Supporting information for Attention-based generative models for *de novo* molecular design

Orion Dollar[1], Nisarg Joshi[1], David A.C. Beck[1,2], Jim Pfaendtner[1]

[1]Department of Chemical Engineering, University of Washington, Seattle 98185, WA, USA;

[2]eScience Institute, University of Washington, Seattle 98185, WA, USA.

Additional Details on Model Construction

All models are constructed using the PyTorch python package and trained on NVIDIA GeForce RTX 2080 Ti GPUs. A batch size of 500 was used for all model types besides Moses. SMILES strings are tokenized during training and embeddings the same size as the model dimension are used as inputs. The maximum length of SMILES token for all model types is 127 including <start>, <stop> and padding tokens. All recurrent layers are unidirectional and masking for the transformer is done sequentially such that for a sequence of leng th t, the model attempts to predict the token at position t+1. The same model architectures are used for both the ZINC and PubChem datasets. For the recurrent models, teacher forcing is partially used during training by concatenating the input embeddings with the unbottlenecked model memory prior to being sent to the decoder GRU layers.

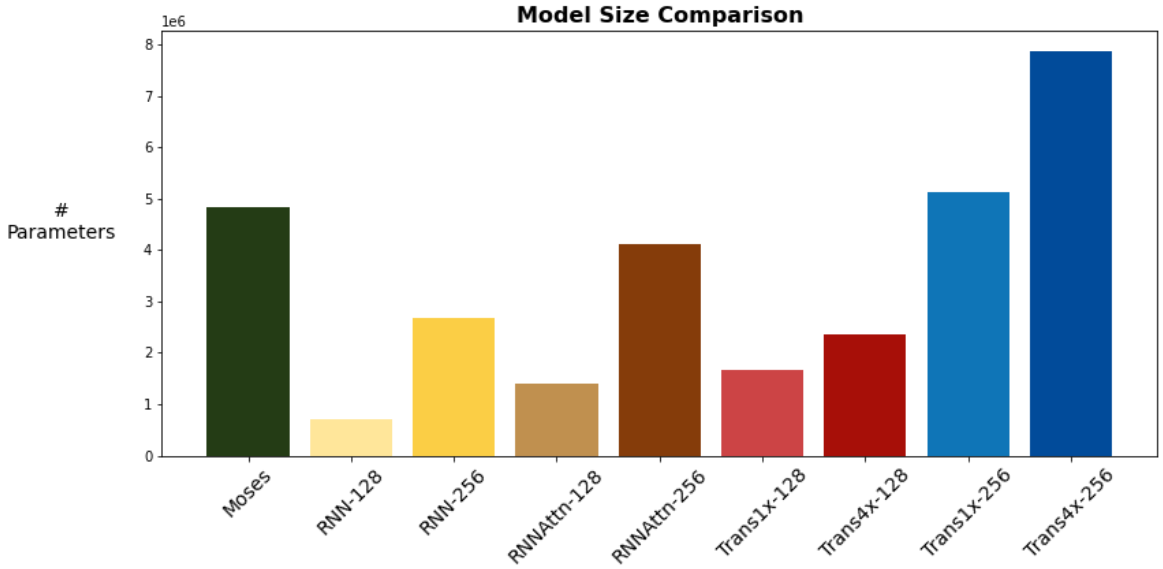

Figure S1. *Size comparison of all model types*

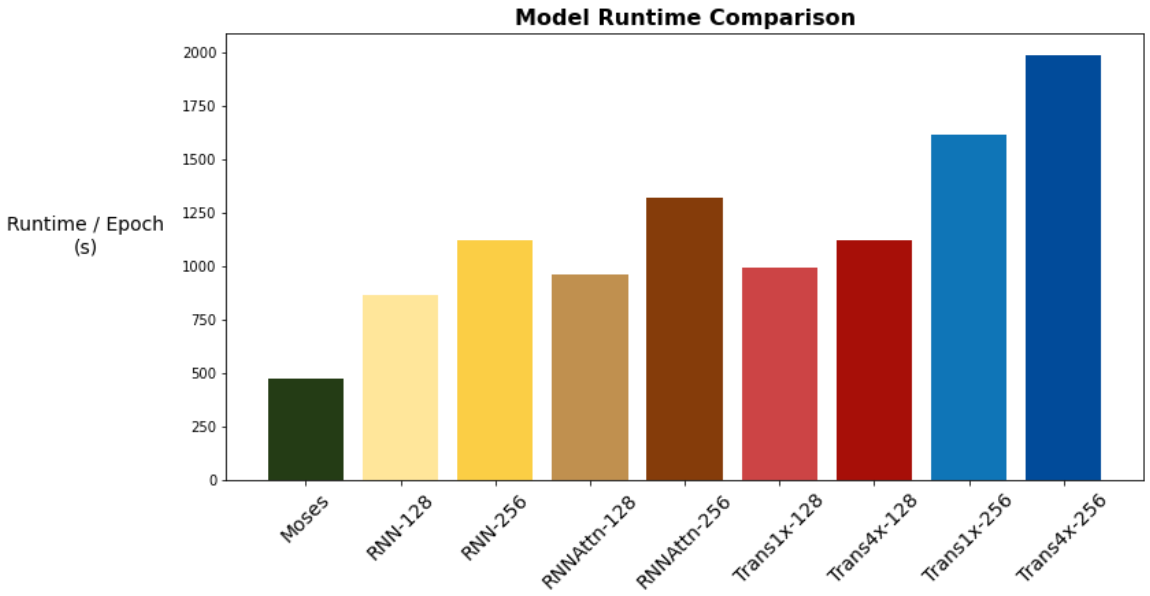

Figure S2. *Runtime of all model types on ZINC training set. The increased efficiency of the Moses model is due to the number of padding tokens. This is variable for the Moses construction based on the input data but fixed for the other models based on the longest SMILES string within the PubChem dataset. Fixing the maximum sequence length simplified the construction of the convolutional bottleneck for different model dimensions.*

