ATML Report - CMVAE

Elvi Mihai Sabau Sabau *sabaue@usi.ch*
Fabian Gobet *gobetf@usi.ch*
Pietro Miotto *miottpi@usi.ch*
Yassine Oueslati *ouesly@usi.ch*

Abstract

Our study reproduces and validates the findings of Deep Generative Clustering Multimodal Variational Autoencoders (CMVAE) Palumbo et al. (2024). CMVAE extends multimodal VAEs by introducing a latent clustering structure and sets the ground for integrating diffusion models in order to improve generative performance.

The original paper is well-documented, but its brevity ends up in the omission of several key concepts and implementation details (not present even in the Appendix). n addition, although the provided code is well-documented and structured, it includes numerous elements that remain unused. Nonetheless, the most significant challenge in replicating the results was managing the unexpectedly extensive and resource-intensive training periods. As we will show below, this resulted in enormous restrictions, both in time and economic resources, which, in the end, made this reproduction harder than expected. Lastly, our reproduction highlights the absence of a stable diffusion implementation, which could further explore the model's potential. The paper only provides mathematical insights on how to integrate that, but does not define any other relevant implementation detail (e.g. model and hyperparameters). Despite the above, our findings support CMVAE's robustness and suggest areas for refinement.

1 Introduction

Our work centered on reproducing a complex research study. To address its theoretical and implementation challenges, we closely examined and manually revised the underlying formulas and algorithms, while also creating detailed schematics and diagrams guided by the existing code structure. After gaining a thorough understanding of the system's mechanisms, we resolved minor implementation issues before commencing the computationally intensive training process. As shown in Figure 30, achieving a substantial, consistent reproduction of the paper's metrics demanded significant resources: training the model for PolyMnist alone required approximately 170 hours and incurred around $120 in GPU costs. Evaluating the impact of any code modification would have added up to three days of training time and substantial additional expenses.

2 Scope of reproducibility

The paper addresses the intersection of multimodal variational autoencoders (VAEs) and clustering tasks, proposing a novel model, CMVAE, to improve generative and clustering performance in multimodal settings. This method leverages shared latent representations across modalities and integrates recent advancements in diffusion models to address limitations of existing VAE-based methods in generative quality and clustering scalability.

2.1 Main Claims

- **Claim 1. Novel Model for Multimodal Clustering:** The paper introduces the Clustering Multimodal VAE (CMVAE), which enhances the clustering capability by enforcing a mixture distribution in the shared latent space. It demonstrates superior performance from other models Suzuki et al. (2017), Vedantam et al. (2018), Palumbo et al. (2023), Caron et al. (2018), Jiang et al. (2017) in unconditional generation and weakly-supervised clustering compared to existing methods.
- **Claim 2. Post-Hoc Clustering Optimization**:

  CMVAE employs an innovative post-hoc procedure to infer the optimal number of clusters at test time, overcoming the necessity to define the number of clusters during training. This approach reduces entropy in cluster assignments and ensures effective modeling of latent clusters.
- **Claim 3. Integration of Diffusion Models:** The paper describes how incorporation of Denoising Diffusion Probabilistic Models (DDPMs) could improve the generative performance of real-world multimodal data.

3 Methodology

We used the following resources:

- Code: Github repository provided by paper's authors at `https://github.com/epalu/CMVAE/tree/main`, altought we had to modify it due to deprecated functions, unsolved issues and adaptability to different hardware (see Section 5.2)
- Hardware: We were able to run the code and evaluate results by using:
  - **Lighting.ai** using NVIDIA T4 Tensor Core GPU (16GB RAM of memory), we used up to 238 credits ($\tilde{3}$00 hours of compute depending on the GPU).
  - **MAC Book Laptops M2** Due to time constrains and to the time it takes to train the models, we were forced to adapted the code to be able to also train in parallel in our mac-books: Apple M2 Pro 32GB RAM 19Cores
  - **Dekstop Computers** Elvi's desktop computer: RTX 3070 8GB; Fabian's desktop computer: RTX 3070ti 8GB
- Other Platforms: We used **Weight and Biases (Wandb)**[1] to log and have a centralized way to store the results of each model and its hyperparameters. We started by cloning the aforementioned repository and run it with the default hyperparameters (see Section 3.3).

3.1 Model descriptions

The Clustering Multimodal Variational Autoencoder (CMVAE) integrates clustering into a multimodal variational autoencoder, learning shared (z) and modality-specific (w_m) latent spaces while enforcing semantic clustering with a Gaussian mixture prior (c). Each modality has an encoder that extracts latent parameters (μ, σ^2) and a decoder that outputs the parameters for the distribution of the reconstructed data. The training objective maximizes reconstruction quality, aligns latent spaces across modalities, and penalizes deviations from priors. A cross-modality generation mechanism ensures consistency, promoting shared semantic understanding across latent spaces. During training, posteriors and priors are iteratively optimized to enforce clustering. After training, a post-hoc entropy minimization step refines clusters. Inference allows for unconditional generation of synthetic data by sampling from the learned latent spaces, or conditional generation using self-and-cross-modality prior distributions $q(z, w|x)$ and posterior distributions $p(\hat{x}|z, w)$, with learned respective latent spaces. A more detailed overview of the training process may be found in section (see Section A.1).

[1] `https://wandb.ai`

3.2 Datasets

3.2.1 PolyMNIST: Handwritten Digits

The PolyMNIST Dataset, which contains 5 different variations of the MNIST Dataset, is a famously known dataset of handwritten digits from 0 to 9. Each modality depicts MNIST digits patched on random crops from five distinct background images, one for each modality. This means that both the Train and Test PolyMNIST Datasets are structured as follows.

- `m0`: contains 12.000 images for training, 2000 for testing, of handwritten digits edited (as explained above)according to modality 0.
- `m1`: contains 12.000 for training, 2000 for testing, of handwritten digits edited (as explained above)according to modality 1.
- and so on ... up to `m4`

For a descriptive idea of the dataset, revise figure 3.

More specifically, the class `PolyMNISTDataset(Dataset)` handles both the creation of modalities from the original MNIST set and the construction of train and test datasets. The `__getitem__` function returns a tuple formatted as `(images, labels)`, where `images` is an array containing, in each position, another array which stores all the images for a specific modality, i.e. `images[0]` contains, as image tensors, the entirety of `m0` content. The labels are directly inferred from the filename: since in every modality folder the images are named as `id_digit.png`, we can extract the `digit` directly from the filename and set it as a label describing what digit `id_digit.png` image represents.

For the training set, this tuple is kept as returned by `__getitem__`: the training set thus contains $12.000 \times 5 = 60000$ image tensors and labels. Instead, the $2000 \times 5 = 10000$ images of the training set are equally split into test and validation set. Please note that the image tensors are of size `(3, 28, 28)`.

3.2.2 CUBICC: grouped by subspecies CUB images

The **CUBICC dataset** is a variation of the CUB Image-Captions dataset, derived from the Caltech-UCSD Birds (CUB) dataset. This variation organizes subspecies of birds from the original dataset into eight broader species categories, creating a more challenging and realistic multimodal clustering dataset.

As illustrated in the example image, the CUBICC dataset contains 13,150 image-caption paired samples from 22 subspecies, grouped into the following eight species: **Blackbird, Gull, Jay, Oriole, Tanager, Tern, Warbler, and Wren**. Additional details can be seen in Figures 1 and 2.

The dataset consists of pre-grouped images, captions, and labels, which are pre-split into training, validation, and testing sets. These splits are stored in specific `.npy` files, using NumPy arrays for efficient data storage and rapid loading during experimentation.

3.3 Hyperparameters

By cross-referencing the paper and the provided code, we identified the default hyperparameters, which are specified in Table 2.

Note: Rows highlighted in gray indicate parameters that are *hardcoded*, meaning they cannot be modified in the runtime specifications and are defined directly within the code. Non-relevant parameters, such as file paths or project names, have been excluded from the table for clarity.

3.4 Experimental setup and code

The original paper's code repository includes a detailed guide for setting up the experiments. Following the instructions, we downloaded the datasets and extracted their contents directly into the designated data directory. To train the models, a provided bash script facilitates the process by allowing configuration of input parameters such as the number of initial clusters, the dimensionality of each latent space, the use of CUDA for computation, and other relevant settings.

The experiments were evaluated using a comprehensive set of measures designed to assess various aspects of the model's performance and its representations. Below are the specific measures used:

3.4.1 Reconstruction Loss

Reconstruction loss evaluates how well the model reconstructs the input data given the latent representations. This is quantified using metrics such as negative log-likelihood or the evidence lower bound (ELBO), depending on the selected training objective (e.g., IWAE or DReG).

3.4.2 Conditional Coherence

Conditional coherence measures the consistency of cross-modal generations. It evaluates how well the model generates data in one modality conditioned on latent representations inferred from another modality. The metric used is the proportion of correctly matched predictions across modalities, aggregated into a cross-coherence matrix and averaged.

3.4.3 Unconditional Coherence

Unconditional coherence evaluates the semantic consistency of unconditional generations across all modalities. The metric is the proportion of generated samples with consistent class predictions across modalities, where consistency is defined as all modalities agreeing on the same class label.

3.4.4 Latent Representation Classification

This measure evaluates the quality of the learned latent representations for downstream tasks. Classification accuracy is computed using logistic regression (linear classifier) and neural networks (non-linear classifier) trained on the latent representations for both shared (z) and modality-specific (w) spaces.

3.4.5 FID (Fréchet Inception Distance)

The Fréchet Inception Distance assesses the quality of generated images by comparing the distributions of real and generated images in a feature space. FID scores are computed for both unconditional and conditional generations, with lower scores indicating higher similarity between the real and generated data distributions.

3.4.6 Training and Test Loss

Training and test loss monitor the optimization process and generalization ability of the model. The metric used is the average loss per sample on the training and validation datasets, logged after each epoch.

3.4.7 Generative Sample Evaluation

This measure qualitatively assesses the quality of unconditional and conditional samples. Generated samples are visually inspected, and self-reconstruction and cross-modal generation are visualized by logging the images.

3.4.8 Timing and Computational Resources

We trained using our hardware, and also lightning.ai's platform. The training process for all four models (including the time spent on failed trained instances) required approximately 340 hours, utilizing L4 and T4 GPUs, with a total cost of $238.20, which we got from different peers. Since each student gets 30 credits each month, we asked our colleges to donate their credits to us.

For the models trained with CUBICC, it would take around 4 to 5 minutes per epoch, each evaluation step done every 25th epoch takes around 3 minutes. For the model trained with PolyMNIST, each epoch would take around 15 minutes, each evaluation step done every epoch which would take around 7 minutes.

3.5 Computational requirements

During our reproducibility task, we observed that training under the same configuration provided in the CMVAE paper required approximately 30 hours of compute time per CUBICC model, and around 75 hours per Polymnist model. This was conducted on a T4 GPU with 8 CPUs, 16GB of RAM, and 65 TFlops of computational power. The extended training time is attributed to a challenge discussed in Section 5.2 of this report. Additionally, we attempted to adapt the code to newer versions of Python and its associated libraries to enable compatibility with modern hardware, such as the RTX 3070. However, despite addressing numerous issues and warnings, the results deviated from those obtained using the original setup.

4 Results

Our current training procedure for the CUBICC dataset includes only the first three models, excluding the configuration with $K = 40$. Meanwhile, for the PolyMNIST dataset, we have only trained the first model with $\beta = 1.0$ (see Table 1). Throughout these training sessions, we have consistently observed that the validation loss increases across all cases (see Figures 4 and 17). This outcome arises despite employing the same training setup and number of epochs reported in the original study.

We also provide online Wandb reports which allow easy navigation of all the results for PolyMnist[2] and CUBICC[3] experiments.

In the subsequent results, we aim to validate the first two claims (see Section 2): specifically, that the proposed implementation outperforms existing methods, and that the post-hoc clustering approach effectively reduces entropy in cluster assignments while ensuring the accurate modeling of latent clusters. We were unable to validate the third and final claim due to the absence of the necessary code and implementation in the materials provided by the paper.

4.1 Results for CUBICC

4.1.1 FID comparison

When comparing our results across different modalities (see Figures 5, 6, 7) with the results reported in the paper (see Figure 38), it is challenging to draw definitive conclusions. This limitation arises because the results presented in the paper represent an average over multiple models trained with varying random seeds and include both unconditional and conditional structures. In contrast, due to time constraints, we were only able to train a single instance for each parameter configuration.

4.1.2 Qualitative results

Unconditional Generation

In contrast to the PolyMNIST models, unconditional generation on the CUBICC dataset pro-

[2] `https://api.wandb.ai/links/frenzoid-atml/zu7625oi`
[3] `https://api.wandb.ai/links/frenzoid-atml/gmy6uekn`

duces low-quality results, with undefined textures and blended colors, as shown in Figure 8. This outcome aligns with the research paper's observations in Figure 33, where the authors employ a Denoising Diffusion Probabilistic Model (DDPM) to address these limitations. This result highlights that the inclusion of diverse modality types impacts the model's generative quality performance.

4.1.3 Clustering Performance

The clustering performance was evaluated using metrics such as Normalized Mutual Information (NMI), Accuracy (ACC), and Adjusted Rand Index (ARI) on validation and test datasets. Our experiments analyzed the impact of the regularization parameter β (1.0, 2.5, 5.0), focusing on the number of clusters retained after pruning and corresponding performance metrics.

As shown in our results (Figure 13), the number of clusters after pruning varied significantly: $\beta = 1.0$ retained 19 clusters, $\beta = 2.5$ retained 8 clusters, and $\beta = 5.0$ retained 2 clusters. Validation Penalized Norm Entropy (PNE) consistently decreased with pruning, confirming the model's ability to remove insignificant clusters. Test metrics (Figure 9, 10, 11 and 12) peaked at $\beta = 2.5$, achieving NMI of 0.6359, ACC of 0.55, and ARI of 0.54, indicating an optimal balance of latent space structure and clustering accuracy.

Unlike the original study (Figure 35, 37), which averaged results over multiple runs with varying random seeds, we trained single instances for each configuration due to time constraints. Despite this limitation, our findings demonstrate that moderate regularization ($\beta = 2.5$) aligns closely with trends reported in the original work and yields robust performance.

