

259 A Author contribution

260 *Anonymized for the submission*

261 B Training

262 Training a joint generative model was previously shown to result in a good generator together with a
 263 poor predictor (Lasserre et al., 2006; Nalisnick et al., 2019). Moreover, storing the gradients for all
 264 the summands of the objective (Eq. 2) is a significant overhead in memory requirements as compared
 265 to decoder and encoder-only Transformers. To overcome both issues, we propose a practical training
 266 procedure for JOINT TRANSFORMER (Alg. 3) that randomly switches between the input generation
 267 and the property prediction (and encoder training) tasks with a hyperparameter $p_{\text{task}} \in [0, 1]$.

268 The JOINT TRANSFORMER can be trained in an unsupervised, semi-supervised or supervised setting.
 269 Depending whether a target $y \in \mathcal{Y}$ is sampled from the dataset \mathcal{D} or is not available (Step 2, Alg. 3),
 270 one can include the prediction loss $\ln p_{\theta, \phi}(y | \mathbf{x})$ in the penalized log-likelihood objective ℓ (Step 6,
 271 Alg. 3), resulting in a supervised setting, or set the prediction loss to zero, resulting in an unsupervised
 272 setting. For the training data where only a small proportion of samples have accompanying target
 273 values, we split the training procedure of the JOINT TRANSFORMER into first training the model in
 274 an unsupervised manner (Alg. 2), then fine-tuning it with supervised data (Alg. 3).

Algorithm 2 Unsupervised training of JOINT TRANSFORMER

Input: A dataset $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$. JOINT TRANSFORMER $p_{\theta, \phi}(\mathbf{x}, y)$ with parameters θ, ϕ containing
 a decoder $p_{\theta}(\mathbf{x})$, encoder $\prod_{d=1}^D p_{\theta}(x_d | \mathbf{m} \odot \mathbf{x}_{-d})$.
 Task probability $p_{\text{task}} \in [0, 1]$ and a masking distribution $q(\mathbf{m})$.
 1: **while** a stopping criterion is not met **do**
 2: Uniformly sample \mathbf{x} from the dataset \mathcal{D}
 3: Sample an indicator $u \sim \text{BERNOULLI}(p_{\text{task}})$
 4: **if** $u = 0$ **then**
 5: Sample mask $\mathbf{m} \sim q(\mathbf{m})$
 6: Calculate loss $\ell(\theta, \phi) = -\sum_{d=1}^D \ln p_{\theta}(x_d | \mathbf{m} \odot \mathbf{x}_{-d})$
 7: **else**
 8: Set mask to the causal mask
 9: Calculate loss $\ell(\theta, \phi) = -\ln p_{\theta}(\mathbf{x})$
 10: **end if**
 11: Update parameters θ, ϕ using an optimizer w.r.t. loss ℓ
 12: **end while**

Algorithm 3 Training of JOINT TRANSFORMER

Input: A dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$. JOINT TRANSFORMER $p_{\theta, \phi}(\mathbf{x}, y)$ with parameters θ, ϕ
 containing a decoder $p_{\theta}(\mathbf{x})$, encoder $\prod_{d=1}^D p_{\theta}(x_d | \mathbf{m} \odot \mathbf{x}_{-d})$ and a predictor $p_{\theta, \phi}(y | \mathbf{x})$.
 Task probability $p_{\text{task}} \in [0, 1]$ and a masking distribution $q(\mathbf{m})$.
 1: **while** a stopping criterion is not met **do**
 2: Uniformly sample (\mathbf{x}, y) from the dataset \mathcal{D}
 3: Sample an indicator $u \sim \text{BERNOULLI}(p_{\text{task}})$
 4: **if** $u = 0$ **then**
 5: Sample mask $\mathbf{m} \sim q(\mathbf{m})$
 6: Calculate loss $\ell(\theta, \phi) = -\sum_{d=1}^D \ln p_{\theta}(x_d | \mathbf{m} \odot \mathbf{x}_{-d}) - \ln p_{\theta, \phi}(y | \mathbf{x})$
 7: **else**
 8: Set mask to the causal mask
 9: Calculate loss $\ell(\theta, \phi) = -\ln p_{\theta}(\mathbf{x})$
 10: **end if**
 11: Update parameters θ, ϕ using an optimizer w.r.t. loss ℓ
 12: **end while**

C Sampling

C.1 Unconditional Generation

In the unconditional generation task, we sample from JOINT TRANSFORMER in a two-step manner that results in an unconditional sample $(\mathbf{x}, y) \sim p_{\theta, \phi}(\mathbf{x}, y)$. First, since the decoder part $p_{\theta}(\mathbf{x})$ does not depend on parameters ϕ and it properly defines an ARM, we sample $\mathbf{x} \sim p_{\theta}(\mathbf{x})$. Next, we sample a target y from the predictive distribution $y \sim p_{\theta, \phi}(y | \mathbf{x})$. The key feature that allows for successful sampling from the joint model is the ability of the JOINT TRANSFORMER to simultaneously operate in two separate modes, namely generate novel examples and predict their target values, which is directly encouraged by training with the penalized log-likelihood objective in Eq. 2.