Token Weights and KL Annealing

SMILES strings have an unbalanced token distribution so it is necessary to weigh loss by the frequency with which the token appears so that the model does not just learn to repeat the most frequent token. Characters are weighed by their proportional log frequency and then scaled to values between 0.5 and 1.0 (so the most frequent characters can represent no less than 50% of their calculated loss). The padding character is manually set to a weight of 0.1. A KL Annealer is used to linearly increase β to avoid posterior collapse. For all trials except the MosesVAE, β is increased from 0 to 0.083 over 100 epochs. For MosesVAE, β is linearly increased from 0 to 0.05 over 100 epochs.

Convolutional Bottleneck Layers

A convolutional bottleneck was used for the two attention-based architectures based on the size of the contextual embedding exiting the encoder and entering the decoder. Conceptually, the embedding matrix of these models is closer in shape to the learned image representation in a CNN than the contextual embedding vector of the RNN so our intuition was to try bottlenecking with convolutional layers. Empirically, we found our intuition to be correct as the reconstruction performance of the models is much better when using the convolutional bottleneck (Fig. S3).

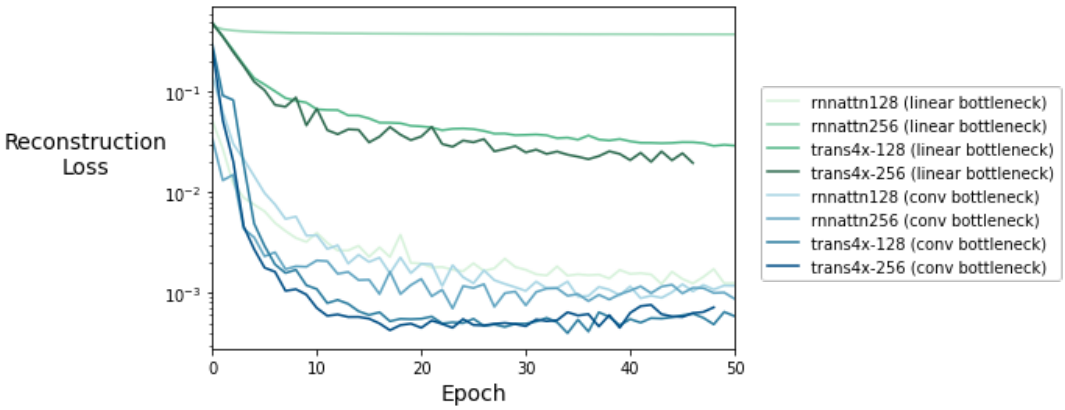

Figure S3. *Comparison of reconstruction loss for the attention-based architectures with a linear bottleneck and convolutional bottleneck. With the exception of the RNNAttn-128 model, the convolutional bottleneck outperforms the linear bottleneck.*

