## Limitations

Like other unsupervised disentanglement methods such as $\beta$-VAE [3] or $\beta$-TCVAE [1], some random initializations of the NashAE method resulted in better disentanglement scores than others, and training can fail in some very rare cases. This may be related to the findings of Locatello et al. [11], mentioned in the paper. Poor choices of hyperparameters such as a learning rate that is far too large can cause more frequent failures. No "outliers" were removed from the reported data in the paper, and the reported scores for all algorithms reflect honest averages.

Furthermore, each algorithm is trained with the same amount of data, and each autoencoder (AE) or variational AE (VAE) has roughly the same amount of parameters in each experiment (VAE has marginally more to implement the $\mu$, $\log(\sigma^2)$ parallel latent space). However, the NashAE method tends to collectively take more parameters and computations due to its use of the predictor ensemble. We observe that training for NashAE takes longer than VAE-based approaches due to the iterative predictor training. This process could be sped up by training the $m$ predictors in parallel.

Another limitation of NashAE is that it is not a generative model; it does not explicitly have a method to compute the likelihood of the data observations.

## MinMax Formulation

**Showing $\mathbf{z}' \to \mathbb{E}[\mathbf{z}]$.** If $\mathbf{z}$ and $\mathbf{z}'$ are independent, $\mathbf{z}' \to \mathbb{E}[\mathbf{z}]$. Consider $\mathbf{z}' = \mathbb{E}[\mathbf{z}] + \delta_{\mathbf{z}'}$, where $\mathbb{E}[\delta_{\mathbf{z}'}] = 0$. Plugging this into $\text{MSE}(\mathbf{z}, \mathbf{z}')$, we get: $\frac{1}{2}\mathbb{E}[(\mathbf{z} - \mathbb{E}[\mathbf{z}] - \delta_{\mathbf{z}'})^2] = \frac{1}{2}\mathbb{E}[\mathbf{z}^2 + \mathbb{E}[\mathbf{z}]^2 - 2\mathbf{z}\mathbb{E}[\mathbf{z}] - 2\mathbf{z}\delta_{\mathbf{z}'} + 2\mathbb{E}[\mathbf{z}]\delta_{\mathbf{z}'} + \delta_{\mathbf{z}'}^2] = \frac{1}{2}(\mathbb{E}[\mathbf{z}^2] - \mathbb{E}[\mathbf{z}]^2 + \mathbb{E}[\delta_{\mathbf{z}'}^2]) = \frac{1}{2}(\sigma_{\mathbf{z}}^2 + \sigma_{\mathbf{z}'}^2)$. Since $\sigma_{\mathbf{z}'}^2 \geq 0$, this implies that SGD on $\mathbf{z}'$ reduces $\sigma_{\mathbf{z}'} \to 0$ to minimize the MSE objective. Hence, under these conditions, $\mathbf{z}' \to \mathbb{E}[\mathbf{z}]$.

**Partial Derivative of $\text{Cov}(A, B)$.** Consider taking the partial derivative of the covariance between $K$ samples of two jointly distributed random variables $A$ and $B$ with respect to an observation $b_q$, where $1 \leq q \leq K$. Hence, $\frac{\partial}{\partial b_q}\text{Cov}(A, B) = \frac{\partial}{\partial b_q}\frac{1}{K}\sum_{i=1}^{K}(a_i - \mathbb{E}[A])(b_i - \mathbb{E}[B])$. If $\mathbb{E}[A]$ and $\mathbb{E}[B]$ are computed as empirical averages across the $K$ samples, we have $\frac{\partial}{\partial b_q}\frac{1}{K}\sum_{i=1}^{K}(a_i - \mathbb{E}[A])(b_i - \mathbb{E}[B]) = \frac{1}{K}[(a_q - \mathbb{E}[A]) - \frac{1}{K}\sum_{i=1}^{K}(a_i - \mathbb{E}[A])] = \frac{1}{K}[(a_q - \mathbb{E}[A]) - \mathbb{E}[A] + \mathbb{E}[A]] = \frac{1}{K}(a_q - \mathbb{E}[A])$. Without loss of generality, this result can be applied to the statement of the next section.