4.2 Results for PolyMNIST

4.2.1 Qualitative results

Conditional Generation
Figure 19 demonstrates strong cross-modality consistency in conditional generation, aligning with the observations in Figure 32. The same digit is consistently generated across modalities while preserving modality-specific characteristics such as color, texture, and style.

Unconditional Generation
Unconditional generation results, shown in Figure 18, highlight the model's ability to produce diverse and realistic samples across modalities. These results are consistent with the findings in Figure 31, where the generated digits exhibit high visual quality and distinct modality-specific traits.

Summary
The model demonstrates high visual fidelity, strong disentanglement of shared and modality-specific features, and diversity in both conditional and unconditional settings, validating its expected suitability for multimodal generative tasks.

4.2.2 Coherence comparison

When comparing the cross coherence of clustering results presented in Figure 34 of the referenced paper with those obtained from our training (see Figures 20, 21, 22, 23, 24, and 25), we observe that the coherence levels are consistent with the findings reported in the paper. This consistency indicates that our model effectively clusters data of the same kind. Furthermore, these results provide empirical support for the paper's primary claim (see Section 2.1), demonstrating that the proposed model enhances clustering performance by enforcing a mixture distribution within the shared latent space.

4.3 Clustering Performance

We evaluated clustering performance on the PolyMNIST dataset using metrics such as NMI, ACC, ARI, nad PNE. Due to time constraints, we trained only a single model with $\beta = 1.0$.

Our results (Figure 26, 27,28, 29) are consistent with the trends reported in the original study (Figure 36). While the x-axis in our graphs reflects the number of clusters pruned, compared to the number of actual clusters in the original study, the shape of the graphs is identical. This alignment indicates that our results match the reported performance despite training only a single model instead of averaging over multiple runs with different random seeds.

In our experiment, as the number of clusters was pruned, validation PNE decreased steadily, confirming the effective removal of less significant clusters. NMI, ACC, and ARI peaked when the optimal number of clusters was reached, demonstrating strong alignment with true latent structures. Validation NMI exceeded 0.8, and ACC reached over 0.85, highlighting the effectiveness of our single-instance model in capturing meaningful clusters.

The model successfully identified the optimal number of clusters (K=10) for the PolyMNIST dataset, demonstrating its ability to effectively discover the true latent structure as reported in the original study.

4.4 Results beyond original paper

4.4.1 Additional Result - CUBICC

For training with the CUBICC dataset, the following parameters must be used (for all models): Objective: dreg, batch size: 32, Preposterior: Normal, Optimizer: Adam

4.4.2 Additional Result - PolyMNIST

For training with the PolyMNIST dataset, the following parameters must be used (for all models): Objective: dreg, batch size: 128, Preposterior: Laplace, Optimizer: Adam

5 Discussion

5.1 What was easy

The code is well-documented with clear explanations and annotations, making it easy to read and understand. It closely follows the methodologies and concepts outlined in the associated research paper, effectively bridging the gap between theory and implementation.

The code is structured around Object-Oriented Programming (OOP) principles, with well-organized classes and modular components, ensuring a clear organization of logic while promoting reusability, scalability, and readability.

5.2 What was difficult

While attempting to replicate the paper's findings, we encountered multiple obstacles. We addressed code-related issues and documented our fixes in a forked repository.[4]

Data preparation remained ambiguous due to insufficient dataset instructions. Although the authors referenced a dataset, they provided no clear directory structures.

The described computational environment was outdated, causing errors on contemporary NVIDIA GPUs (L4, RTX 3070) due to old libraries and Python dependencies. We resorted to using T4 GPUs, which slowed experimentation. Training the *cubicc* model with $K = 40$ demanded high VRAM—only available on newer hardware—further complicating replication.

Parameter specifications were scattered across the paper, forcing extensive cross-referencing. The code's reliance on CUDA-specific components also limited adaptability and parallelization. We uncovered anomalies in the code—variables that were computed but never used, undefined variables in training scripts—that made pruning initially impossible.

[4] `https://github.com/Frenzoid/CMVAE`

The paper did not explore whether qualitative results could enhance performance on the Cubicc dataset, even though they significantly improved results on PolyMNIST. Similarly, stable diffusion results were reported but not implemented in the codebase.

Finally, the approach to modeling conditional dependencies—using complex variational inference techniques—lacked detailed derivations. Reconstructing these steps from scratch (see Section A.1.2) proved challenging, adding complexity to fully interpreting the paper.

5.3 Communication with original authors

We reached out to the authors of the paper with specific inquiries regarding topics such as how the data generation process works, the influence of individual latent spaces on the shared latent space, the semantics of the outputs from each VAE decoder, diffusion handling, training times, and hardware or optimization advice. Unfortunately, we have not received a response from their side.

6 Conclusion

This work allowed us to grasp extremely relevant insights on how state of the art variations of VAEs can be implemented and developed. Thus, from a theorical point on view, this configured as a springboard for acquiring an in-depth understanding of extremely complex topics and implementations, extending our knowledge of this area of research. We do not refer only to the mathematics and the key ideas behind the paper, but also on how such complex theorical achievements can be implemented in code. On the other hand, the reproducibility of the paper was limited from the beginning by the extensive training times and resources needed. We were able to set-up the original code so that it could run and produce relevant outcomes, but this came with extremely high expenses of time and resources. We also provide a GitHub fork to assist future extension projects with a cold start. The primary focus of further advancements should be on optimizing the code to align with, and ideally surpass, the results reported in the paper. Please be aware that what mentioned above could be possible only if adequate resources are found.

Member contributions

- Elvi Mihai Sabau Sabau: Contributed to replicating the paper's code and worked on training the CUBICC and Polymnist models. Resolved pre-existing issues in the codebase and adapted the implementation to work with the latest library versions. Also communicated with the paper's authors for clarifications and contributed to writing the report.

- Fabian Gobet: Contributed to replicating the paper's code by analyzing its implementation and ensuring consistency with the theoretical framework described in the publication. Worked on reviewing the code to identify and resolve redundancies and technical issues, improving its functionality. Assisted in validating the implementation against the underlying theory and contributed to the preparation and writing of the report.

- Yassine Oueslati: Contributed to replicating the paper's code by reviewing its implementation for consistency with the theoretical framework, particularly the mathematical components. Worked on analyzing and documenting the code, including creating diagrams, and assisted in validating the implementation against the theory. Also contributed to the preparation and writing of the report.

- Pietro Miotto: Contributed to the report content and overall consistency. Worked on analyzing and reviewing the code and implementation by also creating diagrams, to help overall understanding. Focused also on datasets handling and creation.

References

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, 2018.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

Emanuele Palumbo, Imant Daunhawer, and Julia E Vogt. Mmvae+: Enhancing the generative quality of multimodal vaes without compromises. In *International Conference on Learning Representations (ICLR)*, 2023.

Emanuele Palumbo, Laura Manduchi, Sonia Laguna, Daphné Chopard, and Julia E Vogt. Deep generative clustering with multimodal diffusion variational autoencoders. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=k5THrhXDV3`.

Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*, 2017.

Ramakrishna Vedantam, Ian Fischer, Jonathan Huang, and Kevin Murphy. Generative models of visually grounded imagination. In *International Conference on Learning Representations (ICLR)*, 2018.

A Model Explaination, Citations, figures, and tables

A.1 Model descriptions

The models we found that were trained are 4 models with CUBICC and 4 models for PolyMNIST with the following generic parameters shown in this table 1.

Unfortunately, as discussed in Chapter 5.2, the models with $K = 40$ were not trained due to the extensive training times required.

A.1.1 Architecure of Clustering Multimodal Variational Autoencoder (CMVAE)

For this section we will introduce the general architecture of all the model then we will describe the mechanism.

The Clustering Multimodal Variational Autoencoder (CMVAE) is designed to learn shared and modality-specific latent representations while integrating a clustering mechanism in the latent space. The architecture aims to correlate different classes between different data modalities, and it consists of the following components:

Latent Space Structure

- **Shared Embeddings** (z): Represents the common information shared across all modalities.
- **Modality-Specific Embeddings** (w_m): Captures unique features of each modality m.
- **Clustering Prior** (c): A categorical distribution formed by the imposition of a mixture model (e.g. Gaussian Mixture Model) in the space of $Z \mid C$

Encoder Network

- Each modality m has its own encoder $E_m(x_m)$, which takes as input the raw data x_m of that modality.
- **Outputs:**
  - Parameters of the shared latent variable z: μ_z and σ_z^2
  - Parameters of the modality-specific latent variable w_m: μ_{w_m} and $\sigma_{w_m}^2$

Decoder Network

- Each modality m has a corresponding decoder $D_m(z, w_m)$.
- **Inputs:** The shared latent variable z and the modality-specific latent variable w_m.
- **Outputs:** Parameters of the distribution for the reconstructed data $\hat{x}_m$: $\mu_{\hat{x}_m}$ and $\sigma_{\hat{x}_m}^2$

Clustering Mechanism

- A Gaussian mixture prior is used in the shared latent space z, which encourages the formation of distinct clusters during training.
- Clustering assignments c are represented by a categorical distribution, and the model optimizes for cluster consistency using a post-hoc entropy minimization procedure.

A.1.2 Mechanism of Training and Inference

Below we provide a step by step explanation of how the training process works, and we explain all of its relevant components.

Objective Function

The objective function mentioned in the paper is the following:

$$\mathcal{L}_{\text{CMVAE}}(\boldsymbol{X}) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\substack{q(\boldsymbol{c}|\boldsymbol{z},\boldsymbol{X}) \\ q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}|\boldsymbol{X}) \\ q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m|\boldsymbol{x}_m)}} \left[G_{\boldsymbol{\pi},\Phi_{\boldsymbol{z}},\phi_{\boldsymbol{w}_m},\Theta}(\boldsymbol{X},\boldsymbol{c},\boldsymbol{z},\boldsymbol{w}_m)\right], \tag{1}$$

where

$$G_{\boldsymbol{\pi},\Phi_{\boldsymbol{z}},\phi_{\boldsymbol{w}_m},\Theta}(\boldsymbol{X},\boldsymbol{c},\boldsymbol{z},\boldsymbol{w}_m) = \log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z},\boldsymbol{w}_m) + \sum_{n\neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n\sim r_n(\boldsymbol{w}_n)}\left[\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z},\tilde{\boldsymbol{w}}_n)\right] + \beta \log \frac{p_\pi(\boldsymbol{c})\, p_\theta(\boldsymbol{z}\mid\boldsymbol{c})\, p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}\mid\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m\mid\boldsymbol{x}_m)\, q(\boldsymbol{c}\mid\boldsymbol{z},\boldsymbol{X})} \tag{2}$$

Hence, from **(1)** and **(2)** follows:

$$\begin{aligned}
\mathcal{L}_{\text{CMVAE}}(\boldsymbol{X}) =& \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\substack{q(\boldsymbol{c}|\boldsymbol{z},\boldsymbol{X}) \\ q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}|\boldsymbol{X}) \\ q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m|\boldsymbol{x}_m)}} \left[\log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z},\boldsymbol{w}_m) + \sum_{n\neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n\sim r_n(\boldsymbol{w}_n)}\left[\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z},\tilde{\boldsymbol{w}}_n)\right] + \beta \log\left(\frac{p_\pi(\boldsymbol{c})\, p_\theta(\boldsymbol{z}\mid\boldsymbol{c})\, p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}\mid\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m\mid\boldsymbol{x}_m)\, q(\boldsymbol{c}\mid\boldsymbol{z},\boldsymbol{X})}\right)\right] \\
=& \frac{1}{M}\sum_{m=1}^{M} \left(\log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z},\boldsymbol{w}_m) + \sum_{n\neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n\sim r_n(\boldsymbol{w}_n)}\left[\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z},\tilde{\boldsymbol{w}}_n)\right] + \beta\, \mathbb{E}_{\substack{q(\boldsymbol{c}|\boldsymbol{z},\boldsymbol{X}) \\ q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}|\boldsymbol{X}) \\ q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m|\boldsymbol{x}_m)}} \left[\log\left(\frac{p_\pi(\boldsymbol{c})\, p_\theta(\boldsymbol{z}\mid\boldsymbol{c})\, p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}\mid\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m\mid\boldsymbol{x}_m)\, q(\boldsymbol{c}\mid\boldsymbol{z},\boldsymbol{X})}\right)\right] \right) \\
=& \frac{1}{M}\sum_{m=1}^{M} \left(\log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z},\boldsymbol{w}_m) + \sum_{n\neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n\sim r_n(\boldsymbol{w}_n)}\left[\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z},\tilde{\boldsymbol{w}}_n)\right] + \beta \left(\log\left(\frac{p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}\mid\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m\mid\boldsymbol{x}_m)}\right) \mathbb{E}_{\substack{q(\boldsymbol{c}|\boldsymbol{z},\boldsymbol{X}) \\ q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}|\boldsymbol{X}) \\ q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m|\boldsymbol{x}_m)}} \left[\log\left(\frac{p_\pi(\boldsymbol{c})\, p_\theta(\boldsymbol{z}\mid\boldsymbol{c})}{q(\boldsymbol{c}\mid\boldsymbol{z},\boldsymbol{X})}\right)\right] \right) \right) \\
=& \frac{1}{M}\sum_{m=1}^{M} \left(\log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z},\boldsymbol{w}_m) + \sum_{n\neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n\sim r_n(\boldsymbol{w}_n)}\left[\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z},\tilde{\boldsymbol{w}}_n)\right] + \beta \left(\log\left(\frac{p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}\mid\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m\mid\boldsymbol{x}_m)}\right) \sum_{c_i\in C} \left[\log\left(\frac{p_\pi(\boldsymbol{c_i})\, p_\theta(\boldsymbol{z}\mid\boldsymbol{c_i})}{q(\boldsymbol{c_i}\mid\boldsymbol{z},\boldsymbol{X})}\right) p(\boldsymbol{c_i}\mid\boldsymbol{z})\right] \right) \right)
\end{aligned}$$

Relating back to the code and it's variables, we have that:

$lpx\_u \equiv \log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z},\boldsymbol{w}_m) + \sum_{n\neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n\sim r_n(\boldsymbol{w}_n)}\left[\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z},\tilde{\boldsymbol{w}}_n)\right]$