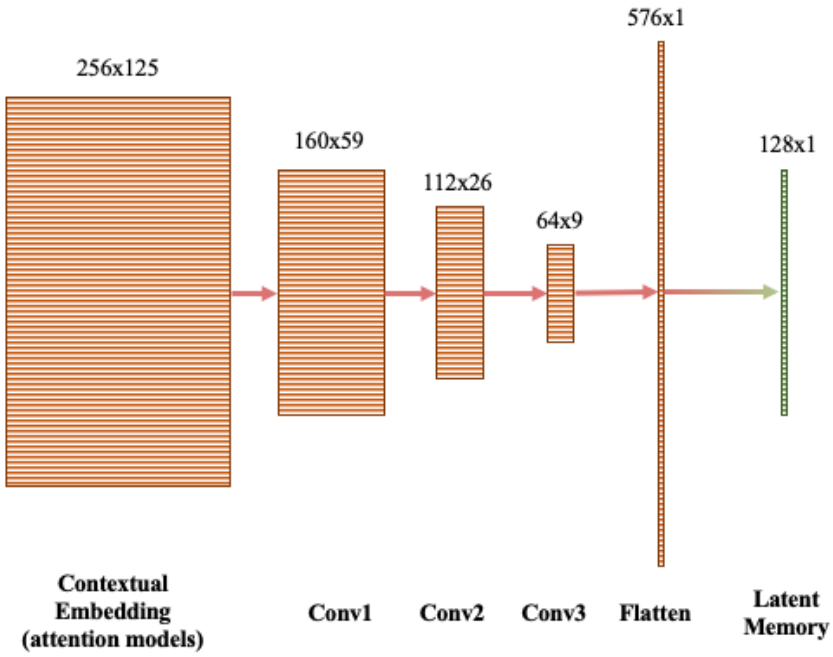

Figure S4*. The shape of the contextual embedding within the model as it travels through the convolutional bottleneck. A similar set of deconvolutional layers are used to upsample back to the original shape from the latent memory before being sent into the decoder.*

We illustrate the compression of data through the convolutional bottleneck in Fig. S4. The architecture consists of three 1D convolutional layers that compress the contextual embedding to a size 576 vector for model dimensions of 128, 256 or 512. The final 576 size vector is then compressed with a linear layer to the mean and logvar vectors before reparameterization. Each 1D conv layer is attached with a 1D MaxPool layer of size 2. After compression, three 1D deconvolutional layers are used to upsample the bottleneck back to the original size of the contextual embedding. The parameters for the convolutional layers depend on the size of the model and are listed in full in Table S1.

Table S1. *Convolutional Bottleneck Parameters*

	Conv 1		***Conv 2***		***Conv 3***	
Model Dim	***Channel Size***	***Kernel Size***	***Channel Size***	***Kernel Size***	***Channel Size***	***Kernel Size***
128	96	9	80	9	64	8
256	160	9	112	9	64	8
512	288	9	176	9	64	8

	Deconv 1			***Deconv 2***			***Deconv 3***		
Model Dim	***Channel Size***	***Kernel Size***	***Stride***	***Channel Size***	***Kernel Size***	***Stride***	***Channel Size***	***Kernel Size***	***Stride***
128	80	11	4	92	11	3	128	11	1
256	112	11	4	148	11	3	256	11	1
512	176	11	4	260	11	3	512	11	1

Predicting SMILES Length During Transformer Inference

There are two inputs to the transformer decoder, the model memory and the input mask which consists of 0s for all padding tokens and 1s for every other token. The mask explicitly tells the model the length of the SMILE string so during inference it must also have this information or else it will decode molecules from memory that are much longer than they should be. To account for this, we attached two linear layers of size d_{model}*2 to the latent memory and instructed the model to predict the correct length of the molecule during training and included this in our transformer loss function. Then during inference, we first predict the length of the SMILE string using our randomly sampled latent vector as an input and use this to create the correct length mask to send to the decoder (so we do not need to know the length of the SMILE string before we decode it).

Model Complexity

The definition of complexity we use throughout this work stems from the definition introduced by Tishby et al.[1] in which the bottleneck of a model can be analyzed in two ways – predictive ability and compressibility. There is a "generalization gap" which bounds the amount of salient information the model *could have* captured but didn't and a "complexity gap" which bounds the amount of "unnecessary complexity" that exists within the bottleneck. The concept of complexity in this case is tied to the compressibility of the bottleneck (i.e. it is not an algorithmic complexity but merely a synonym for low information content noise). The use of the phrase "unnecessary complexity" implies a corresponding "necessary complexity" and so we have drawn a link between the total information content contained within the bottleneck and the "complexity" of the model. Thus mentions of model

complexity are referring to the models ability to efficiently compress all of the salient information needed for reconstruction within the bottleneck. Others have made similar observations about the relationship between the loss function of the β-VAE and the compression of the latent memory,[2] although explicit use of the term complexity to describe this phenomenon has been limited to Tishby et al. and the descriptions herein as far as we are aware.