**Equivalence between MSE and Covariance Objectives.** The fixed point of minimizing $\lambda\sum_{i=1}^{m}\text{Cov}(\mathbf{z}_i', \mathbf{z}_i)$ is equivalent to minimizing $\mathbb{E} - \frac{\lambda}{2}||\mathbf{z}' - \mathbf{z}||_2^2$ (i.e., maximize the MSE between $\mathbf{z}$ and $\mathbf{z}'$). This is because when a representation is disentangled, knowledge of other latent variables lends no useful information to the predictor. To minimize the MSE on its regression task, the uninformed $i$-th predictor will output $\mathbb{E}[\mathbf{z}_i]$ everywhere, making the gradient of $-\frac{\lambda}{2K}||\mathbf{z}' - \mathbf{z}||_2^2$ equal to $\frac{\lambda}{K}(\mathbf{z} - \mathbb{E}[\mathbf{z}])$, which *is equivalent to* the gradient term of minimizing $\lambda\sum_{i=1}^{m}\text{Cov}(\mathbf{z}_i', \mathbf{z}_i)$ with respect to the elements of an observation of $\mathbf{z}'$. In other words, $\frac{\partial}{\partial \mathbf{z}'}\lambda\sum_{i=1}^{m}\text{Cov}(\mathbf{z}_i', \mathbf{z}_i) = \frac{\partial}{\partial \mathbf{z}'}\mathbb{E} - \frac{\lambda}{2}||\mathbf{z}' - \mathbf{z}||_2^2$ if the elements of $\mathbf{z}$

are independent. Note that the AE uses the above gradient term along with $\frac{\partial \mathbf{z}'}{\partial \mathbf{z}}$ (evaluated at each sample in the batch) to adjust its representations $\mathbf{z}$ during the adversarial game (gradients are taken through the predictors only).

### Experiment Setup

All experiments were conducted using the PyTorch environment [12]. Each algorithm is trained with the same amount of data - enough to ensure that both are converged for the given task. The learning rate is 0.001 for the AE and the VAEs, and 0.01 for the predictors in every experiment. We choose $k = 5$ for the number of predictor update iterations per AE update. It is likely that $k$ can be reduced from 5, but we did not explore other values of $k$ in this work. The details for each task are reported in table 1. SELU means self-normalizing activation function, from Klambauer et al. [7]. The signature for Conv is (*width* x *height*)Conv(*in_channels*, *out_channels*, *pad*, *stride*). Recall that $m$ denotes the latent space size. The predictor architecture for every experiment is the following: Linear($m$, 40), SELU, Linear(40, 40), SELU, Linear(40, 1). The ReLU networks (for dSprites) are initialized via a Kaiming normal distribution [2]. We employ the Adam optimizer [5] with $\beta_1 = 0.9,\ \beta_2 = 0.999$ for all AEs, VAEs, and predictors.

