types of Li-ion batteries. We apply the ConGen model to generate candidate molecules for LHCE diluents through structural chemical properties such as: Mol. Wt, n_F, n_O, IE, EA, and Log. Vis. To achieve this, we train ConGen model on a mixture of 5 publicly available datasets:

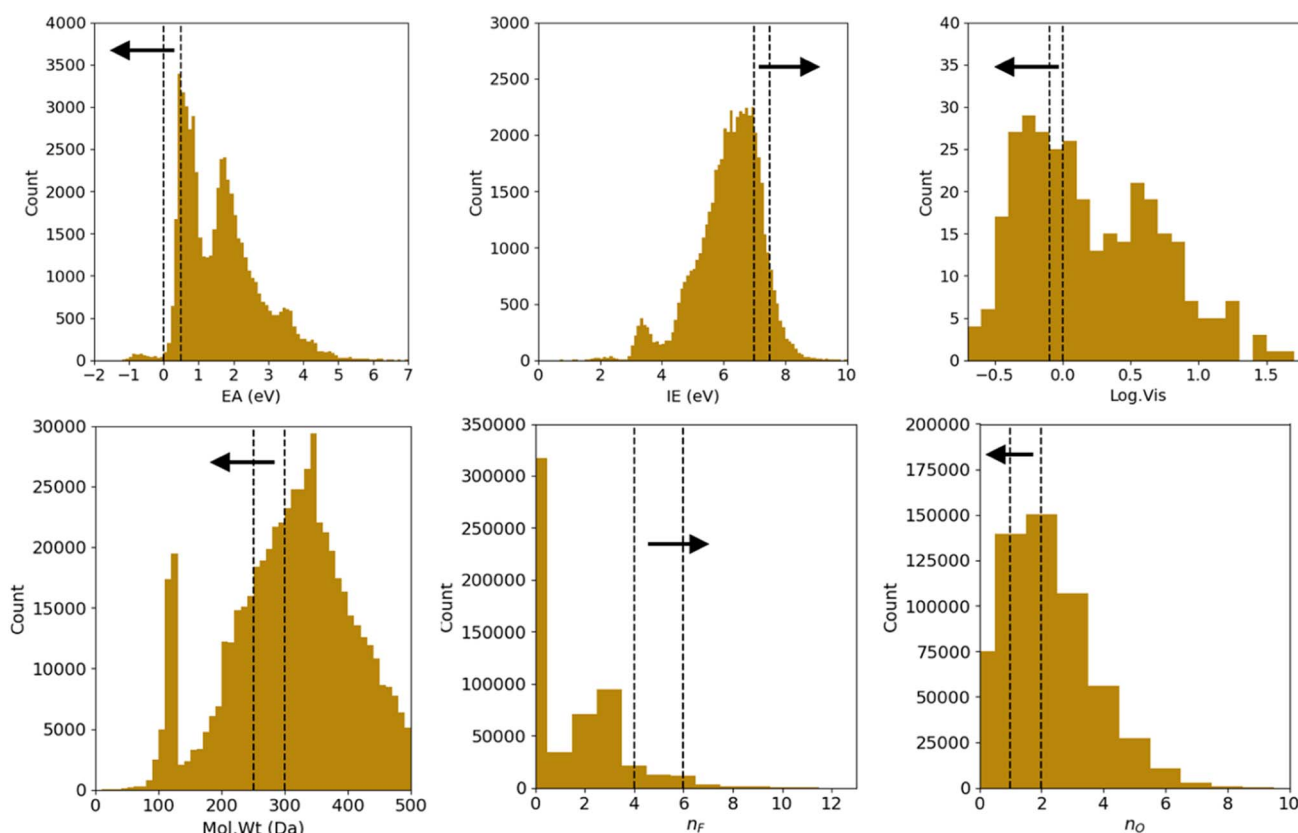● Mol. Wt database from ZINC[14,18] (310 000 unique entries)

Fig. 3 Training data molecular property label distribution. The dashed lines indicate the property label 'anchors' we will use for subsequent conditional molecular generation. The arrows indicate the preferred generated molecules' property range. The anchors are respectively: EA = [0, 0.5], IE = [7.0, 7.5], Log. Vis = [−0.1, 0.0], Mol. Wt = [250 300], n_F = [4, 6], n_O = [1, 2].