Supplementary Figures

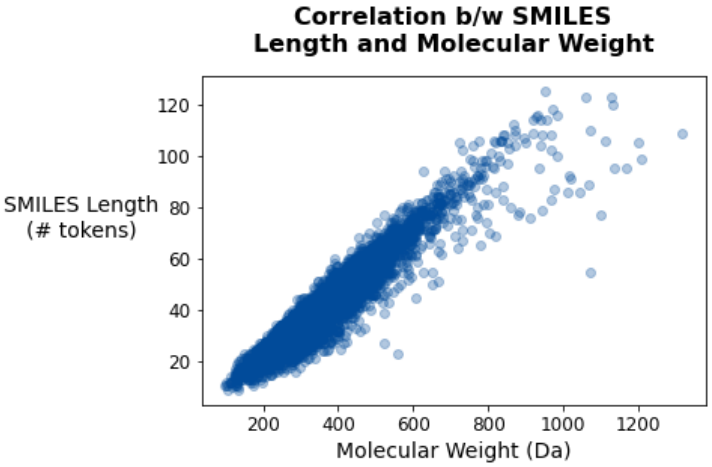

Figure S5*. The relationship between SMILES length and MW (PubChem dataset)*

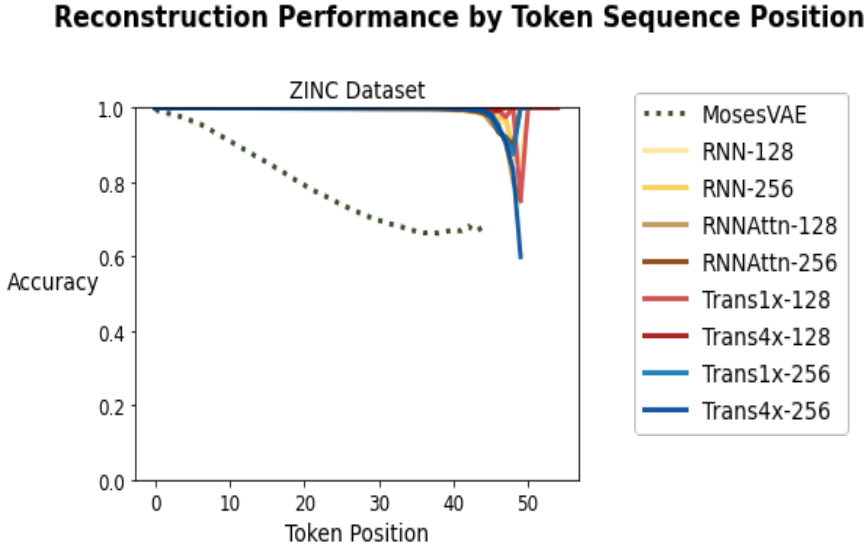

Figure S6. *The reduced reconstruction performance of the Moses model may be the result of a number of architectural and hyperparameter decisions. In addition to the differences mentioned in the procedure, we also used a more concise tokenization scheme (for instance `Br` was treated as a single token rather than tokenizing as `B` and `r` separately), we updated model weights more aggressively for tokens that appeared less frequently and we used larger token embeddings. The exact degree to which these factors played a role in the model's performance is still unknown. Because we were able to replicate all of the reported metrics from the original Moses paper (Fig. S7) we believe this is an accurate portrayal of the Moses model and include it to highlight an example of 'smeared' latent memory formation.*

Table S2. *Reconstruction performance of all model types on ZINC dataset (MosesVAE was not saved at epoch 100 so accuracy at epoch 90 is shown instead).*

Model Type	***Token Accuracy***	***SMILES Accuracy***
MosesVAE (Epoch 90)	0.1416	0.000
RNN-128	0.9988	0.9955
RNN-256	0.9986	0.9957
RNNAttn-128	0.9990	0.9963
RNNAttn-256	0.9986	0.9948
Trans1x-128	0.9996	0.9978
Trans4x-128	0.9996	0.9979
Trans1x-256	0.9997	0.9983
Trans4x-256	0.9996	0.9980