Table 1: Network architecture and hyperparameters for each experiment

| Dataset | NashAE Architecture | VAE Architecture |
|---|---|---|
| **Beamsynthesis** | Linear(1000, 200), SELU | Linear(1000, 200), SELU |
| Mean/STD Data Norm. | Linear(200, 80), SELU | Linear(200, 80), SELU |
| | Linear(80, 40), SELU | Linear(80, 40), SELU |
| $BS \leftarrow 100$ | Linear(40, $m$), Sigmoid | 2× Linear(40, $m$), (see VAE [6]) |
| | Linear($m$, 40), SELU | Linear($m$, 40), SELU |
| | Linear(40, 80), SELU | Linear(40, 80), SELU |
| | Linear(80, 200), SELU | Linear(80, 200), SELU |
| | Linear(200, 1000) | Linear(200, 1000), Gaussian |
| **dSprites** [3] | Linear(4096, 1200), ReLU | Linear(4096, 1200), ReLU |
| Flatten Tensor | Linear(1200, 1200), ReLU | Linear(1200, 1200), ReLU |
| | Linear(1200, $m = 10$), Sigmoid | 2× Linear(1200, $m = 10$), (see VAE [6]) |
| $BS \leftarrow 200$ | Linear($m = 10$, 1200), ReLU | Linear($m = 10$, 1200), Tanh (see [3]) |
| | Linear(1200, 1200), ReLU | Linear(1200, 1200), Tanh |
| | Linear(1200, 4096) | Linear(1200, 1200), Tanh |
| | - | Linear(1200, 4096), Bernoulli |
| **CelebA** [10] | $4 \times 4$ Conv(3, 32, 1, 2), SELU | $4 \times 4$ Conv(3, 32, 1, 2), SELU |
| Resize width to 96 | $4 \times 4$ Conv(32, 32, 1, 2), SELU | $4 \times 4$ Conv(32, 32, 1, 2), SELU |
| Center Crop $64 \times 64$ | $4 \times 4$ Conv(32, 64, 1, 2), SELU | $4 \times 4$ Conv(32, 64, 1, 2), SELU |
| Mean/STD Data Norm. | $4 \times 4$ Conv(64, 64, 1, 2), SELU | $4 \times 4$ Conv(64, 64, 1, 2), SELU |
| | Linear(1024, 256), SELU | Linear(1024, 256), SELU |
| $BS \leftarrow 200$ | Linear(256, $m = 32$), Sigmoid | 2× Linear(256, $m = 32$), (see VAE [6]) |
| | Linear($m = 32$, 256), SELU | Linear($m = 32$, 256), SELU |
| | Linear(256, 1024), SELU | Linear(256, 1024), SELU |
| | $4 \times 4$ ConvT(64, 64, 1, 2), SELU | $4 \times 4$ ConvT(64, 64, 1, 2), SELU |
| | $4 \times 4$ ConvT(64, 32, 1, 2), SELU | $4 \times 4$ ConvT(64, 32, 1, 2), SELU |
| | $4 \times 4$ ConvT(32, 32, 1, 2), SELU | $4 \times 4$ ConvT(32, 32, 1, 2), SELU |
| | $4 \times 4$ ConvT(32, 3, 1, 2) | $4 \times 4$ ConvT(32, 3, 1, 2), Gaussian |

For FactorVAE training, the learning rate for the VAE and discriminator are both set to $10^{-4}$ for all experiments, following [4]. The architecture of the dis-

criminator in all experiments is Linear($m$, 1024), SELU, Linear(1024, 1024), SELU, Linear(1024, 1024), SELU, Linear(1024, 1), sigmoid. We employ the Adam optimizer [5] with $\beta_1 = 0.5,\ \beta_2 = 0.9$, following [4].

### NashAE Algorithm

See algorithm 1 for a detailed description of the NashAE training algorithm.

**Algorithm 1** Training Algorithm for NashAE

**for** data minibatch $b$ in dataloader **do**
  $z \leftarrow \phi(b)$ ▷ $z$ is a minibatch of latent representations
  **for** predictor $\rho_i$ in $\rho$ **do**
    $\overline{\mathbf{z}_i} \leftarrow \mathbf{z} \cdot (\underline{\mathbf{1}} - \mathbb{I}_i); \forall \mathbf{z} \in z$ ▷ $\mathbb{I}_i$ is the $i$-th column of the identity matrix
    **for** number of predictor training iterations $k$ **do**
      $\mathcal{L}_{\rho_i} \leftarrow \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim z}\big(\rho(\overline{\mathbf{z}_i})_i - \mathbf{z}_i\big)^2$ ▷ reconstruction loss for the predictors
      $\rho_i \leftarrow \rho_i - \nu_\rho \nabla_{\rho_i} \mathcal{L}_{\rho_i}$ ▷ SGD update for the $i$-th predictor's MSE objective using learning rate $\nu_\rho$
    **end for**
    $z'_i \leftarrow \rho(\overline{\mathbf{z}_i})_i$ ▷ prediction for the $i$-th latent variable, concatenates into $z'$
  **end for**
  $b' \leftarrow \psi(z)$ ▷ $b'$ is the reconstructed data minibatch
  $\mathcal{L}_R \leftarrow \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim b}\big|\big|\mathbf{x}' - \mathbf{x}\big|\big|_2^2$ ▷ reconstruction loss for the autoencoder
  $\mathcal{L}_A \leftarrow \sum_{i=1}^{m} \mathrm{Cov}(z'_i, z_i)$ ▷ covariance of latent and prediction (across batch)
  $\mathcal{L}_{R,A} \leftarrow (1-\lambda)\mathcal{L}_R + \lambda\mathcal{L}_A$ ▷ combined loss for the AE
  $(\phi, \psi) \leftarrow (\phi, \psi) - \nu_{\phi,\psi} \nabla_{\phi,\psi} \mathcal{L}_{R,A}$ ▷ SGD update for the AE, adversarial gradients are taken through $\rho$, learning rate $\nu_{\phi,\psi}$
**end for**