● Mol. Wt, simulated IE, EA database from the Materials Project Electrolyte Genome[19] (62 274 unique entries)

● Mol. Wt, simulated IE, EA database from Austin Apple Github[29] (26 394 unique entries)

● Oxyfluorocarbon n_F, n_O database from PubChem[17] (200 000 unique entries)

● Experimental Log. Vis database from literature[30] (322 unique entries)

Where applicable, each of these databases are supplemented with the corresponding molecule Mol. Wt, n_F, and n_O missing property labels because it is computationally efficient and inexpensive to do so using RDKit.[22] The combined database has 571 023 unique molecules. Finally, we evaluate the model's performance. Based on known existing LHCE diluents, we hypothesize that we need the following properties for the LHCE diluent molecules:

● Electrochemical properties: EA ≤ 0.5 eV, IE ≥ 7.0 eV

● Viscosity property: Log. Vis ≤ 0.0

● Structural properties: Mol. Wt ≤ 300, n_F ≥ 4, n_O = 1–2

Within the framework of ConGen, we can implement this multi-condition molecular structure generation task by simply deploying simultaneous property label 'anchors' as the decoder input during the generation cycle. For example, we may choose the following label anchors to satisfy the conditions stated above:

(1) EA = 0 or 0.5 eV
(2) IE = 7.0 or 7.5 eV
(3) Log. Vis = −0.1 or 0.0
(4) Mol. Wt = 250 or 300 Da
(5) n_F = 4 or 6
(6) n_O = 1 or 2

We correspondingly have $2^6 = 64$ combinations of multi-constraint property anchors we can use for the conditional generation in the example above. For each set of anchors, we generate 5 molecule samples resulting in 320 conditionally sampled molecules using our RNN-based ConGen model (Query 1). The training data label distributions, based on just available property labels, is shown below in Fig. 3.

Regression on the test set, unconditional molecule generation, as well as conditional molecule generation
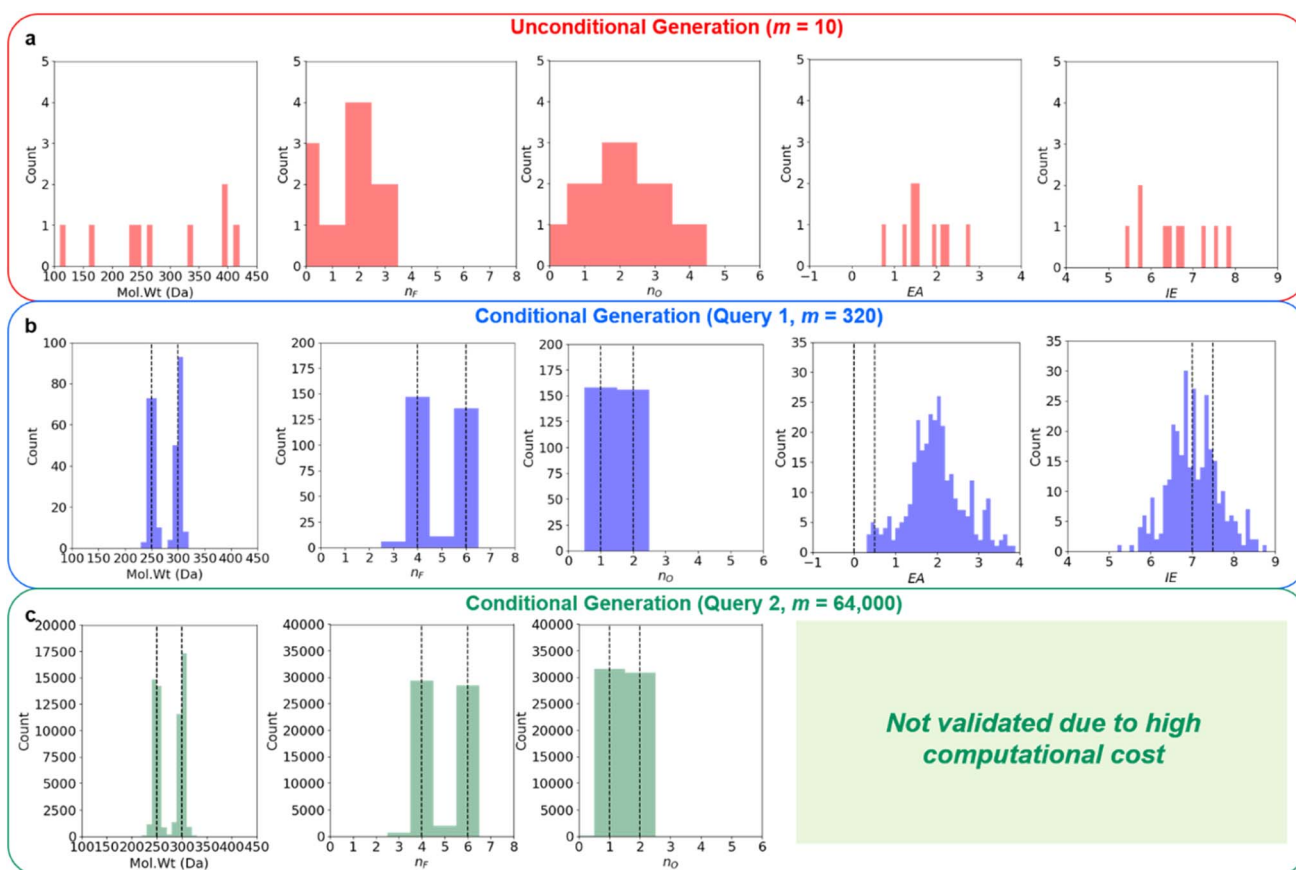