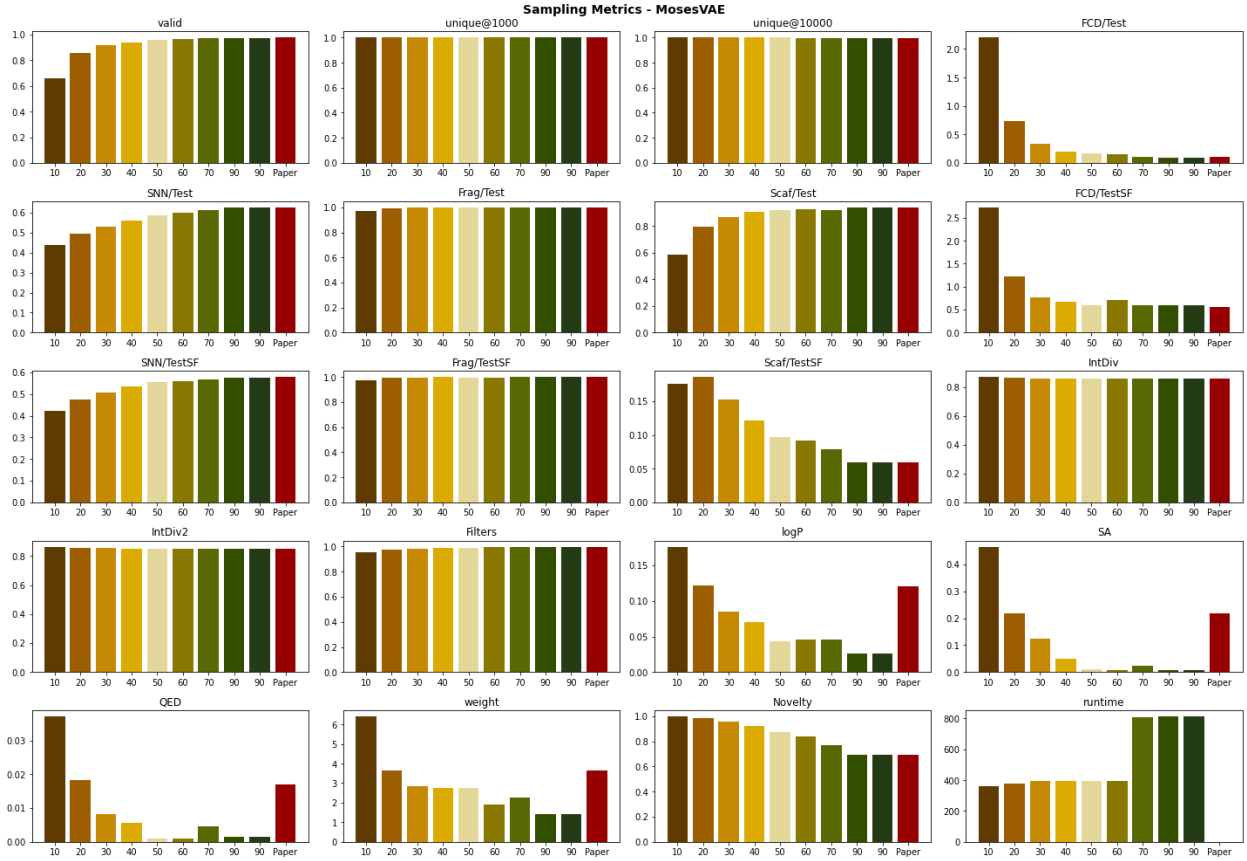

Figure S7. *Evaluating the MosesVAE on the suite of metrics presented in the MOSES paper. After 100 epochs, the model converges to all reported values from the paper validating the use of our trained Moses model as an example of the state-of-the-art as presented by Polykovskiy et al.*