$lpw - lqz\_x - lqw\_x \equiv \log\left(\frac{p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}|\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m|\boldsymbol{x}_m)}\right)$

$((lpc + lpz\_c - lpc\_z) * pc\_z).sum(-1) \equiv \sum_{c_i\in C}\left[\log\left(\frac{p_\pi(\boldsymbol{c_i})\, p_\theta(\boldsymbol{z}|\boldsymbol{c_i})}{q(\boldsymbol{c_i}|\boldsymbol{z},\boldsymbol{X})}\right) p(\boldsymbol{c_i}\mid\boldsymbol{z})\right]$

- The first term in G, $\log p_{\theta_m}(\boldsymbol{x}_m \mid \boldsymbol{z}, \boldsymbol{w}_m)$, measures how well the model reconstructs $\boldsymbol{x}_m$ given the latent variables $\boldsymbol{z}$ and $\boldsymbol{w}_m$.
- The second term in G, $\sum_{n \neq m} \mathbb{E}_{\tilde{\boldsymbol{w}}_n \sim r_n(\boldsymbol{w}_n)} [\log p_{\theta_n}(\boldsymbol{x}_n \mid \boldsymbol{z}, \tilde{\boldsymbol{w}}_n)]$, accounts for the log-likelihood of cross-modal reconstructions, ensuring a congruent mapping of all modalities into the same region in the shared latent space z.
- The final term in G, $\beta \log \frac{p_\pi(\boldsymbol{c})\, p_\theta(\boldsymbol{z}|\boldsymbol{c})\, p(\boldsymbol{w}_m)}{q_{\Phi_{\boldsymbol{z}}}(\boldsymbol{z}|\boldsymbol{X})\, q_{\phi_{\boldsymbol{w}_m}}(\boldsymbol{w}_m|\boldsymbol{x}_m)\, q(\boldsymbol{c}|\boldsymbol{z},\boldsymbol{X})}$, penalizes deviations of the learned posterior distributions from their priors. This can be formulated as an approximation, as proposed in the work of (Jiang et al., 2017), which has the advantage of not requiring additional parameters by using an evidence-based factor, as: $q(\boldsymbol{c} \mid \boldsymbol{z}, \boldsymbol{X}) = p(\boldsymbol{c} \mid \boldsymbol{z}) = \frac{p(\boldsymbol{c})p(\boldsymbol{z}|\boldsymbol{c})}{\sum_{\boldsymbol{c}'=1}^{K} p(\boldsymbol{c}')p(\boldsymbol{z}|\boldsymbol{c}')}$.

The provided objective function equation 1 cannot be directly computed. The paper omits many fundamental steps which allows us to use it in the code, namely the evidence-based normalization factor $q(c|z, X)$ [5]. As shown above, we were able to manually compute and retrieve these steps, by also looking at the code.

Evidence-based factor and Softmax:

$$P(Z|C)\,P(C) \;=\; P(Z,C) \;=\; P(C|Z)\,P(Z) \;\Rightarrow\; P(C|Z) \;=\; \frac{P(Z|C)\,P(C)}{P(Z)} \tag{3}$$

$$P(Z) \;=\; \sum_{c_i \in C} P(Z|c_i)\,P(c_i) \tag{4}$$

$$Softmax(x_i) \;=\; \frac{e^{x_i}}{\sum_j e^{x_j}}, \quad Softmax(X) \;=\; \frac{e^{X}}{\sum_j e^{x_j}}$$

Let $x_j \;=\; \log(P(Z, c_j))$ and $X \;=\; \log(P(Z, C))$, then:

$$\begin{aligned} Softmax(X) \;&=\; \frac{e^{\log(P(Z,C))}}{\sum_{c_j \in C} P(Z, c_j)} \\ &=\; \frac{P(Z,C)}{\sum_{c_j \in C} P(Z, c_j)} \end{aligned} \tag{5}$$

From **(3)**, **(4)** and **(5)** follows:

$$Softmax(X) \;=\; P(C|Z)$$

□

Training Steps:

- Input data is processed by the respective modality Variational Auto-Encoder (VAE), saving the latent spaces paramaters for the prior returned by the enconder into a temporary variable $us_m(w_m, z_m|x_m)$
- From the distribution of the former paramaters $q_m(us_m|x_m)$, K samples are drawn, each one passed through the encoder to get the parameters for the posterior distribution, and then used to built said distribution $p_{m,k}(\hat{x}_m|us_m)$.
- Each VAE has modality-specific latent space paramaters w_m, different from the ones computed from the data in the former steps, whose mean is fixed at 0 and variance can be adjusted during training. This entails the real prior distribution paramaters of the latent space w with respect to VAE m ($\hat{w}_m$).
- For each computed $us_m(w_m, z_m|x_m)$, we extract the parameters z_m, and for all VAE d, such that $d \neq m$:

[5] Jiang et al. (2017)

  - We compute the prior distribution using $\hat{w}_m$ paramaters.
  - Sample K elements from the former prior distribution
  - Each sample is concatenated with z_m, passing through VAE d decoder and finally used to build the posterior distribution $p_{m,k}(\hat{x}_m|\hat{w}_d, z_m)$

- NOTE: the former step is known as cross-modality generation. The goal here is to modulate $\hat{w}_m$ in such away that it upholds the same semantic context within the true shared latent space z for each VAE. By doing this and enforcing a multivariate distribution in the priors, the model effectively learns how to cluster semantic related inputs in the shared latent space.
- Each of the former distributions is saved into a matrix whose entries i, j can be interpreted as the posteriors using modality-specific latent variables $\hat{w}_i$ and shared latent variables z_j. Furthermore, in the main diagonal of this matrix, the modality-specific latent variables come from the data, as opposed to the other entries.
- At this point, for each VAE we $us_m(w_m, z_m|x_m)$ stored in a list, and self-and-cross-modal generation distributions $p(\hat{x}|us)$ in a matrix.
- For each VAE we extract the saved parameters from the first step, split the parameters into z_m and w_m latent space parameters and compute the prior distributions $q_m(z_m|x_m)$ and $q_m(w_m|x_m)$, storing them in separate lists.
- For each of the computed $q_m(z_m|x_m)$
  - We fetch $us_m(w_m, z_m|x_m)$ and split it into z_m and w_m latent space parameters.
  - We fetch the trainable C space parameters (which follow a categorical distribution), and convert them into their logarithm.
  - The model stores trainable parameters μ_z and σ_Z^2, under the conditional distribution $Z|C$, e.g. μ_z is of size $dim(C) \times dim(Z)$. For each C, we fetch the correspondent parameters $\mu_{z|c}$ and $\sigma_{z|c}^2$, compute the posterior distribution and the log probability of z_m, finally summing over all Z's. This gives us a stack of elements which represents $\log p(z|c)$.
  - By summing $\log p(c)$ and $\log p(z_m|c)$, we get the unnormalized posterior $\log p(c|z)$. If we then perform a Softmax on this result, we can calculate the evidence-factor-based distribution $p(c|z)$ as formulated in A.1.2, which we then can apply the logarithm to get the normalized $\log p(c|z)$.
  - The model contains non-trainable parameters w, fixed at $\mu_w = 0$ and $\sigma_w^2 = 0$. A prior distribution is computed using said parameters and the log probability of w_m is calculated, then summing of all parameters of the result. This gives us $\log p(w)$ and enforces the clustering of semantic related inputs.
  - We then calculate the log probability of z_m over all $q_d(z_d|x_d$ and sum all the z's for each resulting vector. The results are then stacked and passed through a function that calculates the logarithm of mean of exponential's, effectively giving us $\log q(z_m|X)$.
  - We calculate the log probability of w_m under $q_m(w_m|x_m)$ and sum all the values of the resulting vector, effectively giving us $\log q(w|x)$
  - Finally, we iterate through all $p(\hat{x}_d|us_d)$ and we compute the log probability of x_d, then summing through all the resulting elements of each vector. This results stacked and summed, finally providing us with $\log p(x|z, w)$.
  - We now have all the elements to compute the formula inA.1.2. This result is added to a list.

- Having computed all results, we calculated the mean over the dimension of clusters and sum the results, effectively giving us the Importance Weighted Auto-encoder value.

Post-Hoc Clustering:

- After training, a post-hoc entropy minimization step identifies the true number of clusters and prunes redundant ones.

Inference Mechanism:

1. Unconditional Generation:

- There are two approaches to unconditional generation: the *random* and the *normal* methods. In the random approach, we first build a categorical distribution using the parameters of the space C, and then sample from it. Thus, when we proceed to sample $\mathbf{Z}$ from the conditional space $\mathbf{Z} \mid C$, $\mathbf{Z}$ is already conditioned on a specific cluster. In the normal approach, however, we sample $\mathbf{Z}$ independently for each cluster C. As a result, each modality returns as many outputs as the cardinality of the space C.

- Samples are drawn from the latent prior distribution ($\mathbf{z}$).

- For each VAE, its parameters ($\mathbf{W}$) are retrieved and used to build the prior distribution.

- Samples are drawn from this distribution, concatenated with the previously sampled $\mathbf{Z}$-vectors from earlier steps, and passed through the VAE decoder to obtain the parameters of the posterior distribution.

- The posterior distribution for that VAE is then constructed, and its mean is retrieved. This mean corresponds to the most probable output under the posterior distribution.

2. Conditional Generation:

- For each input $x^{(m)}$, we pass it through the corresponding VAE encoder to obtain an approximate posterior distribution over the latent space and sample a latent vector $\mathbf{u}^{(m)}$. This latent vector typically decomposes into a private component $\mathbf{w}^{(m)}$ and a shared component $\mathbf{z}$.

- Then, we use each modality's own decoder to reconstruct its input from the sampled latent code, producing a self-reconstruction. This self-reconstruction forms the diagonal entries of a reconstruction matrix, where row and column indices correspond to modalities.

- For cross-modal generation, we select a source modality e. From its latent vector $\mathbf{u}^{(e)}$, we isolate the shared latent component $\mathbf{z}_e$.

- We then choose a target modality $d \neq e$. From the target modality's prior distribution, sample a new private latent vector $\mathbf{w}^{(d)}$. This ensures that the generated sample reflects modality d's domain-specific characteristics.

- We concatenate $\mathbf{w}^{(d)}$ with $\mathbf{z}_e$ to form a cross-modal latent code. Pass this combined latent vector through the decoder of modality d to generate a new sample. This sample is now "conditioned" on the shared representation derived from modality e, but expressed in the style or representation space of modality d.

- We repeat the above process for all pairs of source and target modalities. The resulting matrix contains self-reconstructions on the diagonal and cross-modal reconstructions off the diagonal, demonstrating how one modality's latent representation can condition the generation of another modality's output.

- Finally, from each posterior distribution obtained in the cross-modal process, we extract the mean as the representative output. This provides both self- and cross-modality conditioned generations, ensuring that the generated samples reflect the underlying semantic content gleaned from the source modality.

A.2 Figures

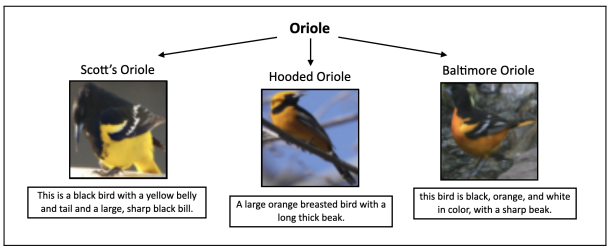


Figure 1: Example of CUBICC data subspecies (made up by a label - substituting sub-species labels, a caption and an image)

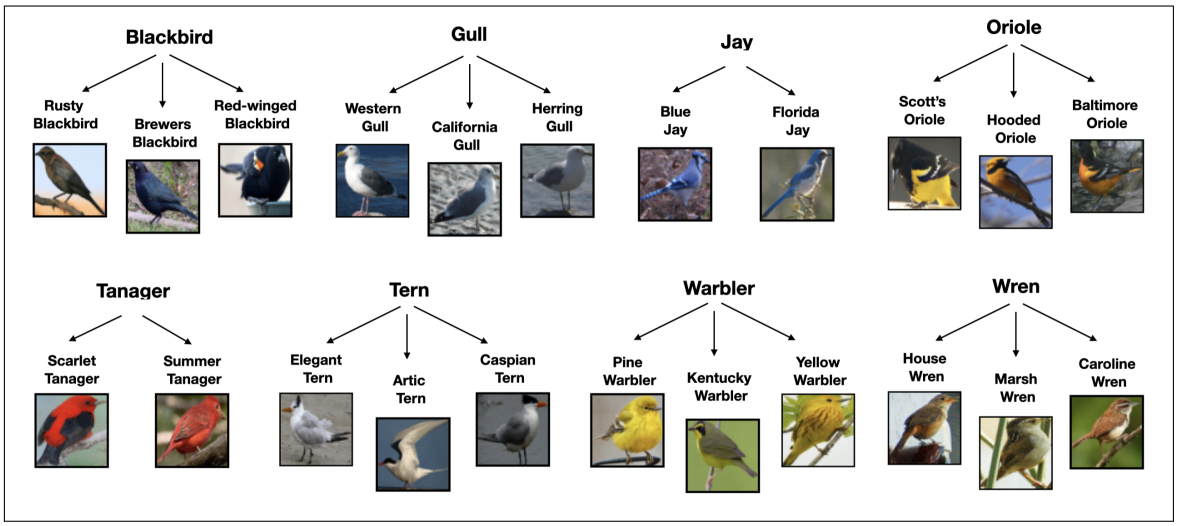


Figure 2: Example of CUBICC data bird spices and subspices.