Fig. 4 ConGen unconditional and multi-constraint conditional molecular generation property distributions. (a) Unconditional molecule generation showing property distribution without constraints. When multiple co-constraints are utilized for the conditional generation, we have very targeted molecule generation. Structural and electrochemical stability properties validation for Query 1 with 320 molecules is shown in (b), while structural property validation for Query 2 with 64k molecules is shown in (c). We can see that the molecules generated with simultaneous multi-property constraints still obey their conditional property anchors quite well (simultaneously, although the hardest property EA distribution is slightly shifted), and that the generated molecules' property distribution is very different from molecules generated with no property constraint. Conditional generation property anchor inputs are shown as dashed lines.

results are shown below in Fig. 4 and Table 3. In order to calculate the ground truth property label values for the generated molecules, several methods are employed. For Mol. Wt, n_F, and n_O, simple cheminformatics tool such as RDKit can be used to quickly calculate their true values. For EA and IE, we used quantum chemistry calculations with identical calculation settings to the prior work[19] to calculate the true values. We see that we have excellent control over the generated molecules' structural properties (Mol. Wt, n_F, and n_O) and IE, although we observe a positive shift of approximately 2.0 eV on the generated molecules' EA compared to the mean of the anchors' EA (0.25 eV). We hypothesize that this systemic shift may be caused by the slight difference in our adiabatic EA calculation workflow compared to the procedure utilized by the Materials Project Electrolyte Genome team, as well as the fact that we query the ConGen model to generate molecules with EA label anchors at the extreme left end of the training dataset EA label distribution (making this the most difficult constraint out of the 6 co-constraints we have employed).

We currently have no experimental validation capability to measure Log. Vis for the generated molecules, so unfortunately no accuracy metric can be displayed for these molecules' Log. Vis property. Nevertheless, we have listed all the molecules that

Table 3 Molecular property prediction accuracy and the generated molecule's property distribution statistics for LHCE diluent molecules. From the regression test result, we can see the predictor sub-model is reasonably accurate in predicting molecular property. In addition to that, the discrepancy in distributed molecules' properties for unconditional *vs.* conditional generation cases show that the conditional generator is generating the right molecules, based on the property label input anchors we have chosen. Regression MAE is calculated using property labels from the database, while generated molecules' properties are validated using either RDKit library or *ab initio* simulation. Out of the 571 023 unique molecules in the full combined database, we have 570 946 Mol. Wt, 571 023 n_F, 571 023 n_O, 55 829 EA, 52 425 IE, and 320 Log. Vis labels available. However, only 105 unique molecules have all 6 property labels fully available