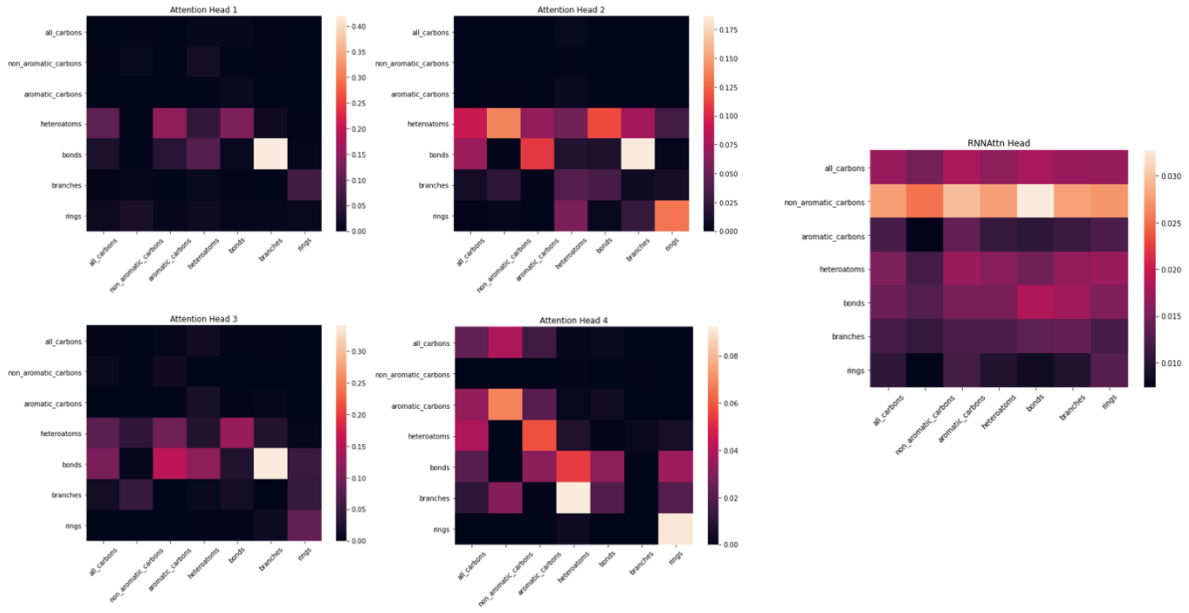

Figure S8. *Analysis of attention weights between structural and atomic groups. The four attention heads of the transformer learn unique molecular grammar rules, even for higher-level relationships such as the relationship between all heteroatoms and all explicitly enumerated bonds present within the structure. The RNNAttn head has given the most weight to the relationship between non-aromatic carbons and all other atomic/structural groups which is more useful for compressing long-range information efficiently than learning specific relationships that are important to molecular structure.*