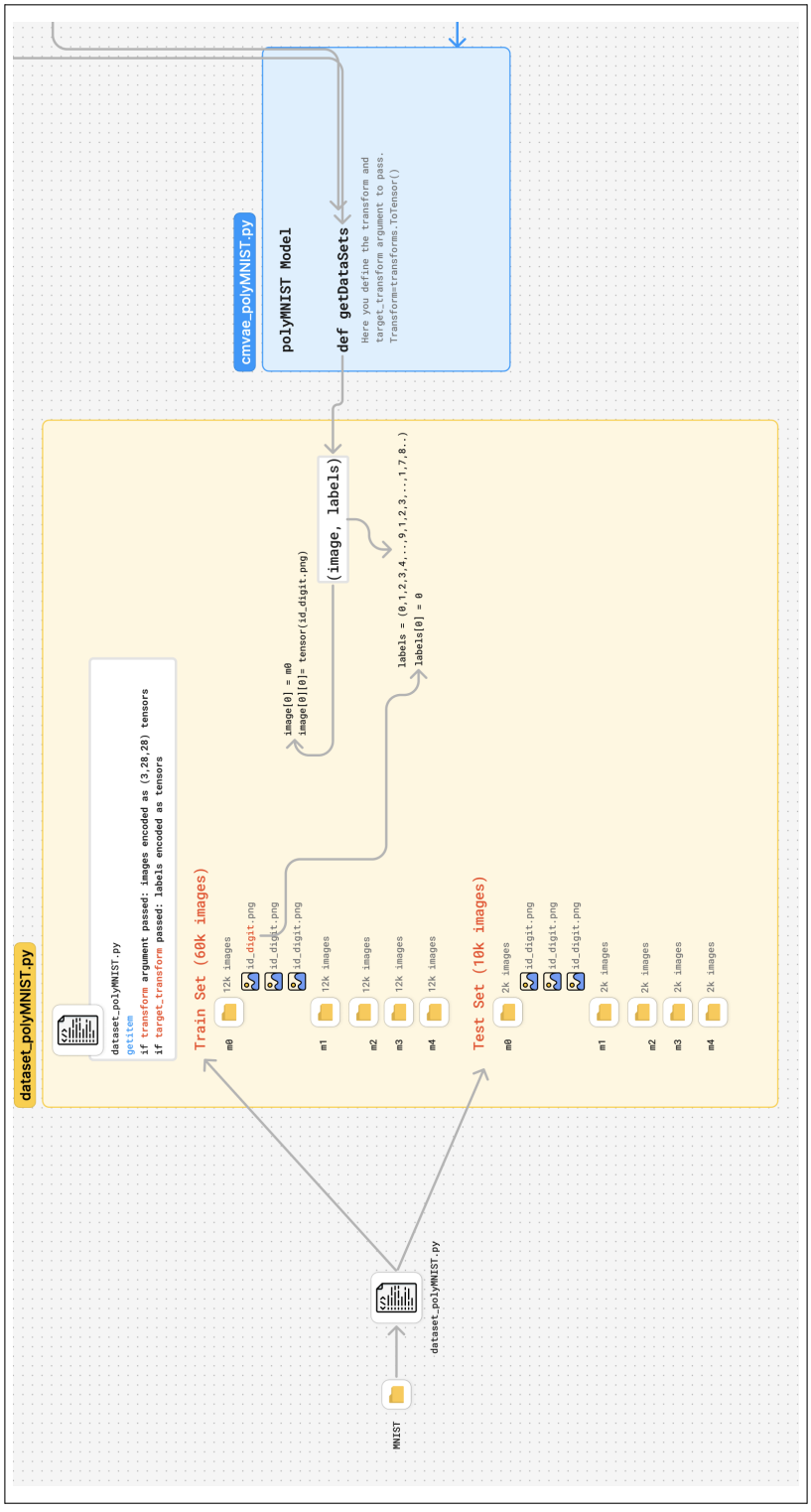


Figure 3: Conceptual scheme of PolyMnist dataset.

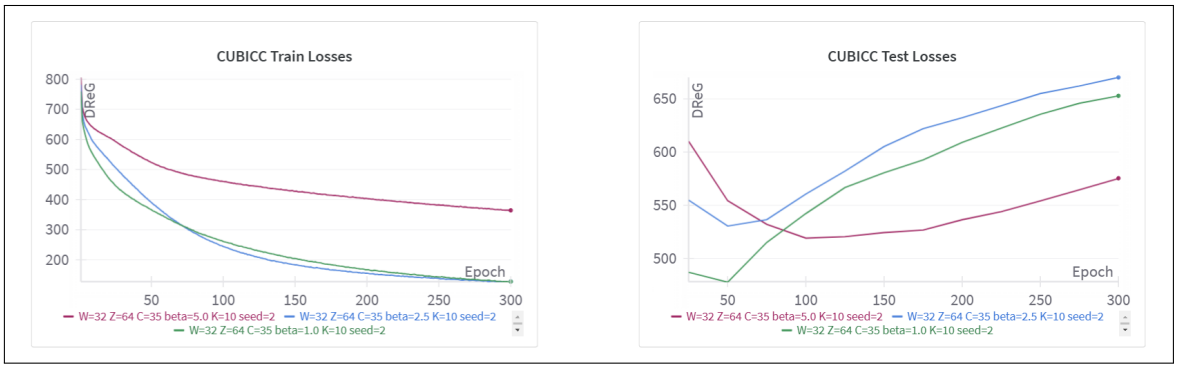


Figure 4: Losses for the models trained with the CUBICC dataset.

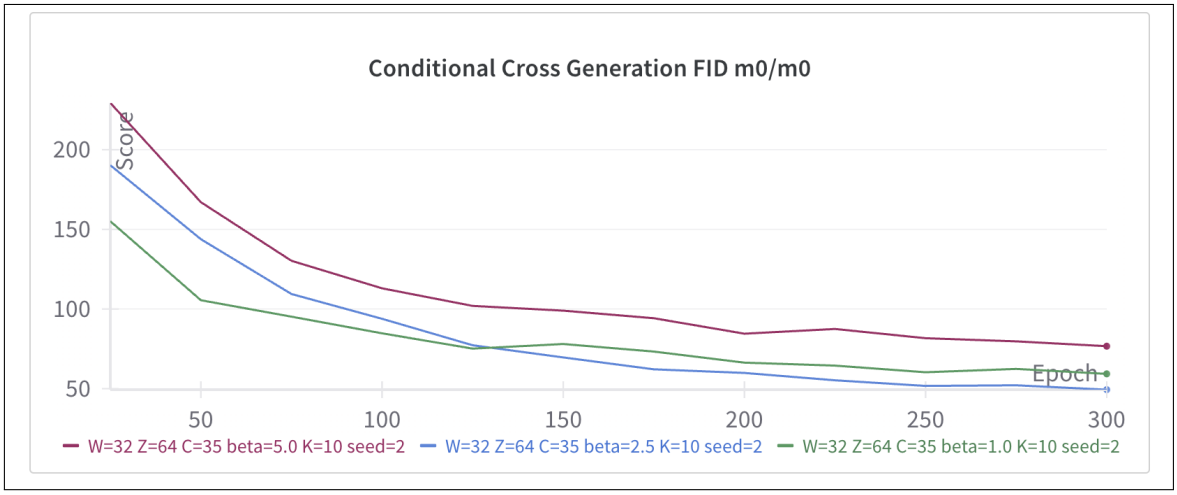


Figure 5: Conditional Cross generation m0/m0 for the models trained with the CUBICC dataset.

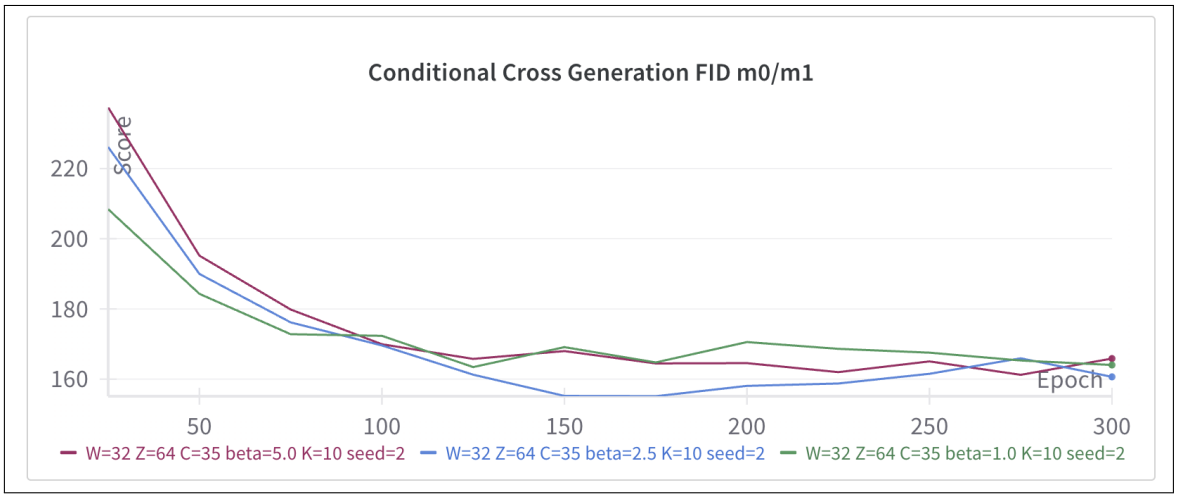


Figure 6: Conditional Cross generation m0/m1 for the models trained with the CUBICC dataset.

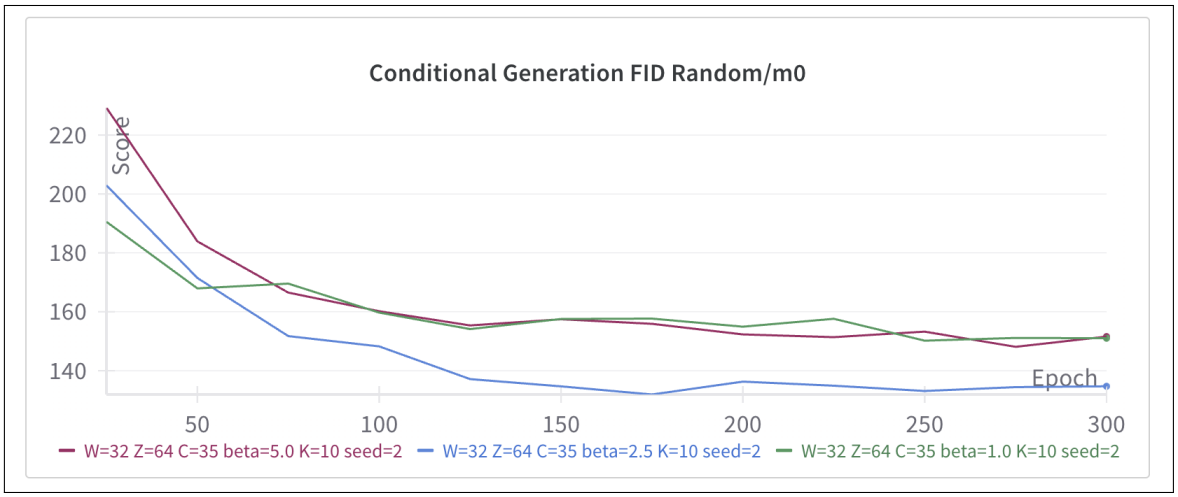


Figure 7: Conditional Cross generation random/m1 for the models trained with the CUBICC dataset.

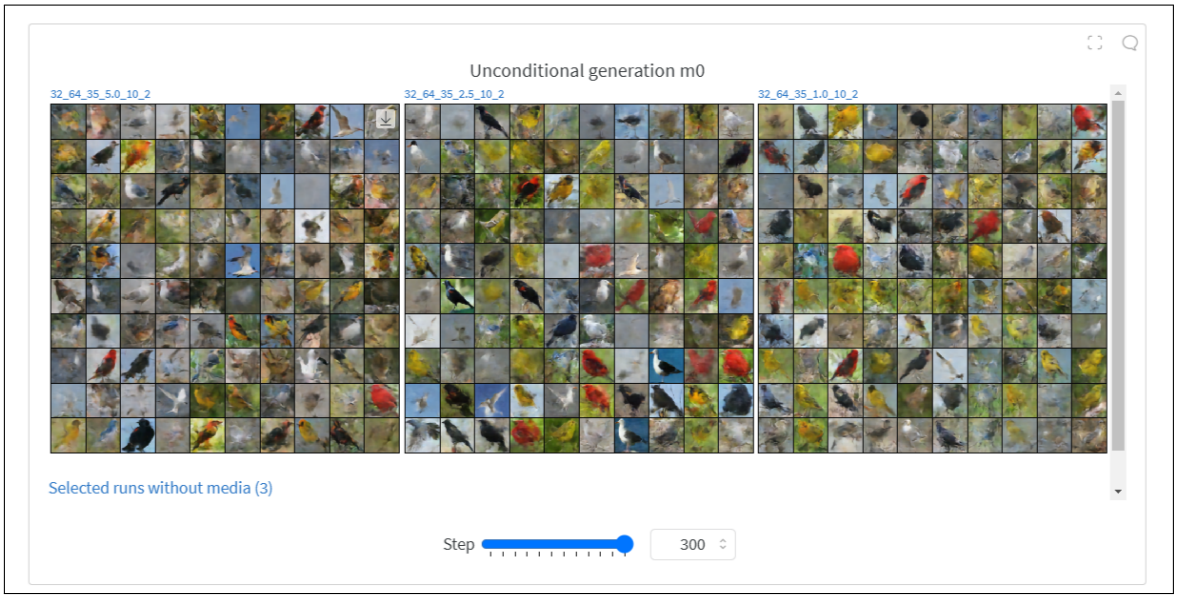


Figure 8: Qualitative overview of unconditional generation m0 for the models trained with the CUBICC dataset.