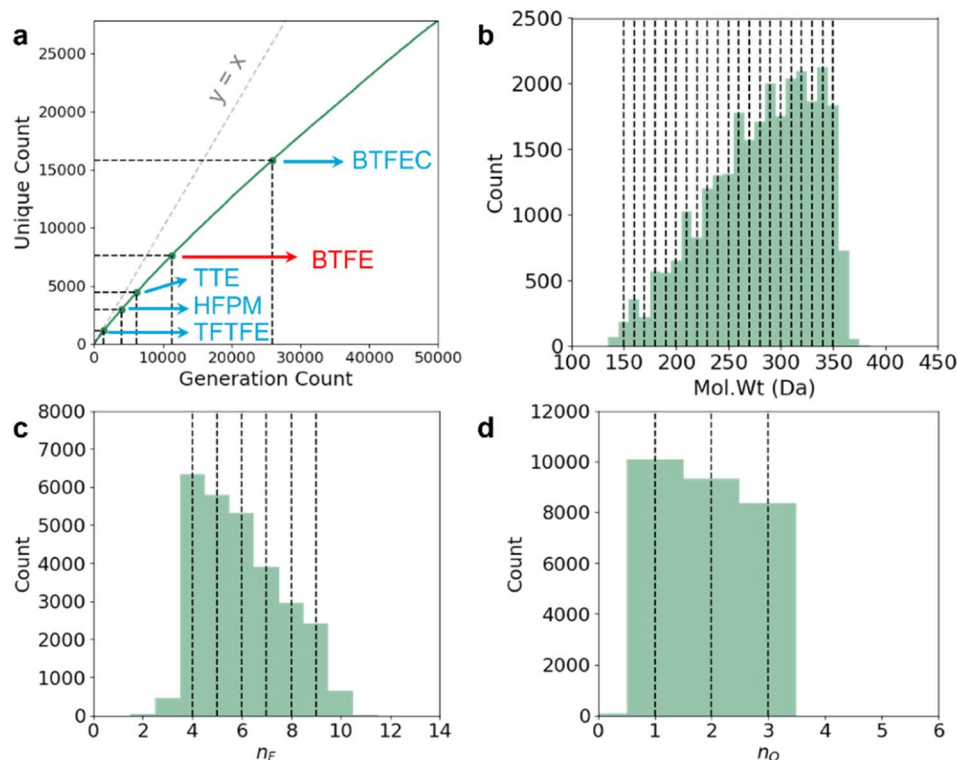| Task	Mol. Wt (Da)	n_F	n_O	EA (eV)	IE (eV)	Log. Vis
Predictor regression test set MAE	1.60	0.01	0.02	0.20	0.21	0.14
Decoder unconditional generation	302 ± 110	1.50 ± 1.12	2.30 ± 1.42	1.71 ± 0.54	6.58 ± 0.75	N/A
Decoder conditional generation (query 1)	275 ± 26	5.02 ± 1.08	1.50 ± 0.50	1.99 ± 0.73	7.04 ± 0.61	N/A
Decoder conditional generation (query 2)	274 ± 26	5.02 ± 1.05	1.49 ± 0.50	N/A	N/A	N/A

Fig. 5 Leave-out cross validation to re-discover experimentally proven LHCE diluent molecules. (a) Out of 50 000 queries, RNN-based ConGen generates 27 838 unique candidate LHCE diluent molecules and re-discovers known diluents such as TFTFE, HFPM, TTE, BTFE, and BFTEC. Out of these 5 diluents, BTFE (red) is within the training dataset while the other 4 (blue) are not. The $y = x$ dotted line is shown for clarity. (b–d) The Mol. Wt, n_F, and n_O distribution of the unique molecules generated by ConGen. The ConGen property anchors we used are marked with dotted lines. EA, IE, and Log. Vis of these molecules are not validated due to computational and experimental constraints.

Table 4 Top 35 candidates of Li-ion battery LHCE diluent molecules. These molecules are selected out of 49 032 unique LHCE diluent candidate molecule SMILES generated by the ConGen model throughout this work (including those which exist in the training databases) based on synthesizability and novelty criteria