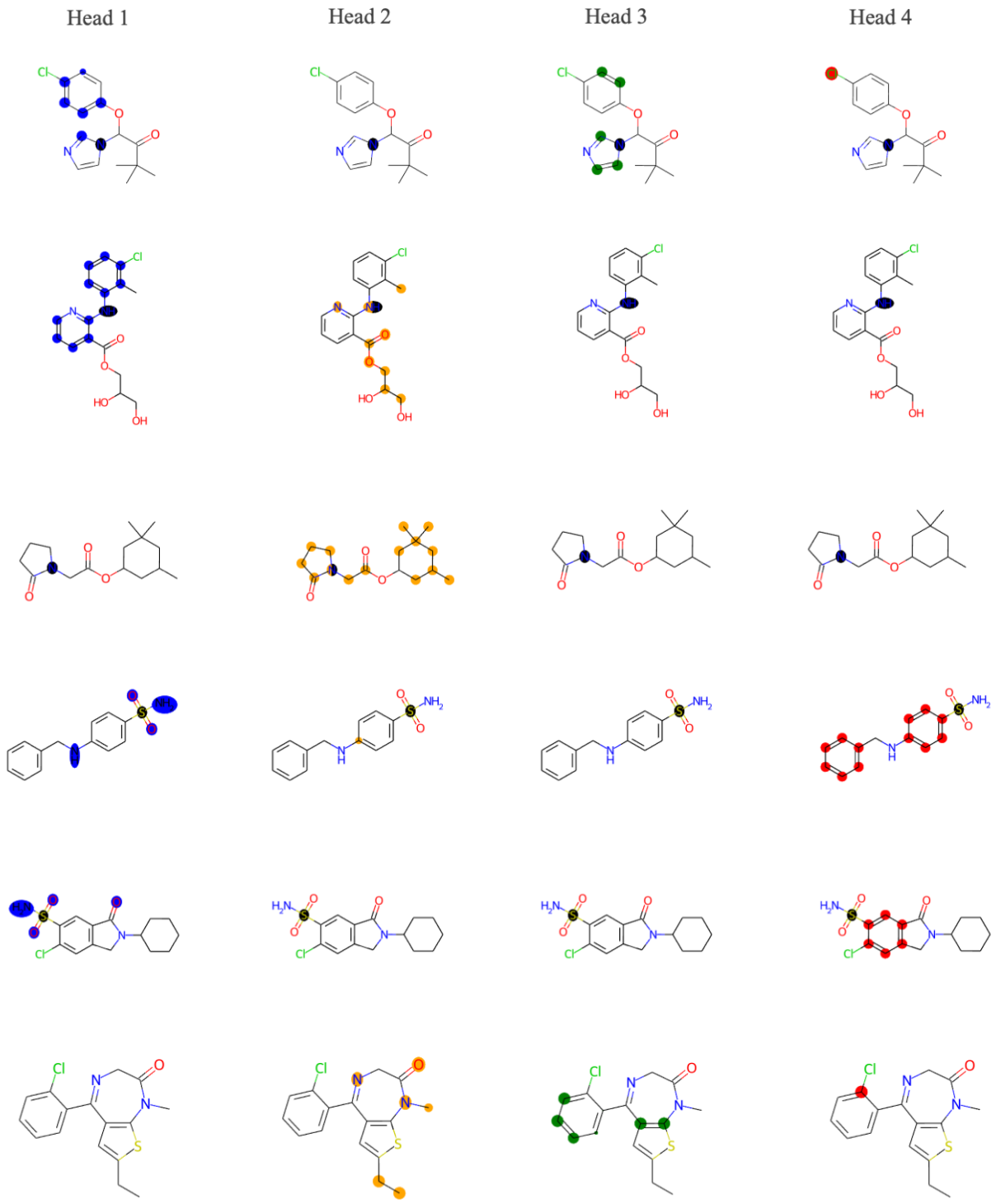

Figure S9. *Visualization of attention weights within the Trans4x-256 model of S and N heteroatoms for a variety of molecular structures. The learned patterns depend on the type of heteroatom. For instance, attention head 1 shows the relationship between N and aromatic carbons however a similar relationship between S and aromatic carbons is stored within head 4. The patterns are usually consistent for the same atom type across different molecular structures, however different patterns may also emerge depending on the molecular context around the atom (i.e. the aromatic S atom vs. the sulfonyl group). These relationships are heavily influenced by the input representation and may potentially be tuned by altering the type of information the model has access to.*