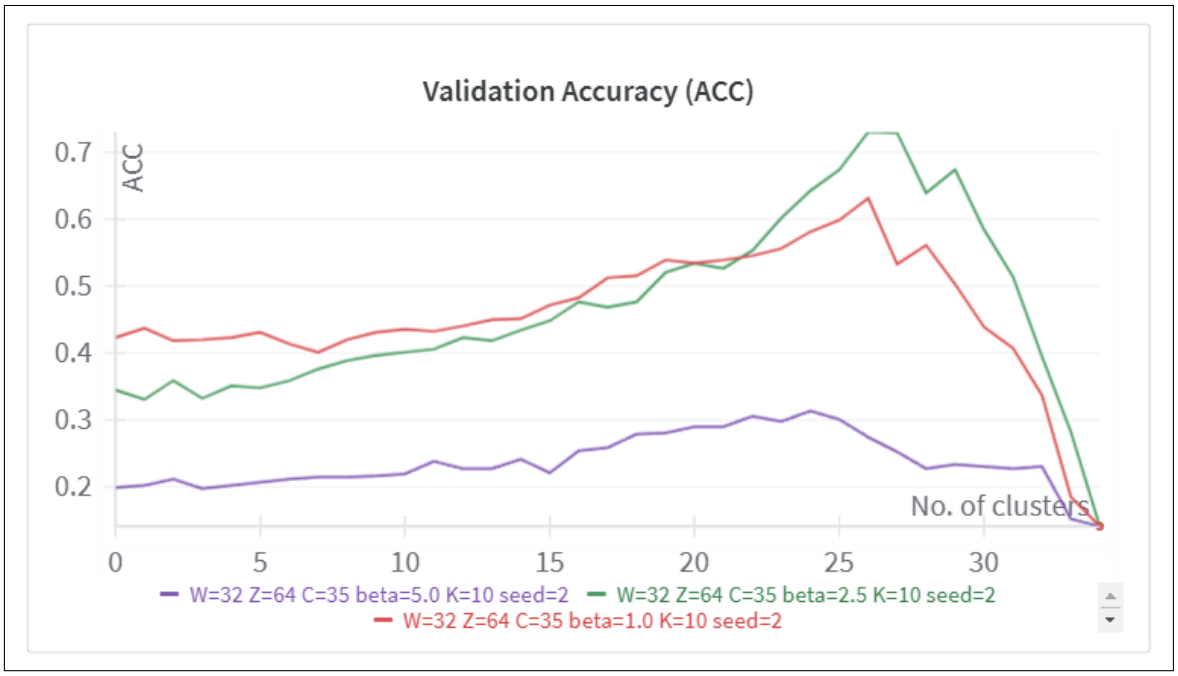


Figure 9: Accuracies for post-hoc pruning for models trained with CUBICC dataset

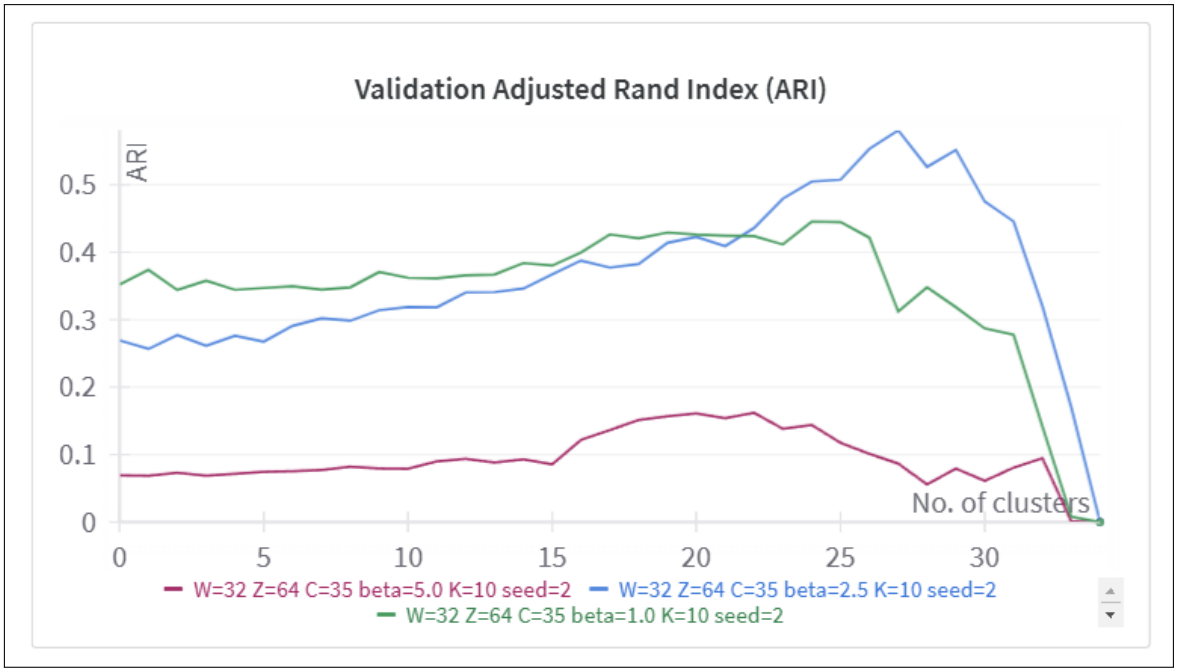


Figure 10: ARI for post-hoc pruning for models trained with CUBICC dataset

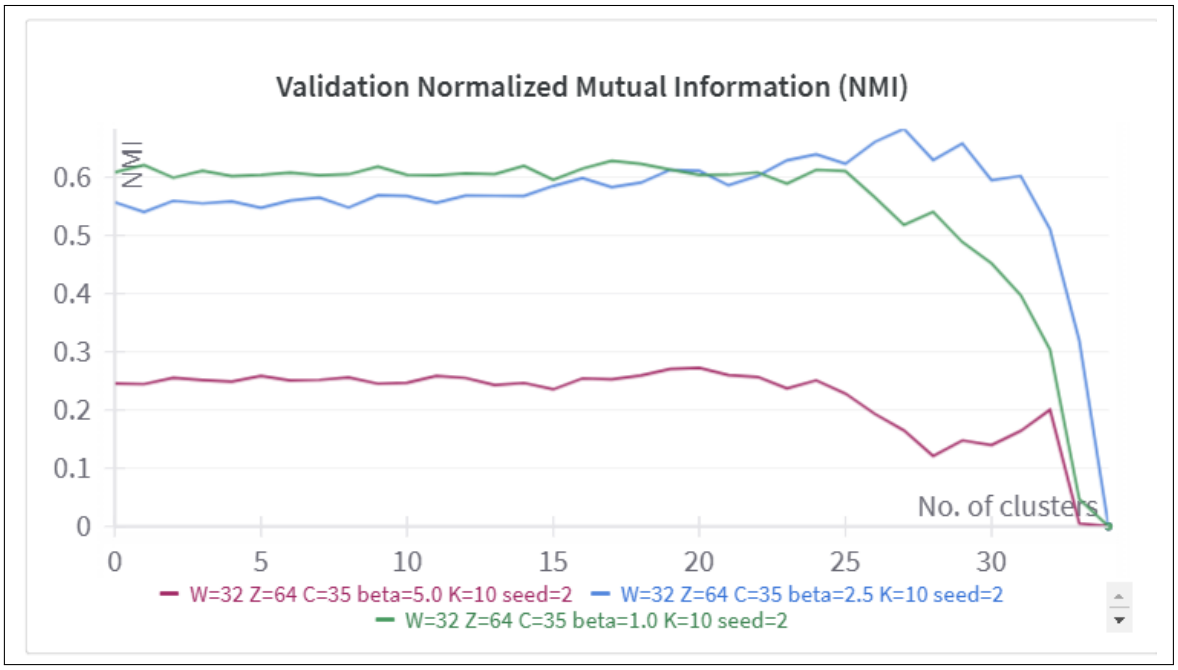


Figure 11: NMI for post-hoc pruning for models trained with CUBICC dataset

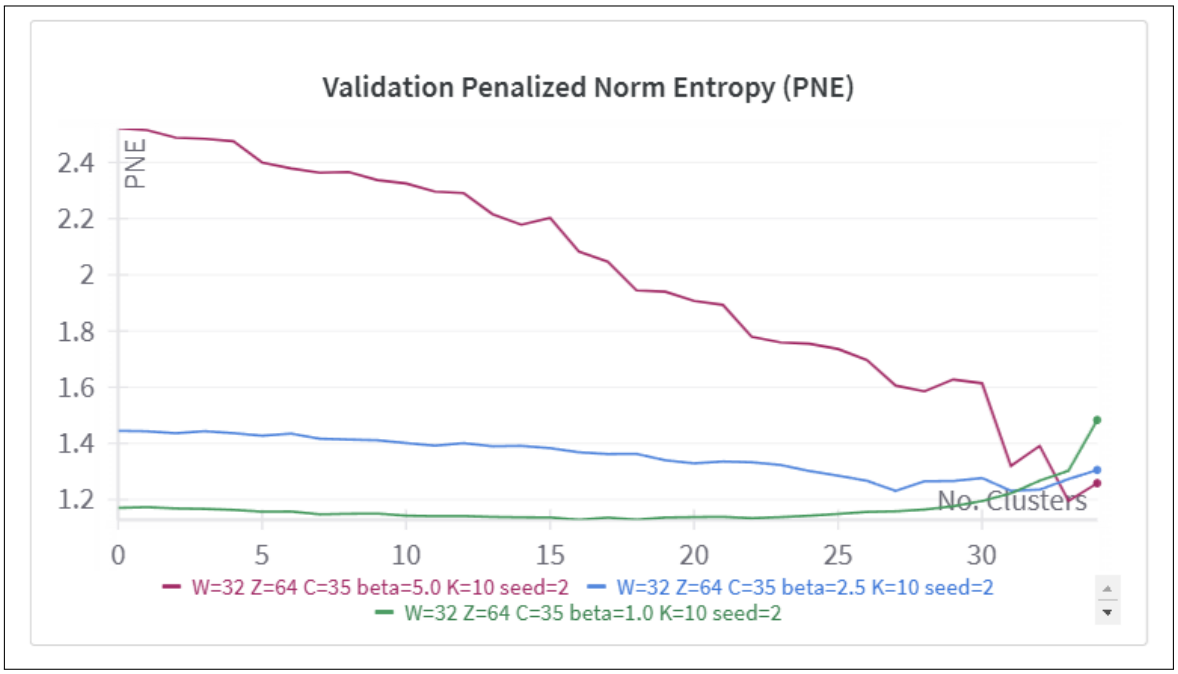


Figure 12: PNE for post-hoc pruning for models trained with CUBICC dataset

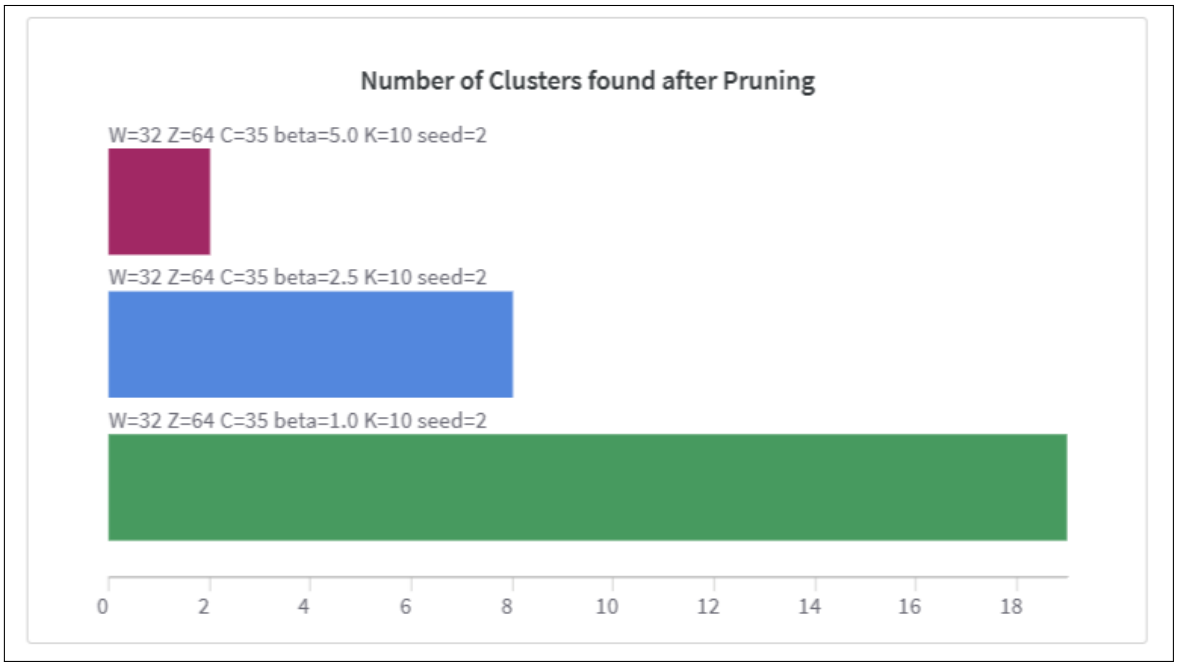


Figure 13: Number of clusters found after post-hoc pruning for models trained with CUBICC dataset

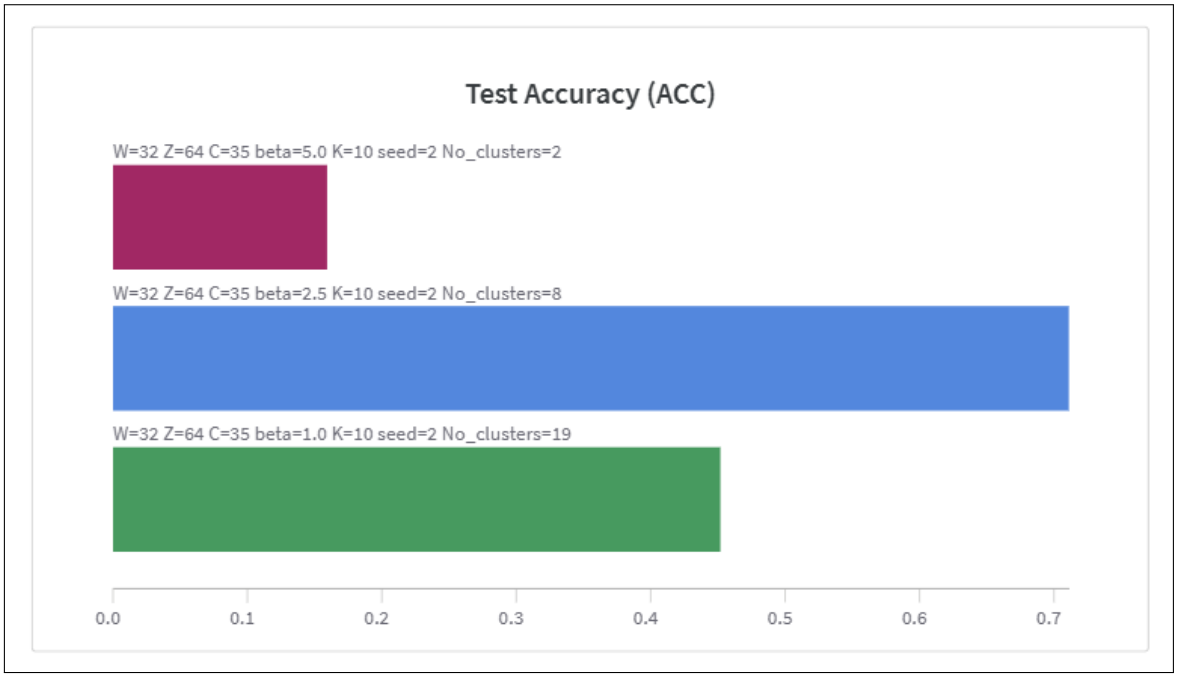


Figure 14: Test accuracies with optimal number of clusters for models trained with CUBICC dataset