O=S(=O)(c1ccc(F)-
c(F)c1)c1c(F)cccc1F-
SA score = 2.06

Oc1ccc2c(c1)OC(F)(F)O2
SA score = 2.49

Oc1c(F)c(F)c(F)-
c(–c2ccncc2)c1F
SA score = 2.46

O=C(CSc1nccs1)NCC(F)(F)F
SA score = 2.46

O=C(Nc1c(F)c(F)nc(F)c1F)
C1CCCC1
SA score = 2.44

CC(C)C(=O)Nc1cc(C(F)(F)F)
cc(S(=O)(=O)C(F)F)c1
SA score = 2.40

FC(F)(F)CCOCC1CC1
SA score = 2.37

FC(F)(F)C(F)(F)C(F)(F)
OCc1ccc(Cl)cc1
SA score = 2.37

Cc1ccc2oc(–c3cc(F)c(F)c(F)c3F)
nc2c1 SA score = 2.34

Nc1cc(F)cc(Oc2cc(F)cc
(F)c2F)c1
SA score = 2.29

CCOc1cc(OC(F)(F)F)-
cc(OC(F)F)c1
SA score = 2.23

CCOC(=O)C(F)(F)F
SA score = 2.05

FC(F)Oc1ccc(OC(F)F)cc1
SA score = 1.89

O=C(NCC(F)(F)F)c1ccccc1
SA score = 1.59

O=C(NC1CC1)C(F)(F)F
SA score = 2.14

O=C(Nc1ccc(C(F)(F)F)cc1)
Nc1ccc(F)cc1
SA score = 1.62

FC(F)(F)c1ccc (Oc2ccc-
(C(F)(F)F)cc2)cc1
SA score = 1.63

FC(F)(F)C(F)(C(F)(F)F)
-N1CCOCC1
SA score = 2.97

CS(=O)(=O)NC
C(F)(F)C(F)(F)C(C)(F)F
SA score = 2.98

FC(F)COCCOC(F)F
SA score = 2.99

C[C@H]1O[C@H]-
(c2ccc(F)c(F)c2)C1
SA score = 2.82

COCCCNc1nnc-
(C(F)(F)C(F)(F)F)s1
SA score = 2.83

OC(O)C(F)(F)C(F)(F)-
C(F)(F)C(F)(F)F
SA score = 2.88

OCCN(CC(F)(F)F)CCCC(F)(F)F
SA score = 2.64

O=Cc1ccc(C(F)(F)-
C(F)(F)C(F)(F)Cl)cc1
SA score = 2.63

O=S(=O)
-(Oc1c(F)cc(F)cc1F)
c1c(F)cc(F)cc1CF
SA score = 2.78

CC(C)(C)c1nc2
c(F)c(F)c(F)c(F)c2o1
SA score = 2.94

COC(=O)c1c(F)-
c(F)c(F)c(F)c1F
SA score = 2.27

Fc1c(F)nc(OC2CCCCC2)-
c(F)c1F SA score = 2.67

O=C(c1cccnc1)-
N1C C(F)(F)C(F)(F)C1
SA score = 2.86

O=C(OC(=O)-
C(F)(F)C(F)(F)F)
CC(F)(F)F
SA score = 2.95

CC(C)(C)OC(C(F)(F)F)
-C(F)(F)F
SA score = 2.93

Cc1c(C(F)(F)F)-
c[nH]c(=O) c1C(F)(F)F
SA score = 2.90

Oc1c(C(F)(F)C(F)(F)C(F)(F)
C(F)(F)F)ccc2c1CCC2
SA score = 2.89

CC(C)CNC(=O)-
c1c(F)c(F)c(Br)c(F)c1F
SA score = 2.50

the ConGen model has generated based on their property label input anchors in ESI† for future validation by other research groups with experimental capabilities. Additional molecular property criteria are likely needed to further improve the quality of the generated LHCE diluent candidate molecules. Inclusion of further molecular property constraints to help refine this generated LHCE diluent chemical space further should be straightforward, as it can be done by simply adding a new comma-separated-value (CSV) file containing the desired molecular properties for training. Out of the 320 generated molecule SMILES, 6 are invalid molecules, 3 are duplicates, and 5 are within the training set. We have correspondingly generated 306 new unique candidate molecules from this query for computational validation purposes. We further generate 64 000 candidate molecules using the RNN-based ConGen model (1000 queries for each of the anchor combinations, see Fig. 4) although neither EA nor IE *ab initio* computational validation is done for these additional molecules due to the high computation costs (Query 2). Out of this new query for 64 000 molecules, 1486 are invalid, 41 117 are duplicates, and 356 are within the training set. Correspondingly, Query 2 generates 21 041 new unique candidate LHCE diluent molecules. Future work is needed to reduce the number of large-scale-query duplicates.

Following the suggestion we receive during the peer review process, we also perform a leave-out cross-validation experiment to see if our model can generate any known LHCE diluent molecules if they are not included within the training set at all. There are currently very limited number of known LHCE diluent molecules in the literature. In addition to TTE, BTFE, and TFEO that we have previously discussed, the usage of 1,1,1,3,3,3-hexafluoroisopropyl methyl ether (HFPM),[31] 1H,1H,5H-octafluoropentyl 1,1,2,2-tetrafluoroethyl ether (OTE),[32] 2,2,2-trifluoroethyl 1,1,2,2-tetrafluoroethyl ether (TFTFE),[33] bis(2,2,2-trifluoroethyl) carbonate (BTFEC),[34] tri(2,2,2-trifluoroethyl) borate (TFEB),[34] ethoxy(pentafluoro) cyclotriphosphazene (PFPN),[35] 2,2,2-trifluoroethyl trifluoromethanesulfonate (TFEOTf)[36] have also been investigated recently. BTFE and TFEB exist within our training dataset, but this is not the case for any of the other LHCE diluent molecules. In Fig. 4 and Tables 3 and 4, we have focused on demonstrating the fine multi-constraint property control we have on our model. In a real LHCE diluent screening, we will want to use the model slightly differently, by distributing fine multi-constraint control over certain desired ranges (because otherwise our conditional generation criteria will be too restrictive and we will end up screening out useful molecules). To this end, we modify our query condition using the same RNN-based ConGen model we have trained in Fig. 4, but with property anchors uniformly and randomly sampled from:

(1) EA ε [0, …, 2.0] eV with interval of 0.2 eV

(2) IE ε [6.0, …, 8.0] eV with interval of 0.2 eV

(3) Log. Vis ε [−0.5, …, 0.5] with interval of 0.1

(4) Mol. Wt ε [150, …, 350] Da with interval of 10 Da

(5) n_F ε [4, …, 9] with interval of 1

(6) n_O ε [1, …, 3] with interval of 1

We query the model 50 000 times, producing 45 809 valid molecules (27 838 unique) as shown in Fig. 5a. Out of the unique molecules we can recover the diluent molecule which exists in the training dataset (BTFE) and some of the diluent molecules which do not exist in the dataset (TTE, HFPM, BTFEC, and TFTFE). However, we have not recovered TFEO, OTE, TFEB, PFPN, and TFEOTf within these 50 000 queries. We hypothesize that large molecules with very large number of F atoms such as TFEO, OTE, and TFEB (9, 12, and 9 F atoms respectively) are very poorly represented within our training dataset (Fig. 3), making it harder for the model to generate such molecules. Similarly, PFPN and TFEOTf are P-containing and S-containing fluorocarbon molecules respectively. Our training dataset has relatively few fluoro-phasphazene and fluoro-sulfonate compounds to train the model with, leading to the model's failure to re-discover these out-of-distribution chemistries. Overall, within the set of 27 838 unique molecules, only 3049 molecules have more than 8 F atoms, 2472 molecules have any S atoms, and 283 molecules have any P atoms. We note that some of the simpler known fluorocarbon-based diluent molecules such as *m*-fluorotoluene (*m*FT)[37] and benzotrifluoride (BTF)[38] are not within the intended generative target range of our query (they contain no O atoms and have ≤3 F atoms), and correspondingly they are not generated. In future iterations, it will be ideal to further augment the training dataset with fluoro-phosphazene and fluoro-sulfonate families, as well as molecules with large number of F atoms to further diversify the generative model's output.

Because the ConGen model can produce a relatively large number of unique LHCE diluent candidates despite having to satisfy multiple property constraints, it becomes difficult to validate all of them either computationally or experimentally. We propose a filtering mechanism based on synthesizability (only consider molecules with SA score <3.0) and novelty (multiple clustering queries based on molecular fingerprints to further select 100 molecules, followed by manual selection) and suggest the following unique 35 molecules for further investigation in the future as LHCE diluent molecules in Table 4, out of all the molecules we have generated in Fig. 4b, c, and 5. Generative model filtering mechanism is not the focus of our work, and we encourage interested readers to develop their own filtering criteria from the full list of candidate molecules (including those such as flammability, toxicity, *etc.*) which can be found in the electronically available ESI.†

Conclusion

In summary, we demonstrate a novel conditional molecule generation algorithm ConGen, which is based on semi-supervised variational auto-encoder (SSVAE) technology. However, unlike the SSVAE model, the ConGen model is explicitly designed such that it can work with dirty training data with incomplete labels. This is important because in practice, the molecules we can find from publicly available databases or characterized by in-house simulations and experiments will have incomplete sets of properties available, due to various factors (intellectual property, commercial secret, *etc.*) as well as experimental or computational cost considerations. A user of the baseline SSVAE model will need to remove molecules with incomplete labels or assign dummy labels on the molecules at

the cost of lower model accuracy. On the other hand, the Con-Gen model can easily mix dirty training datasets from multiple sources. ConGen is also designed with flexibility for substituting its sub-models with other types of models, enabling the user to include pre-trained models which can be helpful, especially in cases where there is limited training data availability. Finally, we demonstrate the practical use of our model for generating the virtual screening chemical space for Li-ion battery LHCE diluent candidates with multiple co-constraint requirements. We further perform leave-out cross-validation, in the process re-discovering 4 experimentally known LHCE diluent molecules which do not exist in our training dataset.

Experimental methods

Ab initio EA and IE validation

These molecule EA and IE calculations are conducted with PySCF's implementation[39,40] of the DFT Kohn–Sham method at the PBE6-31+G* level[41] with Grimme's dispersion correction[42] for geometry optimizations on the gas-phase and B3LYP/6-31+G* level of theory with the solvation energy corrections of tetrahydrofuran (THF) using the integral equation formalism polarizable continuum model (IEF-PCM)[43] implicit solvation model for single point energies. The vibrational frequencies were computed at the same level of theory at 298.15 K as for the geometry optimizations to confirm whether each optimized stationary point is an energy minimum. Here, we optimize the geometry at different charge states (cation, anion, neutral) to calculate the adiabatic IE/EA.