C.2 Conditional Generation

In the conditional generation task, given a condition $Y \subseteq \mathcal{Y}$, we sample from JOINT TRANSFORMER $p_{\theta, \phi}(\mathbf{x}, y)$ to obtain a conditional sample $(\mathbf{x}, y) \sim p_{\theta, \phi}(\mathbf{x}, y)$, such that $y \in Y$. JOINT TRANSFORMER generates conditional samples by first sampling $(\mathbf{x}, y) \sim p_{\theta, \phi}(\mathbf{x}, y)$ in the above described unconditional way and then accepting the sample if $y \in Y$. In practical applications, due to a finite runtime, we sample a batch of B tuples $(\mathbf{x}, y) \sim p_{\theta, \phi}(\mathbf{x}, y)$ and choose (\mathbf{x}, y) with y ‘closest’ to Y . Proposition 1 shows that, despite its conceptual simplicity, the described conditional generation procedure is equivalent to directly sampling from the conditional distribution $p(\mathbf{x} | y)$. Moreover, Proposition 2 shows conditions under which conditional generation enjoys a finite expected runtime.

Proposition 1. *Let $p(\mathbf{x}, y)$ be a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. Let $y_c \in \mathcal{Y}$ be such that $p(y_c) > 0$. Then*

$$p(\mathbf{x} | y_c) \propto \mathbb{1}_{\{y=y_c\}}(y)p(y | \mathbf{x})p(\mathbf{x}).$$

Proof. Assume that $p(\mathbf{x}, y)$ is a joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. Choose $y_{\max} \in Y$ to be such that $p(y \geq y_c) > 0$. Then a simple application of Bayes rule yields

$$p(\mathbf{x} | \{y \geq y_c\}) = \frac{p(\mathbf{x}, \{y \geq y_c\})}{p(\{y \geq y_c\})} = \frac{\mathbb{1}_{\{y \geq y_c\}}(y)p(y | \mathbf{x})p(\mathbf{x})}{p(\{y \geq y_c\})}. \quad (3)$$

Since $p(\{y \geq y_c\}) > 0$ and it does not depend on \mathbf{x} , we have that

$$p(\mathbf{x} | \{y \geq y_c\}) \propto \mathbb{1}_{\{y \geq y_c\}}(y)p(y | \mathbf{x})p(\mathbf{x}).$$

□

Proposition 2. *Let $p(y)$ be a probability distribution over \mathcal{Y} with a corresponding cumulative distribution function F . Let target $y_c \in \mathcal{Y}$ be such that $p(y_c) > 0$ and let p be the probability of sampling a target $y \sim p(y)$ such that $y > y_c$. The expected number of trials N until obtaining a sample $y \sim p(y)$ such that $y > y_c$ is equal to $1/p$.*

Proof. Let $p(y)$ be a probability distribution over \mathcal{Y} with a corresponding cumulative distribution function F . Let $y_c \in \mathcal{Y}$ be such that $p(y_c) > 0$. Define r.v. N as the number of trials until obtaining a sample $y > y_c$, where y is distributed as $p(y)$. For each $n \in \mathbb{N}$, the distribution of N is given by

$$P(N = n) = (1 - p)^{n-1}p,$$

where $p = 1 - F(y \leq y_c)$. Hence, the number of trials N follows a geometric distribution with an expected value equal to $\mathbb{E}[N] = 1/p$. □

Despite its simplicity, the conditional generation of JOINT TRANSFORMER has the advantage of the predictor $p_{\theta, \phi}(y | \mathbf{x})$, as it is defined in the input space \mathcal{X} , indicating whether the newly generated example enjoys the desired target value. This is in contrast to methods based on LSO and diffusion models, see (Gómez-Bombarelli et al., 2018; Hoogeboom et al., 2022).

D Additional Experiments

D.1 Molecule Generation

Task In the molecule generation task, the goal is to generate valid and novel molecules that follow the chemical distribution of the training data. Following Brown et al. (2019), we evaluate all molecule generation methods on five metrics: validity, a fraction of the generated molecules that are correspond to a valid SMILES string; uniqueness, a fraction of the generated molecules that are unique; novelty, a fraction of the generated molecules that are not present in the training data; KL Divergence, a measure of similarity of the generated molecules to the training set with respect to selected chemical properties (Brown et al., 2019), as well as Fréchet ChemNet Distance (FCD; (Preuer et al., 2018)), a general measure of similarity of the generated molecules to the training set.