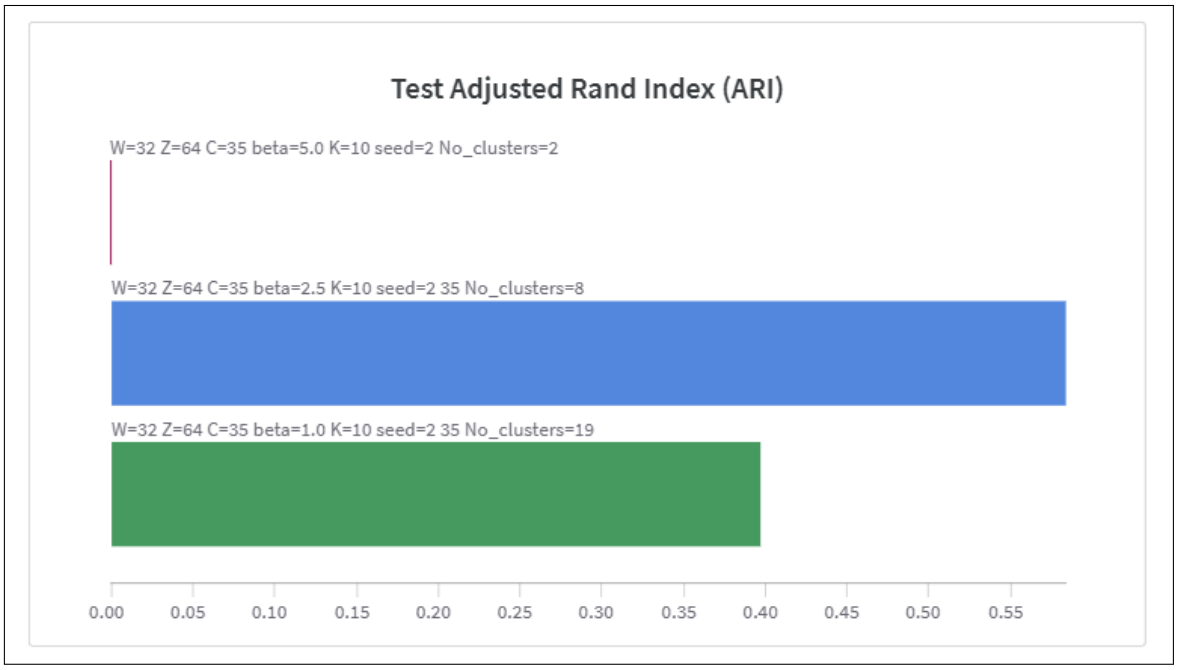


Figure 15: Test ARI with optimal number of clusters for models trained with CUBICC dataset

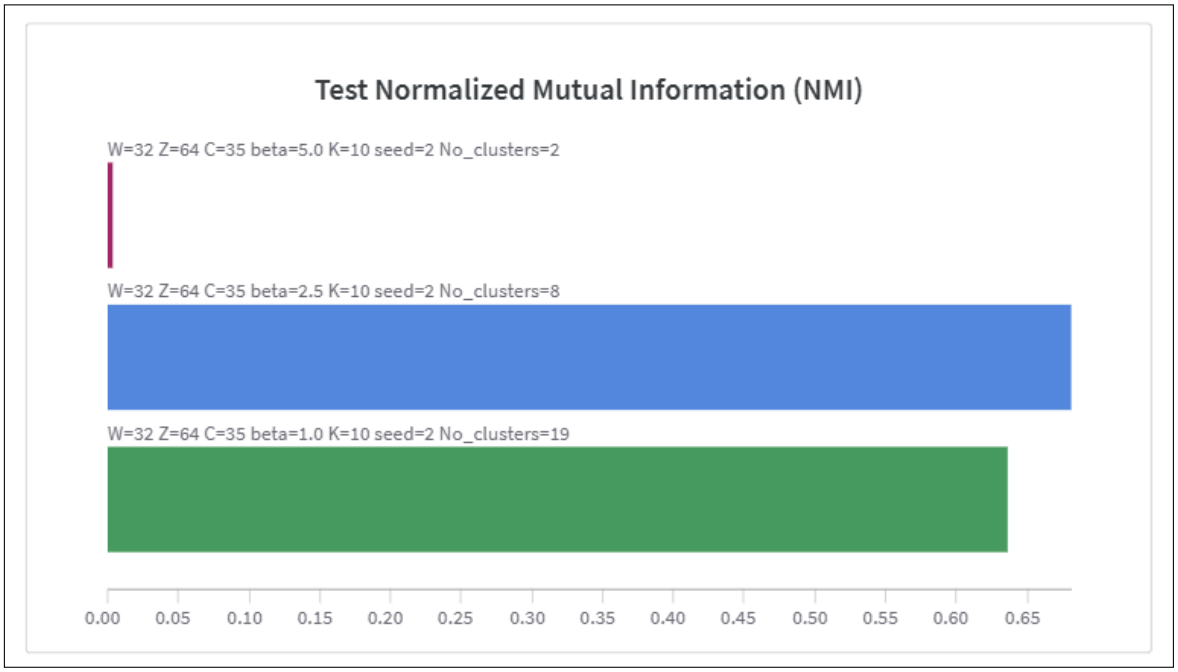


Figure 16: Test NMI with optimal number of clusters for models trained with CUBICC dataset

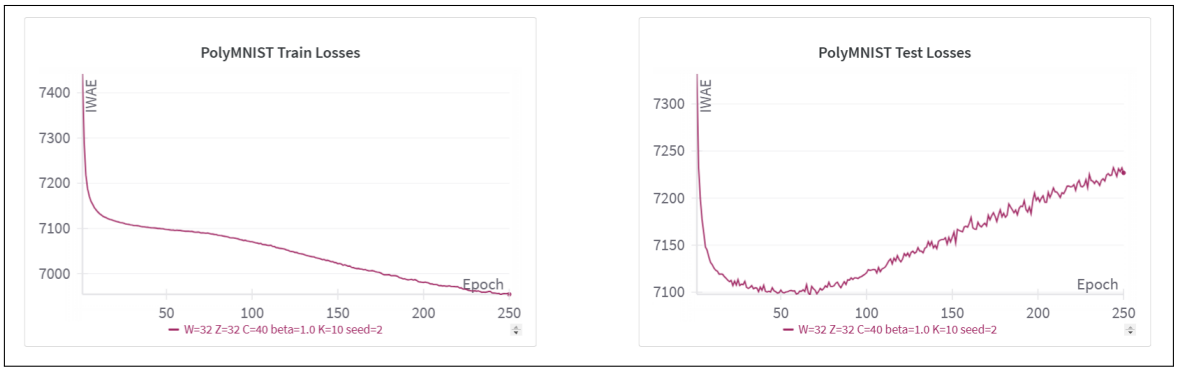


Figure 17: Losses for the models trained with the PolyMNIST dataset.

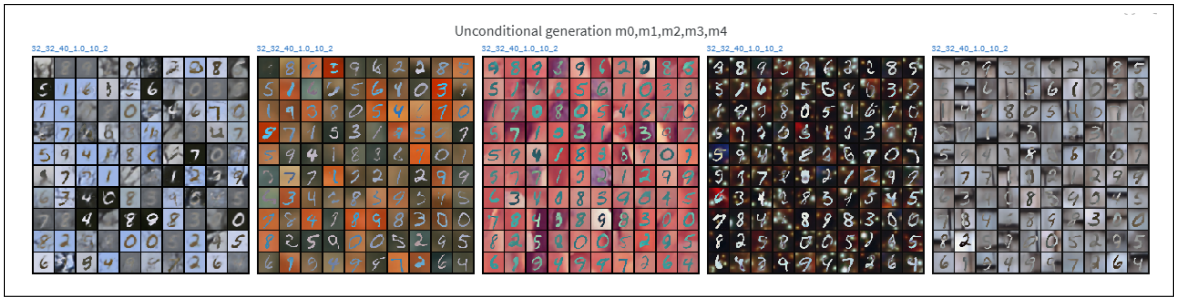


Figure 18: Qualitative overview of unconditional generation m0,m1,m2,m3,m4 for the model trained with the PolyMNIST dataset.

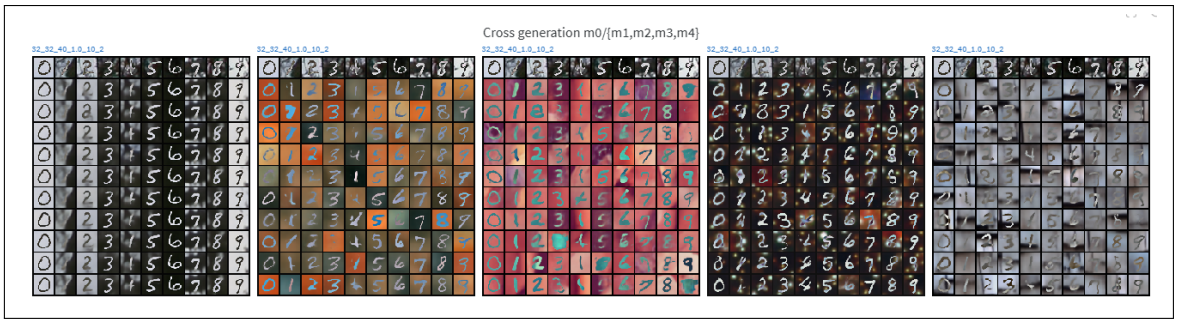


Figure 19: Qualitative overview of conditional generation m0/{m0,m1,m2,m3,m4} for the model trained with the PolyMNIST dataset.

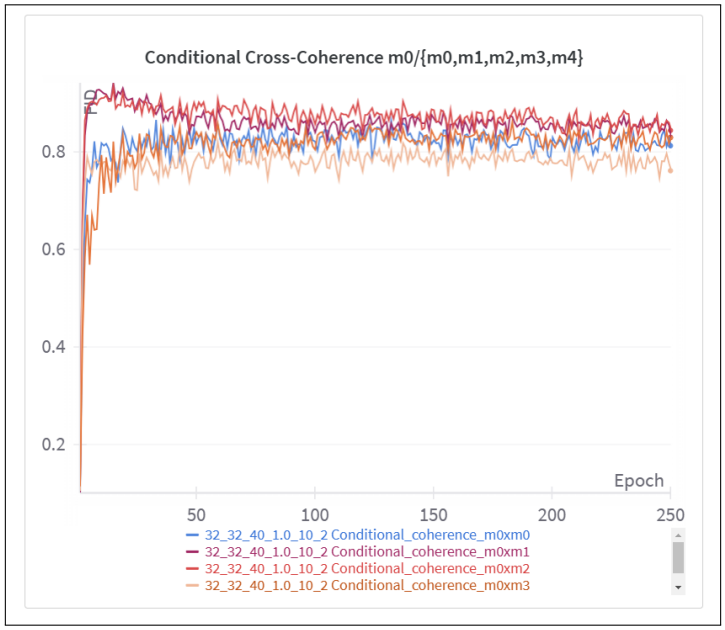


Figure 20: Cross coherence for m0 on conditional generation for the model trained with the PolyMNIST dataset.

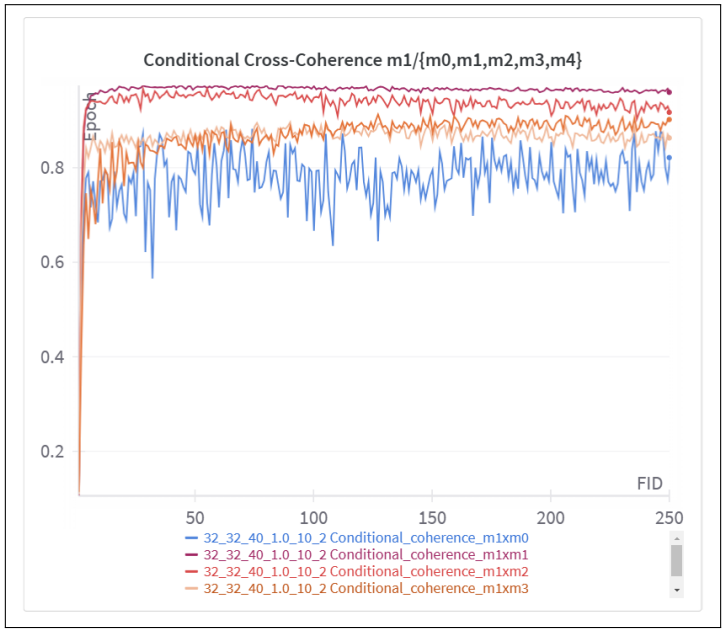


Figure 21: Cross coherence for m1 on conditional generation for the model trained with the PolyMNIST dataset.