Data availability

All the training data sources, as well as all the structural and computational validation of the unconditionally and conditionally generated molecules are available in the ESI.† The full dataset is also available in Github: **https://github.com/jpmailoa/ConGen_Dataset**.

Author contributions

J. P. M. conceptualized this project, developed the ConGen model, and performed and analysed the model training and molecule generation experiments. S. Z. supervised the project. X. L. performed *ab initio* computational validation for the generated molecules. J.Q. supported model training optimization efforts. The manuscript was drafted by J. P. M., and reviewed by all the authors.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

References

1 M. A. Skinnider, *et al.*, A deep generative model enables automated structure elucidation of novel psychoactive substances, *Nat. Mach. Intell.*, 2021, **3**, 973–984.

2 Y. Li, J. Pei and L. Lai, Structure-based de novo drug design using 3D deep generative models, *Chem. Sci.*, 2021, **12**, 13664–13675.

3 H. Peng, Z. Liu, X. Yan, J. Ren and J. Xu, A de novo substructure generation algorithm for identifying the privileged chemical fragments of liver X receptorβ agonists, *Sci. Rep.*, 2017, **7**, 11121.

4 O. Prykhodko, *et al.*, A de novo molecular generation method using latent vector based generative adversarial network, *J. Cheminf.*, 2019, **11**, 74.

5 W. P. Walters and R. Barzilay, Applications of deep learning in molecule generation and molecular property prediction, *Acc. Chem. Res.*, 2021, **54**, 263–270.

6 N. W. A. Gebauer, M. Gastegger, S. S. P. Hessmann, K. R. Müller and K. T. Schütt, Inverse design of 3d molecular structures with conditional generative neural networks, *Nat. Commun.*, 2022, **13**, 973.

7 Y. Tan, L. Dai, S. Zheng, H. Chen and Y. Yang, DRlinker: Deep reinforcement learning for optimization in fragment linking design, *J. Chem. Inf. Model.*, 2022, **62**, 5907–5917.

8 Z. Zhou, S. Kearnes, L. Li, R. N. Zare and P. Riley, Optimization of molecules *via* deep reinforcement learning, *Sci. Rep.*, 2019, **9**, 10752.

9 T. Pereira, M. Abbasi, B. Ribeiro and J. P. Arrais, Diversity oriented deep reinforcement learning for targeted molecule generation, *J. Cheminf.*, 2021, **13**, 21.

10 M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen, Molecular de-novo design through deep reinforcement learning, *J. Cheminf.*, 2017, **9**, 48.

11 J. Wang, *et al.*, Multi-constraint molecular generation based on conditional transformer, knowledge distillation and reinforcement learning, *Nat. Mach. Intell.*, 2021, **3**, 914–922.

12 M. Goel, S. Raghunathan, S. Laghuvarapu and U. D. Priyakumar, MoleGuLAR: Molecule generation using reinforcement learning with alternating rewards, *J. Chem. Inf. Model.*, 2021, **61**, 5815–5826.

13 M. Popova, O. Isayev and A. Tropsha, Deep reinforcement learning for de novo drug design, *Sci. Adv.*, 2018, **4**, eaap7885.

14 S. Kang and K. Cho, Conditional molecular design with deep generative models, *J. Chem. Inf. Model.*, 2018, **59**, 43–52.

15 D. C. Elton, Z. Boukouvalas, M. D. Fuge and P. W. Chung, Deep learning for molecular design - A review of the state of the art, *Mol. Syst. Des. Eng.*, 2019, **4**, 828–849.

16 D. P. Kingma, D. J. Rezende, S. Mohamed and M. Welling, Semi-supervised learning with deep generative models, *Adv. Neural Inf. Process. Syst.*, 2014, **4**, 3581–3589.

17 S. Kim, *et al.*, PubChem in 2021: New data content and improved web interfaces, *Nucleic Acids Res.*, 2021, **49**, D1388–D1395.

18 J. J. Irwin and B. K. Shoichet, ZINC - A free database of commercially available compounds for virtual screening, *J. Chem. Inf. Model.*, 2005, **45**, 177–182.

19 X. Qu, *et al.*, The Electrolyte Genome project: a big data approach in battery materials discovery, *Comput. Mater. Sci.*, 2015, **103**, 56–67.