Baselines As baselines, we select well-established molecule generation models based on SMILES representation (Weininger, 1988): LSTM (Ertl et al., 2018), VAE (Kingma & Welling, 2013; Rezende et al., 2014) and AAE (Kadurin et al., 2016). Additionally, we consider graph-based models: Junction Tree VAE (Jin et al., 2018), MoLeR (Maziarz et al., 2021) and MAGNet (Hetzel et al., 2023). Finally, we include MolGPT (Bagal et al., 2022), which is a Transformer-based model and the backbone for the JOINT TRANSFORMER, sharing the same architecture, but trained differently.

Results In the molecule generation task, JOINT TRANSFORMER successfully generates valid, unique and novel molecules (Tab. 2). Moreover, JOINT TRANSFORMER generates molecules with properties that closely follow the training set distribution, making the newly generated molecules realistic and physio-chemically plausible, as measured by KL Divergence and FCD. Compared to the backbone MolGPT model, JOINT TRANSFORMER achieves identical performance, showing that the modified training procedure does not hurt the generative functionality of the model. From the generative modeling perspective, this result is counterintuitive, as we can include the reconstruction task to the training procedure of the JOINT TRANSFORMER, without sacrificing its generative performance.

Overall, none of the molecule generation methods achieves best performance across all metrics. Graph-based methods outperform others on validity, as they generate always valid molecules by design. However, the improvement of 3% as compared to Transformer-based models (JOINT TRANSFORMER and MolGPT) is negligible. Additionally, it comes at the expense of generating molecules with decreased (from 12% to 19%) values of the KL Divergence and FCD metrics. On the other hand, LSTM achieves top performance on KL Divergence and FCD metrics, slightly (1% and 3%, respectively) outperforming Transformer-based methods, but falls behind in the validity of the generated molecules. All methods successfully generate unique and novel molecules. Overall, JOINT TRANSFORMER strikes a good balance between graph-based and SMILES-based LSTM, making it a viable choice for a go-to molecule generation model.

Table 2: Molecule Generation Task. JOINT TRANSFORMER (JT) matches state-of-the-art performance of different molecule generation methods. Training the JOINT TRANSFORMER model on generation and reconstruction tasks simultaneously does not hurt the generation performance of the model.

MODEL	SIZE	VALIDITY (\uparrow)	UNIQUENESS (\uparrow)	NOVELTY (\uparrow)	FCD (\uparrow)	KL Div (\uparrow)
LSTM	-	0.96	1.0	0.91	0.91	0.99
VAE	-	0.87	1.0	0.97	0.86	0.98
AAE	-	0.82	1.0	1.0	0.53	0.89
JT-VAE	-	1.0	N/A	N/A	0.76	0.94
MAGNET	6.9M	N/A	N/A	N/A	0.73	0.92
MoLER	-	1.0	0.99	0.97	0.78	0.98
MOLGPT	6M	0.98	1.0	1.0	0.91	0.99
MOLGPT (OURS)	6M	0.97	1.0	0.97	0.89	0.98
JT (OURS)	6M	0.97	1.0	0.98	0.89	0.99
JT (OURS)	50M	0.98	1.0	0.95	0.90	0.99

D.2 Unconditional Generation

Moreover, the jointly trained predictor $q_{\theta,\phi}(y \mid \mathbf{x})$ of the JOINT TRANSFORMER generalizes well to data generated with the model $p_{\theta}(\mathbf{x})$. In particular, the prediction error, as measured by mean absolute error, of the JOINT TRANSFORMER fine-tuned on three properties from the Guacamol task (Brown et al., 2019) do not change between the test set and newly generated data (Table 3). This shows good generalization performance of JOINT TRANSFORMER.

Table 3: Mean absolute prediction error (MAE) for the predictor on three property prediction tasks on test and generated data. Mean and standard deviation across independent runs.

METHOD	DATA	PERINDOPRIL MPO	SITAGLIPTIN MPO	ZALEPLON MPO
JOINT TRANSFORMER	TEST	0.014 ± 0.004	0.009 ± 0.001	0.012 ± 0.001
	GENERATED	0.015 ± 0.004	0.009 ± 0.002	0.012 ± 0.001

E Implementation Details

E.1 Data and Tokenization

We use SMILES (Weininger, 1988) based representations of molecules across all experiments. In all experiments we pre-train the JOINT TRANSFORMER in an unsupervised manner using the ChEMBL database, a manually curated database of molecules with drug-like properties (Mendez et al., 2019). As opposed to other datasets like ZINC (Irwin et al., 2020), ChEMBL contains only molecules which have been synthesized. To ensure reproducibility and comparability with molecule generation baselines we use version 24 of the database that contains 1.8M compounds altogether and apply standard data processing used in the Guacamol benchmark (Brown et al., 2019). As for tokenization of the data, we use a tokenizer based on (Schwaller et al., 2020). We additionally use an augmentation method of SMILES representations based on (Tetko et al., 2019) and similar to (Bagal et al., 2022) across all experiments and methods. This ensures the transferability of results obtained by Bagal et al. (2022) to our experiments.