### dSprites Dimensionality Experiment

We present a dimensionality learning experiment similar to the one conducted with Beamsynthesis, but with the dSprites dataset in Table 2. We include the VAE algorithms that were most competitive on Beamsynthesis. NashAE is still the most reliable in recovering the true number of data generating factors. Following the precedent of past works, we do not normalize the dSprites data, leading to ideal hyperparameter (e.g., $\lambda$ & $\beta$) settings that differ from Beamsynthesis and CelebA.

### Beamsynthesis Dataset Description

This dataset is generated by a synthetic beamline model, which models the generation, acceleration and tuning of ion beams in the LINAC (linear particle accelerator) portion of high-energy particle accelerators. A more comprehensive

Table 2: Average absolute difference between the number of factors learned by each algorithm and the true number of data generating factors (5) on the dSprites dataset

| Algorithm | m=10 | m=20 |
|---|---|---|
| NashAE $\lambda = 0.0$ | 5 | 15 |
| NashAE $\lambda = 0.004$ | 0.375 | 1 |
| NashAE $\lambda = 0.006$ | **0.25** | **0.375** |
| NashAE $\lambda = 0.008$ | 0.625 | 0.875 |
| FactorVAE $\beta = 4$ | 0.875 | 1.875 |
| FactorVAE $\beta = 8$ | 0.5 | 0.875 |
| $\beta$-TCVAE $\beta = 1$ | 4.875 | 9.25 |
| $\beta$-TCVAE $\beta = 4$ | 0.5 | 1.125 |
| $\beta$-TCVAE $\beta = 8$ | 1.5 | 1.375 |

description of the LINAC can be found in references such as [9]. Here we only provide a brief description on its underlying mechanism.

The synthetic beamline model has three main components, which covers ion injection, the chopper and the RF acceleration respectively. The ion injection component is based on the first-principle physics model of ion sources, where particles are generated. The ion source cannot be turned on instantaneously, therefore, there is a ramp-up phase of the ion beam, as depicted in the illustrative figures. This part of physics is modeled by six parameters. They are kept constant when the data were generated.

The chopper component models the beam chopper, which "chops" the ion beams into multiple "pulses" before they can be accelerated by the RF (radio frequency) cavities in the LINAC. The speed the chopper rotates and relative opening determine the number of pulses as well as the the active region of each pulse. They are modeled by two parameters in the synthetic model: *frequency* and *duty_cycle*. In the generated waveform, parameter *frequency* represents a categorical latent space, while *duty_cycle* represents a continuous latent space. These two parameters were permuted when the dataset were generated.

The RF acceleration component models the acceleration of the pulses by the RF cavities. The cavities not only accelerate the particles to high speed, but also groups them into separate bundles. This component is modeled by five parameters, and were kept constant in the data generation. A set of beam waveform generated by the synthetic model is shown in Figure. 1. A small amount of white noise has been added to reflect the stochastic nature of particle physics. Note that both x-axis (time) and y-axis (amplitude) have been normalized. Although only a handful of pulses are shown in the figure, hundreds of thousands of pulses can be present in a real accelerator.