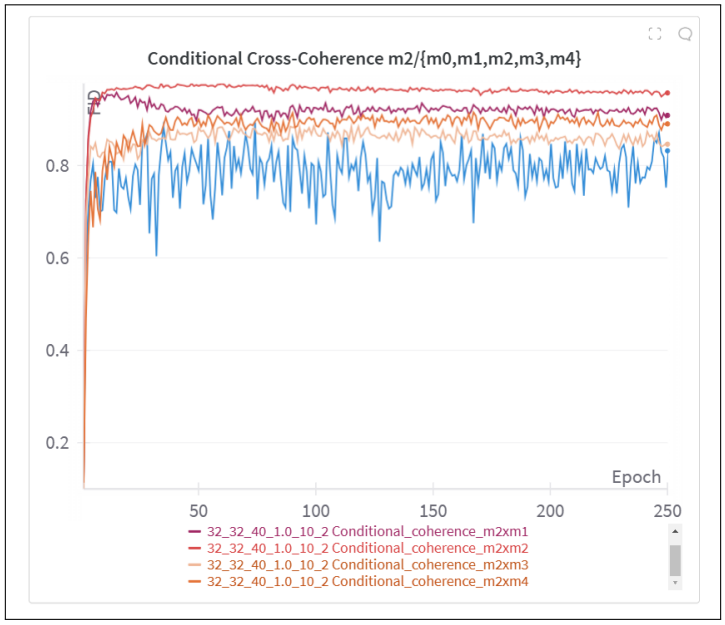


Figure 22: Cross coherence for m2 on conditional generation for the model trained with the PolyMNIST dataset.

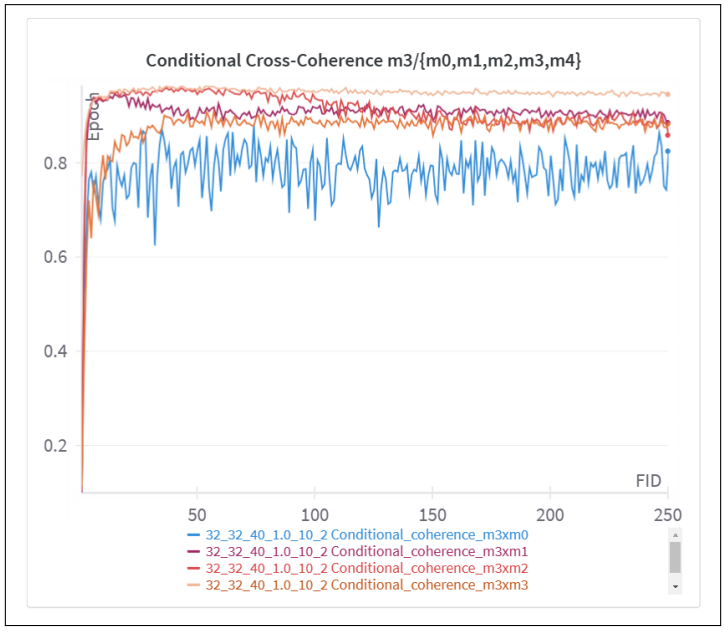


Figure 23: Cross coherence for m3 on conditional generation for the model trained with the PolyMNIST dataset.

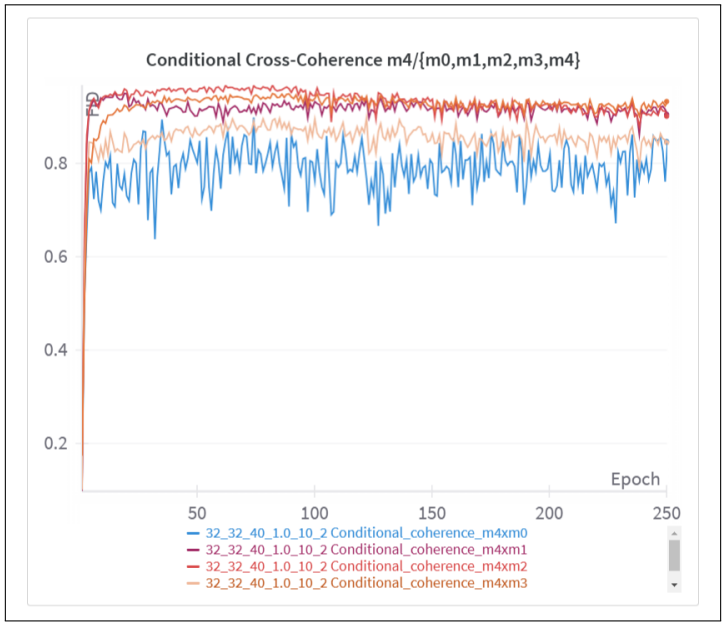


Figure 24: Cross coherence for m4 on conditional generation for the model trained with the PolyMNIST dataset.

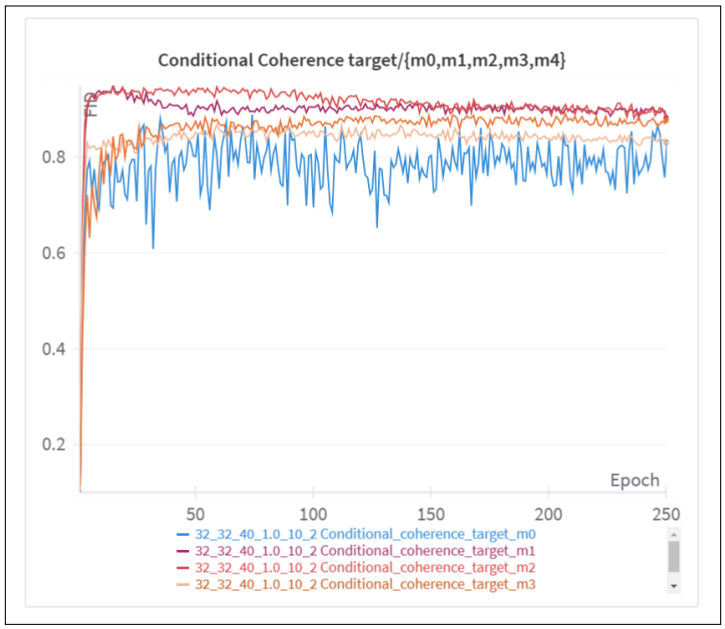


Figure 25: Cross coherence for a target on conditional generation for the model trained with the PolyMNIST dataset.

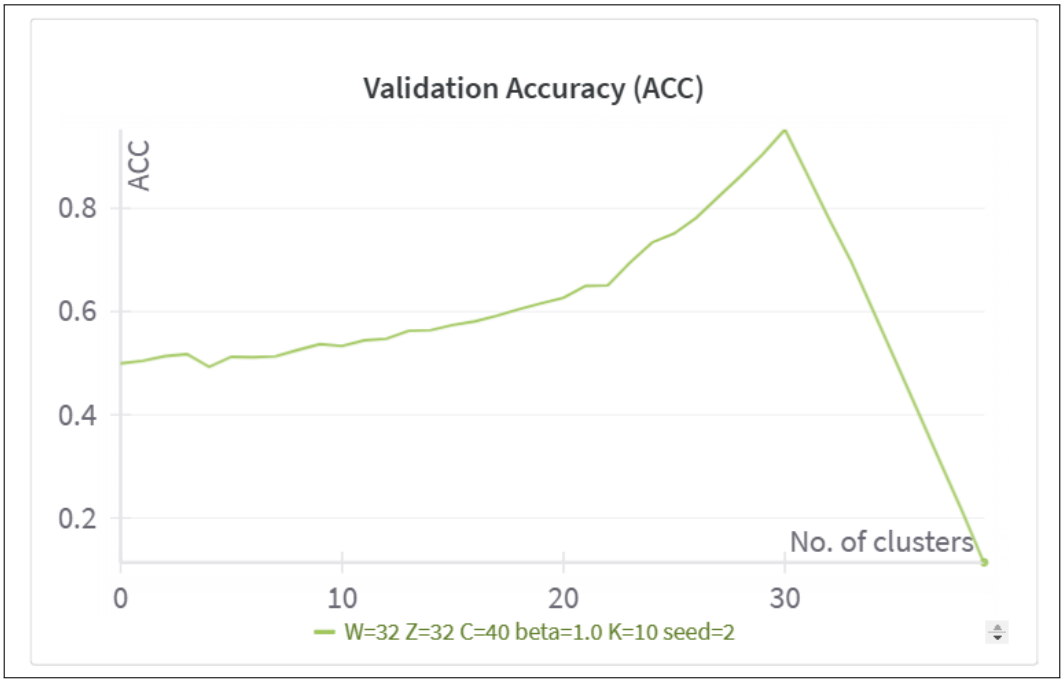


Figure 26: Accuracies for post-hoc pruning of the model trained with PolyMNIST dataset

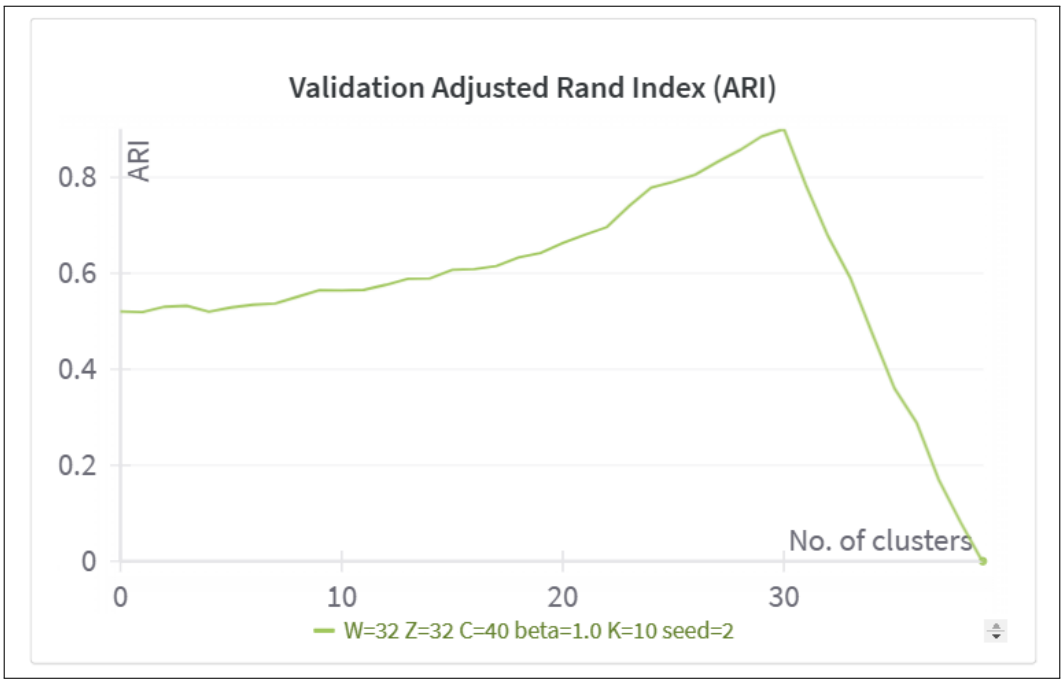


Figure 27: ARI for post-hoc pruning of the model trained with PolyMNIST dataset

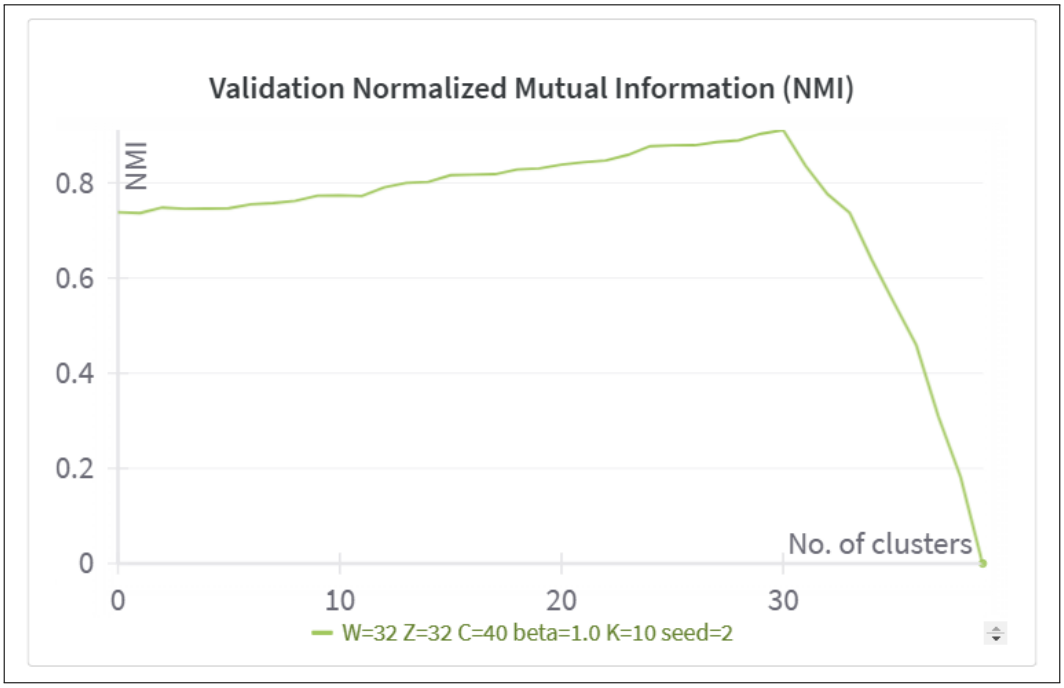


Figure 28: NMI for post-hoc pruning of the model trained with PolyMNIST dataset

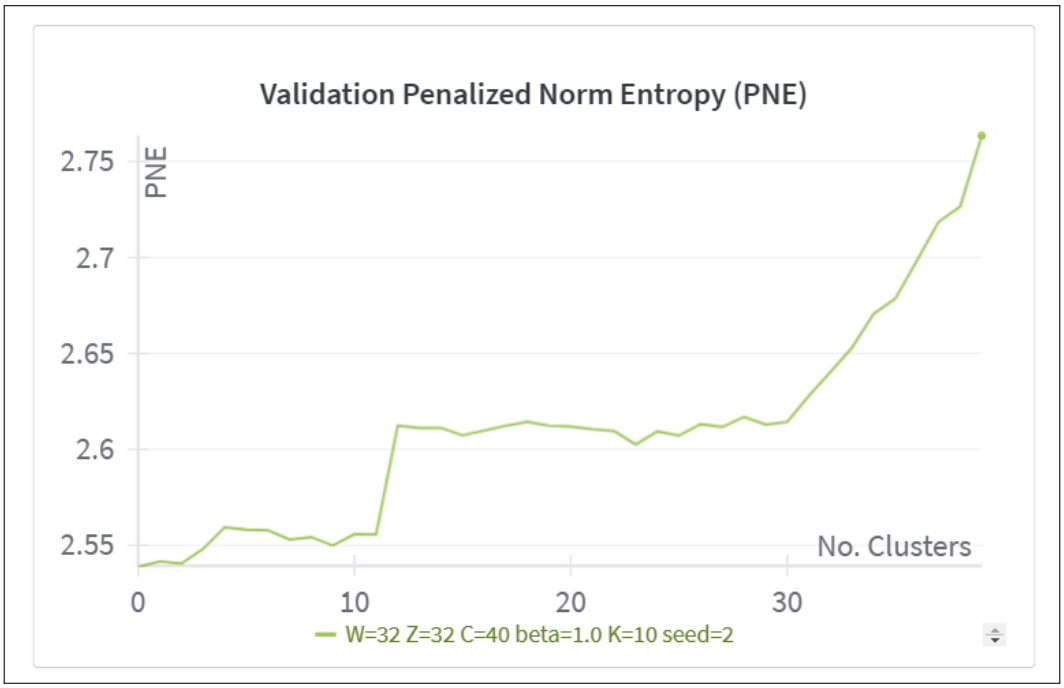
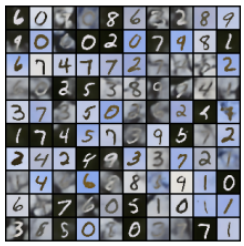


Figure 29: PNE for post-hoc pruning of the model trained with PolyMNIST dataset

Start date	Duration	Cost
28 Nov 2024	170:35:32	119.41

Figure 30: Usage of resources for the PolyMNIST dataset for one instance.