E.2 Architecture

Our implementation of the JOINT TRANSFORMER follows the implementation provided by (Karpathy, 2023), which is a re-implementation of a GPT-2 (Radford et al., 2019) used by MolGPT (Bagal et al., 2022). The only difference is that during each forward pass, we switch between a causal and a bidirectional masking, depending on the task we are optimizing for. We additionally stack an MLP network on the top of the first output token for prediction. The complete list of hyperparameters is presented in Table 4. Our implementation results in a model with 6.5M parameters.

Table 4: Model hyperparameters for the JOINT TRANSFORMER used across all experiments.

HYPERPARAMETER	VALUE
ACTIVATION FN	GELU
EMBED DIM	256
NUM LAYERS	6
NUM HEADS	8
FEEDFORWARD DIMENSION	1024
FEEDFORWARD BIAS	FALSE
LAYER NORM EPSILON	$1e-5$
PREDICTOR HEAD	MLP
PREDICTOR NUM LAYERS	1
PREDICTOR HIDDEN DIM	100

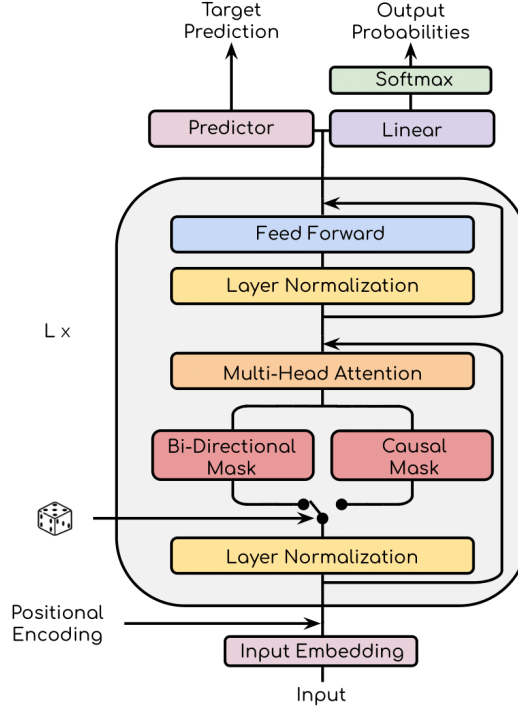


Figure 2: JOINT TRANSFORMER architecture.

Table 5: Training hyperparameters of the JOINT TRANSFORMER used across all experiments.

HYPERPARAMETER	VALUE
BATCH SIZE	64
TOTAL NUMBER OF TRAINING ITERATIONS	4.2 M
OPTIMIZER	ADAMW
WEIGHT DECAY	$1e-1$
BETA 1	0.9
BETA 2	0.95
MAXIMUM LEARNING RATE	$6e-4$
MINIMUM LEARNING RATE	$6e-5$
DECAY LEARNING RATE	TRUE
WARMUP ITERATIONS	2000
NUMBER OF LEARNING RATE DECAY ITERATIONS	4.2 M
VALUE TO CLIP GRADIENTS AT	1.0
DROPOUT	0.1
TASK PROBABILITY p_{task}	0.95

E.3 Training

We provide the complete list of hyperparameters used for training JOINT TRANSFORMER in Table 5. JOINT TRANSFORMER was trained on a single NVIDIA GeForce RTX 2080 TI GPU for 4.2M iterations that took approximately seven days.

E.4 Fine-tuning

As JOINT TRANSFORMER is a joint model, fine-tuning is achieved by standard training (Alg. 3) on the supervised part of the dataset. Unless stated otherwise, we use the same set of hyperparameters for fine-tuning across all tasks, summarized in Table 6. Fine-tuning on a single NVIDIA GeForce

381 RTX 2080 TI GPU for 50K iterations takes approximately an hour. Hyperparameters not listed in
 382 Table 6 are shared with the pre-training task.

Table 6: Fine-tuning hyperparameters for the JOINT TRANSFORMER used across all experiments.

HYPERPARAMETER	VALUE
DECAY LR	FALSE
LEARNING RATE	$3e-5$
NUM OF ITERATION	50K
TASK PROBABILITY p_{task}	0.1