The Beamsyntheiss dataset has a few advantages. Since it is based on well-understood particle physics, one can adjust the parameters to reflect the underlying physics to control the latent space represented in the data. Moreover, since the synthetic model is quick to run, it is possible to generate large quantity of

data for training and validation. We have found this dataset to be highly valuable in the development of the NashAE method. The dataset can be found at: `https://github.com/ericyeats/nashae-beamsynthesis`.

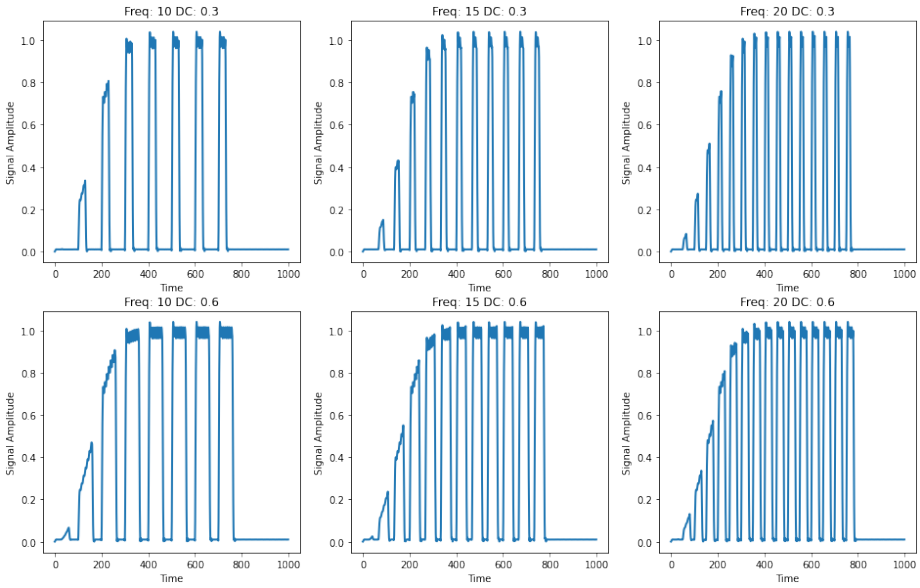


Fig. 1: Depiction of varying *frequency* (across a row) and *duty_cycle* (down a column) on the Beamsynthesis dataset. Freq: frequency, DC: duty_cycle

### Details on Learning Latent Feature Dimensionality (Beamsynthesis Dataset)

A total of 8 independent trials are run for each algorithm and each set of hyperparameters. The absolute difference between the number of *learned* latent features and the ground truth (2) is averaged between the 8 trials.

For this experiment, the presence of a *learned latent feature* is determined in the following manner. For NashAE, the difference between the maximum and minimum latent scores for each latent feature is recorded. If the difference is larger than or equal to 0.2, it is considered a *learned latent feature*. Otherwise it is not counted. Likewise, for $\beta$-TCVAE and FactorVAE, a latent variable is considered *learned* if the $\mu$ component of the VAE has a range greater than 2.0. The $\beta$-VAE approach is inspired by how the authors count *learned latent features* in the original work [3]. If the average learned variance across the dataset is less than or equal to 0.8, the latent is counted as a learned latent feature. Otherwise, if the average learned variance is greater than 0.8, it is not counted. We also tried thresholding the maximum $\mu$ score minus the minimum $\mu$ score (similar to

the criterion used for NashAE, $\beta$-TCVAE, and FactorVAE), but that resulted in consistently worse scores for $\beta$-VAE than the ones reported. We attribute this preferred method difference between $\beta$-VAE, $\beta$-TCVAE, and FactorVAE to the different weighting of the ELBO loss components during training.

### $\beta$-VAE Metric Details

Like in Higgins et al. [3], we classify for *x-position*, *y-position*, *scale*, and *orientation* on the dSprites dataset. The linear classifier is initialized with a Kaiming Normal distribution [2] and trained until convergence on batches of 100 examples each on a categorical crossentropy objective with a learning rate of 1.0. For good measure, the learning rate is set to 0.05 for the last 5% of batches. The linear classifier converges in all tests. For training and testing, the constant data generating factor used to craft the batch is chosen with uniform probability (e.g., each factor gets a probability of 0.25 to be used to craft the batch). The linear classifier accuracy scores are collected over 1000 batches of size 100 each. Recall from Higgins et al. [3] that only the average absolute difference across each difference-batch is used to train and evaluate the classifier.