20 U. Lorenzo-Seva and P. J. Ferrando, Not positive definite correlation matrices in exploratory item factor analysis: Causes, consequences and a proposed solution, *Struct. Equ. Model.*, 2021, **28**, 138–147.

21 S. Chithrananda, G. Grand and B. Ramsundar, ChemBERTa: Large-scale self-supervised pretraining for molecular property prediction, *J. neural inf. process.*, 2020.

22 RDKit, *Open-Source cheminformatics*, 2020, https://www.rdkit.org/.

23 P. Ertl and A. Schuffenhauer, Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions, *J. Cheminf.*, 2009, **1**, 1–11.

24 X. Cao, H. Jia, W. Xu and J.-G. Zhang, Review—Localized high-concentration electrolytes for lithium batteries, *J. Electrochem. Soc.*, 2021, **168**, 010522.

25 X. Ren, *et al.*, Enabling high-voltage Lithium-metal batteries under practical conditions, *Joule*, 2019, **3**, 1662–1676.

26 S. Chen, *et al.*, High-efficiency Lithium metal batteries with fire-retardant electrolytes, *Joule*, 2018, **2**, 1548–1558.

27 Y. Yamada and A. Yamada, Review—Superconcentrated electrolytes for Lithium batteries, *J. Electrochem. Soc.*, 2015, **162**, A2406–A2423.

28 G. A. Giffin, The role of concentration in electrolyte solutions for non-aqueous lithium-based batteries, *Nat. Commun.*, 2022, **13**, 5250.

29 *Austin Apple Github*, 2020, https://github.com/AustinApple/SSVAE-for-electrolyte-molecule-design.

30 V. Goussard, *et al.*, A new machine-learning tool for fast estimation of liquid viscosity. Application to cosmetic oils, *J. Chem. Inf. Model.*, 2020, **60**, 2012–2023.

31 F. Ren, *et al.*, Solvent–diluent interaction-mediated solvation structure of localized high-concentration electrolytes, *ACS Appl. Mater. Interfaces*, 2022, **14**, 4211–4219.

32 J. Zheng, *et al.*, High-fluorinated electrolytes for Li–S batteries, *Adv. Energy Mater.*, 2019, **9**, 1803774.

33 H. Lu, *et al.*, Solvate ionic liquid electrolyte with 1,1,2,2-tetrafluoroethyl 2,2,2-trifluoroethyl ether as a support solvent for advanced lithium-sulfur batteries, *RSC Adv.*, 2016, **6**, 18186–18190.

34 X. Cao, *et al.*, Effects of fluorinated solvents on electrolyte solvation structures and electrode/electrolyte interphases for lithium metal batteries, *Proc. Natl. Acad. Sci. U. S. A.*, 2021, **118**, e2020357118.

35 C. Zhang, *et al.*, Nonflammable, localized high-concentration electrolyte towards a high-safety lithium metal battery, *Energy Stor. Mater.*, 2022, **52**, 355–364.

36 M. Zhu, X. Jiao, W. Wang, H. Chen and F. Li, Localized high-concentration electrolyte enabled by a novel ester diluent for lithium metal batteries, *Chem. Commun.*, 2022, **59**, 712–715.

37 J. Chen, *et al.*, Design of localized high-concentration electrolytes via donor number, *ACS Energy Lett.*, 2023, 1723–1734, DOI: 10.1021/acsenergylett.3c00004.

38 X. Peng, Y. Lin, Y. Wang, Y. Li and T. Zhao, A lightweight localized high-concentration ether electrolyte for high-voltage Li-Ion and Li-metal batteries, *Nano Energy*, 2022, **96**, 107102.

39 Q. Sun, *et al.*, Recent developments in the PySCF program package, *J. Chem. Phys.*, 2020, **153**, 024109.

40 Q. Sun, Libcint: An efficient general integral library for Gaussian basis functions, *J. Comput. Chem.*, 2015, **36**, 1664–1671.

41 J. P. Perdew, K. Burke and M. Ernzerhof, Generalized Gradient Approximation Made Simple, *Phys. Rev. Lett.*, 1996, **77**, 3865–3868.

42 S. Grimme, J. Antony, S. Ehrlich and H. Krieg, A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu, *J. Chem. Phys.*, 2010, **132**, 154104.

43 J. Tomasi, B. Mennucci and E. Cancès, The IEF version of the PCM solvation method: An overview of a new method addressed to study molecular solutes at the QM ab initio level, *J. Mol. Struct.: THEOCHEM*, 1999, **464**, 211–226.