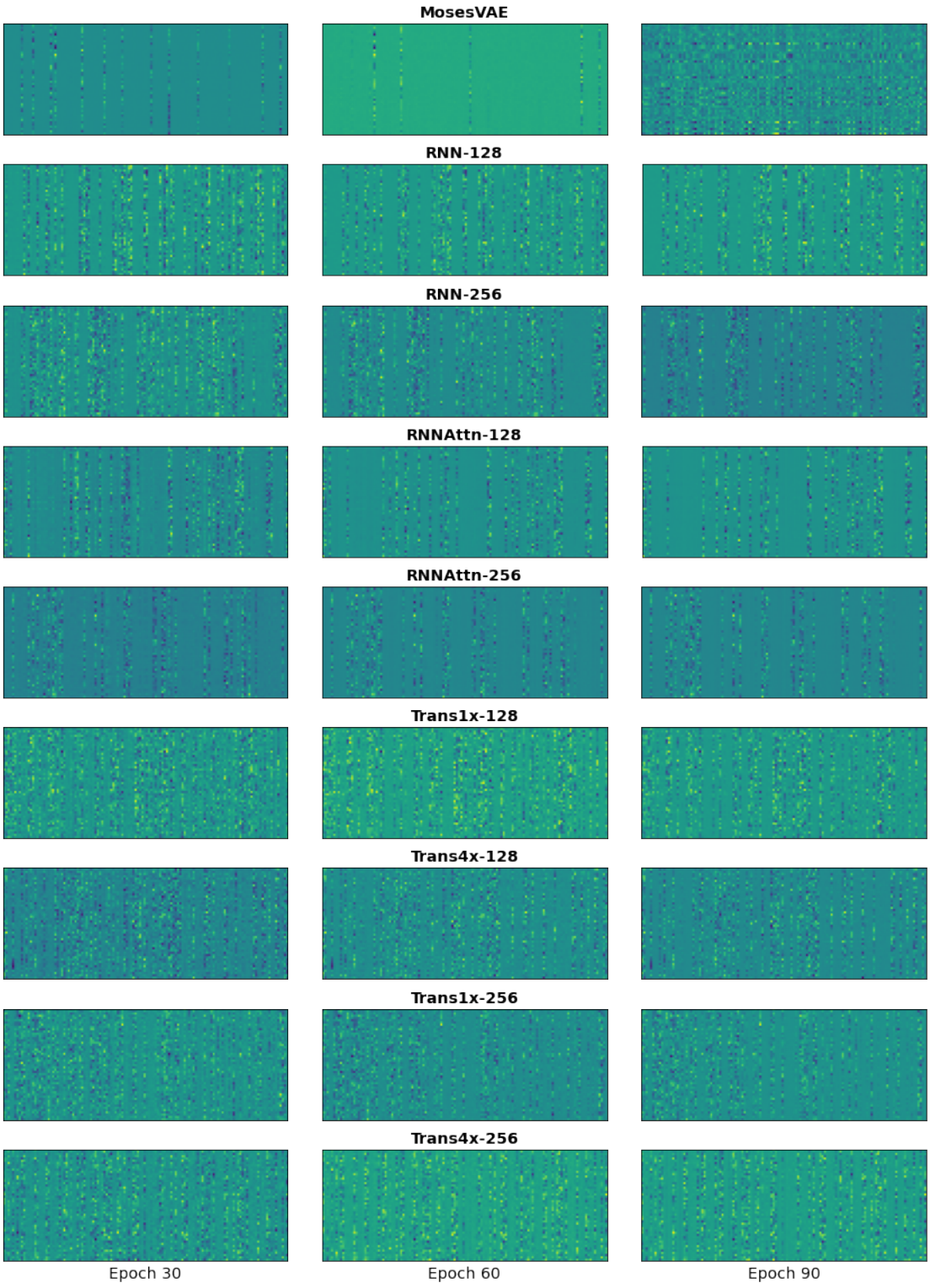

Figure S10. *Memory structures for all model types at epochs 30, 60 and 90*

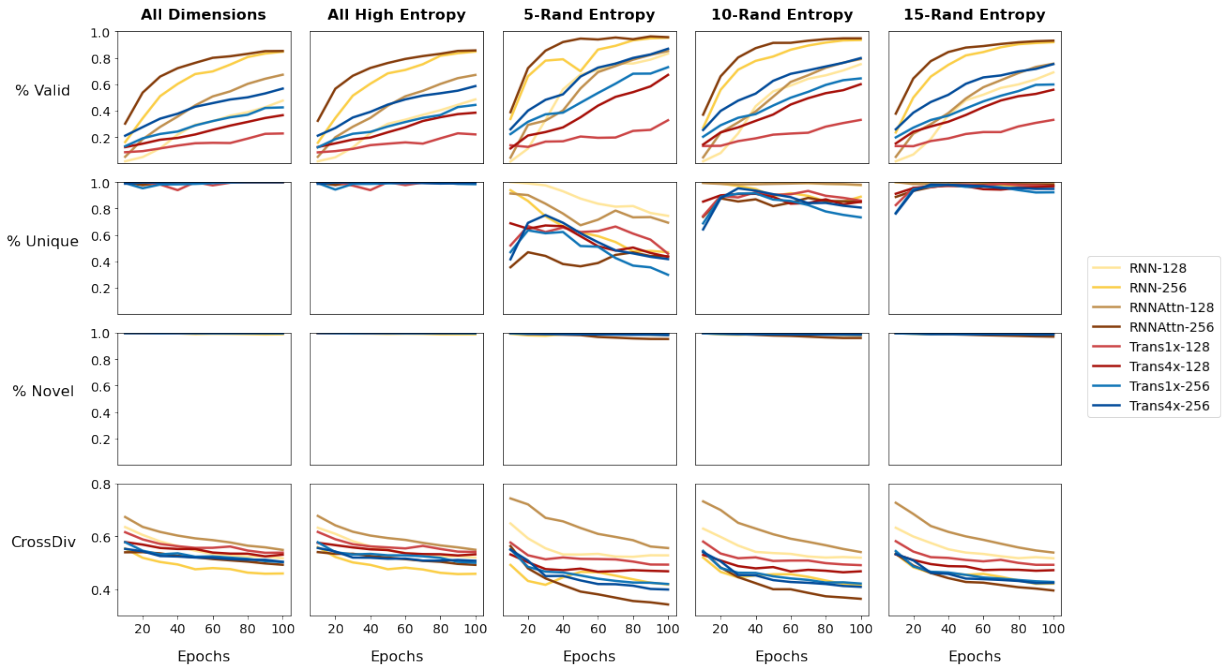

Figure S11. *Five different sampling schemes are tested for their effect on generative performance metrics – sampling all 128 latent dimensions, sampling only those dimensions with a high entropy (> 5 nats) and sampling k-random high entropy dimensions (k=5,10,15). 30,000 molecules were generated for each scheme. There is essentially no difference between sampling all dimensions and randomly sampling just the high entropy dimensions, however there is an improvement in validity when sampling from a small number of randomly selected high entropy dimensions. Sampling 15 random high entropy dims significantly increases % validity for all model types while maintaining high uniqueness, novelty and exploration.*

References

1 N. Tishby and N. Zaslavsky, in *2015 IEEE Information Theory Workshop (ITW)*, Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1–5.

2 A. A. Alemi, I. Fischer, J. v. Dillon and K. Murphy, *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.