(a) modality 1

(b) modality 2

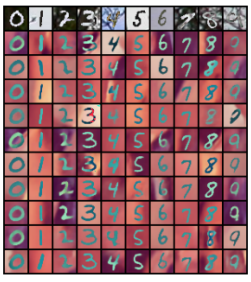
(c) modality 3

(d) modality 4

(e) modality 5

Figure 31: CMVAE research paper qualitative overview of unconditional generation for the model trained with the PolyMNIST dataset.

(a) modality 2



(b) modality 3

(c) modality 4

(d) modality 5

Figure 32: CMVAE research paper qualitative overview of conditional generation for the model trained with the PolyMNIST dataset.

Figure 33: CMVAE research paper qualitative overview of unconditional generation on m0 for the model trained with the CUBICC dataset.

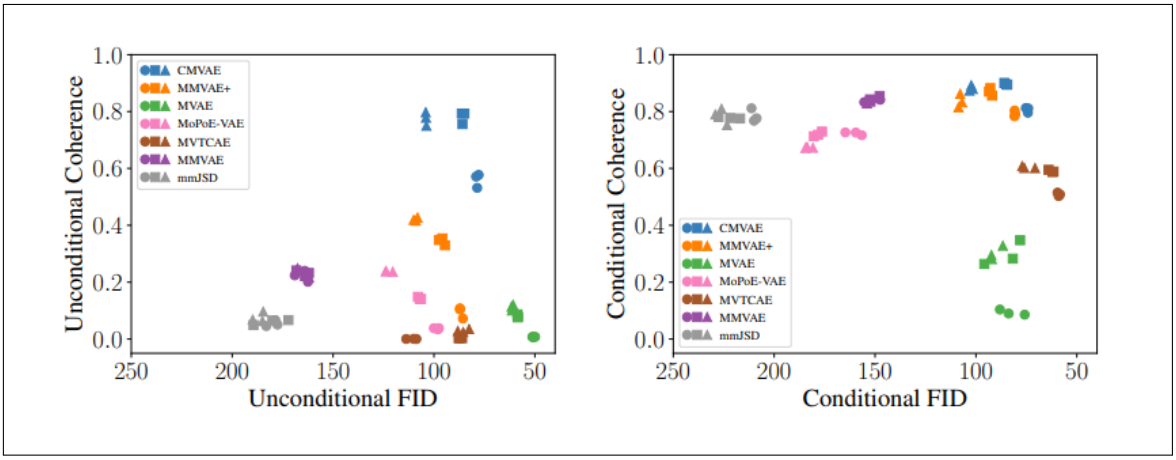


Figure 34: CMVAE research paper visual coherence in clustering for both datasets.

	PolyMNIST			CUBICC			
	NMI	ARI	ACC	NMI	ARI	ACC	$\bar{K}$ not given for training
VaDE	0.43 (0.04)	0.36 (0.04)	0.54 (0.05)	0.15 (0.01)	0.08 (0.01)	0.27 (0.01)	✗
DeepCluster	0.12 (0.02)	0.08 (0.02)	0.26 (0.04)	0.19 (0.01)	0.10 (0.01)	0.29 (0.01)	✗
CMC	0.97 (0.01)	0.97 (0.01)	0.99 (0.01)	0.37 (0.05)	0.10 (0.03)	0.31 (0.04)	✗
CMVAE	0.97 (0.02)	0.97 (0.02)	0.99 (0.01)	0.67 (0.07)	0.59 (0.09)	0.76 (0.07)	✓

Figure 35: CMVAE research paper pruning metrics overview of models in both datasets.

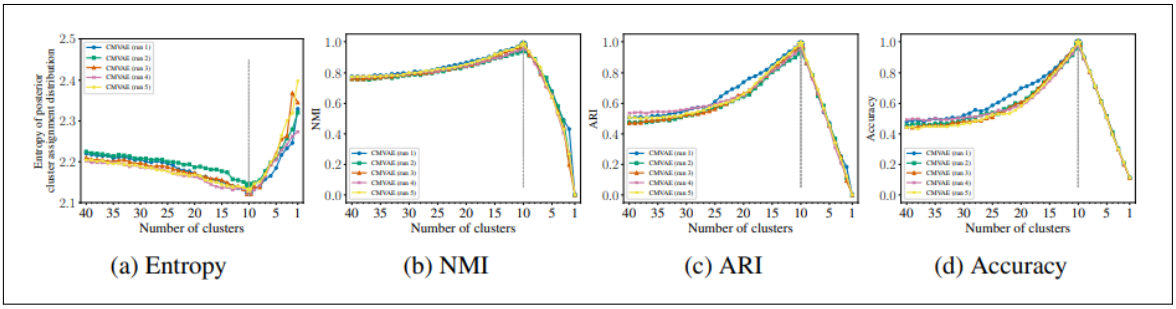


Figure 36: CMVAE research paper pruning metrics plots overview of models in both datasets.

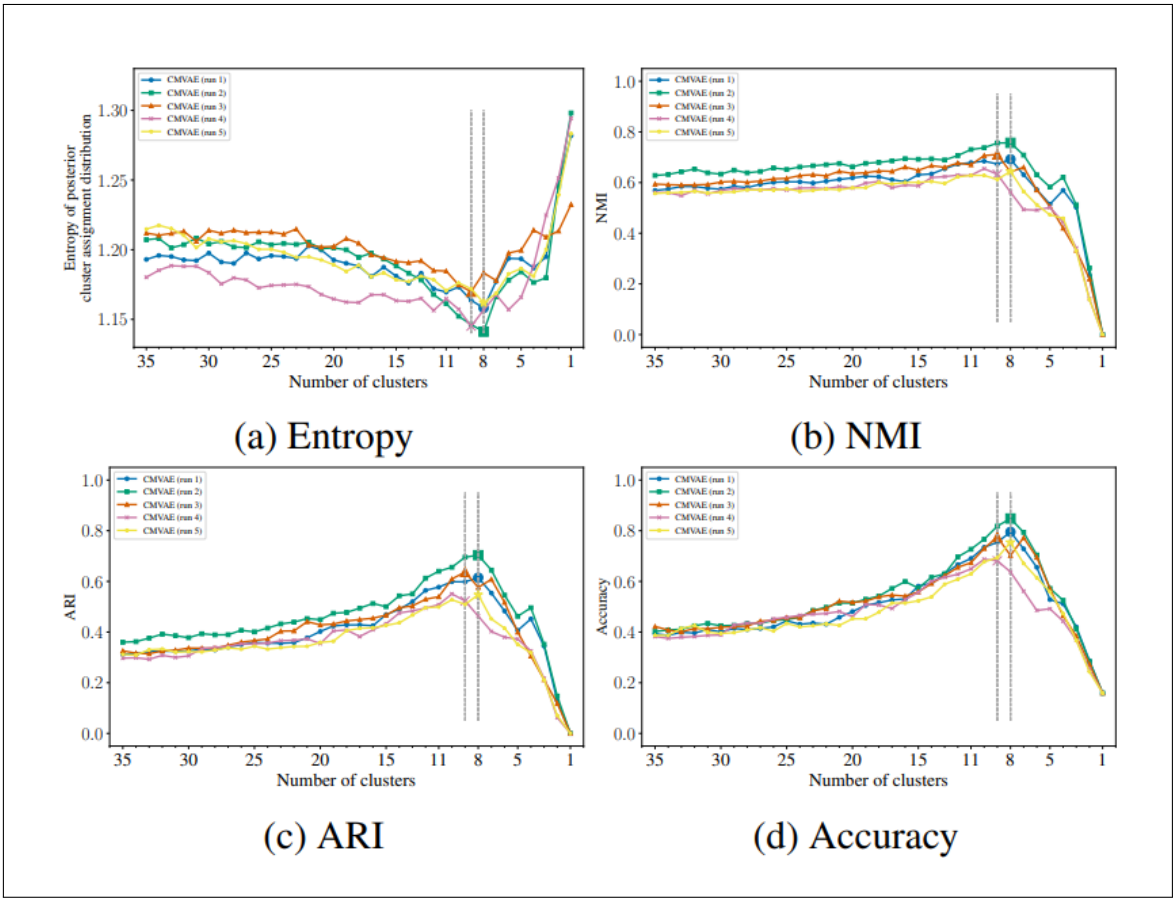


Figure 37: CMVAE research paper pruning metrics plots overview of models in CUBICC dataset.

$\beta = 1.0$	Unconditional		Conditional	
	FID	Coherence	FID	Coherence
MVAE	50.65 (0.72)	0.007 (0.001)	82.59 (6.22)	0.093 (0.009)
MVTCAE	110.85 (2.61)	0.000 (0.000)	58.98 (0.62)	0.509 (0.006)
mmJSD	179.76 (2.97)	0.054 (0.011)	209.98 (1.26)	0.785 (0.023)
MoPoE-VAE	98.56 (1.32)	0.037 (0.002)	160.29 (4.12)	0.723 (0.006)
MMVAE	165.17 (3.40)	0.222 (0.019)	152.11 (4.11)	0.837 (0.004)
MMVAE+	86.64 (1.04)	0.095 (0.020)	80.75 (0.18)	0.796 (0.010)
CMVAE	78.52 (0.63)	0.560 (0.025)	74.53 (0.64)	0.806 (0.009)

$\beta = 2.5$	Unconditional		Conditional	
	FID	Coherence	FID	Coherence
MVAE	58.53 (0.12)	0.080 (0.006)	85.23 (9.37)	0.298 (0.044)
MVTCAE	87.07 (0.89)	0.003 (0.000)	62.55 (1.30)	0.591 (0.004)
mmJSD	180.55 (8.67)	0.060 (0.010)	222.09 (5.34)	0.778 (0.003)
MoPoE-VAE	107.11 (0.780)	0.141 (0.005)	178.27 (2.01)	0.720 (0.008)
MMVAE	164.71 (3.17)	0.232 (0.010)	150.83 (2.69)	0.844 (0.010)
MMVAE+	96.01 (2.10)	0.344 (0.013)	92.81 (0.78)	0.869 (0.013)
CMVAE	85.68 (0.66)	0.781 (0.021)	85.12 (0.75)	0.897 (0.003)

$\beta = 5.0$	Unconditional		Conditional	
	FID	Coherence	FID	Coherence
MVAE	61.25 (0.40)	0.112 (0.010)	90.37 (3.20)	0.301 (0.024)
MVTCAE	85.43 (2.80)	0.029 (0.001)	74.61 (3.41)	0.604 (0.004)
mmJSD	186.49 (2.89)	0.076 (0.018)	226.20 (2.91)	0.784 (0.029)
MoPoE-VAE	122.68 (1.96)	0.238 (0.001)	182.99 (1.96)	0.673 (0.002)
MMVAE	164.29 (2.97)	0.229 (0.017)	152.11 (3.18)	0.839 (0.010)
MMVAE+	109.08 (1.41)	0.421 (0.006)	107.78 (0.88)	0.836 (0.023)
CMVAE	103.95 (0.16)	0.775 (0.024)	102.36 (0.83)	0.882 (0.010)

Figure 38: CMVAE research paper FID results for Conditional and Unconditional training with the CUBICC dataset.

A.3 Tables

Table 1: Overview of models trained in the paper.

Dataset	Betas	Dim z	Dim w	K
CUBICC	1.0, 2.5, 5.0	64	32	10, 40 (with beta 5.0)
PolyMNIST	1.0, 2.5, 5.0	32	32	10, 40 (with beta 5.0)

Table 2: Hyperparameters used to train the models.

Name	Def. Value	Description
`obj`	`'iwae'`	Objective to use. Choices: `'iwae'` (polyMNIST), `'dreg'` (CUBIC).
`K`	`1`	Number of samples when resampling in the latent space.
`llik_scaling`	`0.0`	Likelihood scaling.
`batch-size`	`128`	Batch size for training.
`epochs`	`250`	Number of training epochs.
`latent-dim-w`	`32`	Latent modality-specific dimensionality.
`latent-dim-z`	`32`	Latent shared dimensionality.
`latent-dim-c`	`40`	Number of latent clusters.
`learn-prior-c`	`True`	Learn prior probabilities for latent clusters.
`print-freq`	`50`	Frequency with which to print stats.
`no-cuda`	`False`	Disable CUDA use.
`seed`	`2`	Random seed.
`beta`	`2.5`	Beta hyperparameter in VAE objective.
`priorposterior`	`'Laplace'`	Distribution choice for prior and posterior. Choices: `'Normal'`, `'Laplace'`.
`lr`	`1e-3`	Learning rate (both for CUBIC and polyMNIST).

A.4 Links

- Project Github repository: https://github.com/Frenzoid/CMVAE
- Wandb CUBICC report: https://api.wandb.ai/links/frenzoid-atml/zu7625oi
- Wandb PolyMNIST report: https://api.wandb.ai/links/frenzoid-atml/gmy6uekn