We also evaluated the algorithms with the $\beta$-VAE disentanglement metric on the Beamsynthesis dataset. Both methods achieved a 100% score nearly 100% of the time. This was also true for NashAE $\lambda = 0$ and $\beta$-VAE $\beta = 1$, and even when only one latent variable was learned by either method on this dataset that has two independent data generating factors. In the latter case, only one of the two factors was captured, but the linear classifier could tell which of the factors was held constant by the *presence of* variation or *lack of* variation in the single learned variable. In general, the high accuracy is due to the dataset being relatively simple.

### Total AUROC Difference Details

**Checking Reasonableness of TAD** We design a simple experiment for two synthetic latent variables $\mathbf{z}_\alpha$ and $\mathbf{z}_\beta$ and one bernoulli data generating factor $\mathbf{c}$ in which $p(\mathbf{z}_\alpha|c=1) = \mathcal{N}(\mu, 1)$ and $p(\mathbf{z}_\alpha|c=0) = \mathcal{N}(-\mu, 1)$. $p(\mathbf{z}_\beta)$ is distributed similarly to $p(\mathbf{z}_\alpha)$, but it is correlated to $c$ with strength $r$. When $\mu = 1$ and $r = 0.0$, TAD is 0.417. When $\mu = 0.1$ and $r = 0.0$, TAD is 0.07. When $\mu = 1$ and $r = 0.9$, TAD is 0.043. When $\mu = 1$ and $r = -0.9$, TAD is 0.046. When $\mu = 1$ and $r = 0.0$, but there is a 1:50 true vs. false imbalance, TAD is 0.412. These experiments indicate that TAD increases for a confident, independent latent space, and it is not sensitive to class imbalance. We used 10k samples to compute TAD in each trial.

**TAD on CelebA** We measured TAD on the CelebA dataset [10]. Since we are interested in measuring the TAD for only *independent* attributes, we measure the proportion of entropy reduction of each attribute given knowledge of any other single attribute, and we disqualify any attribute with an entropy reduction greater than 0.2 from counting towards TAD. This disqualified attributes with high mutual information such as *male*, *wearing lipstick*, and *wearing earings*.

Furthermore, we are interested in measuring TAD for attributes which the algorithm is a good detector for. The algorithm is considered a good detector for an attribute if it has a maximum AUROC score of 0.75. If the attribute does not make this cutoff, it is not considered "learned" by the algorithm and does not count. Lower thresholds for the maximum AUROC (e.g. 0.7) still resulted in the same general trends as those presented in the paper.

**How AUROC is Calculated** We test 100 evenly spaced detector threshold values from the minimum value to the maximum value for each latent on a large number of samples (e.g., 5000). We calculate the TPR and FPR for each of these 100 thresholds and both potential orientations (+ is increasing in latent values or + is decreasing in latent values) and take the maximum of the two AUROCs to ensure that the minimum AUROC score is 0.5.

**Comparison with other Work** The TAD metric is akin to the SAP metric proposed by Kumar et al. [8], which uses difference in the factor classification accuracy of latent representations to measure the degree to which information of a single data generating factor is isolated in a single latent variable. However a key difference between SAP and TAD is that TAD is built off the AUROC score, which is threshold-independent and more informative for data with significant class imbalance (*e.g.*, CelebA [10]).

### Reconstruction Quality of Different Methods

We note that the reconstruction of NashAE tended to be better than that of the VAE-based methods, especially when $\beta >> 1$ in $\beta$-VAE. This is likely due to the information bottleneck component of the VAE loss, which both $\beta$-TCVAE and FactorVAE address by enhancing the total correlation (TC) component of the loss only.

### Learned Latent Spaces on CelebA

We include full learned NashAE latent space traversals for the interested reader. See figures 2, 3, 4, and 5. Each row of 10 images corresponds to traversing a feature in the latent space while all others are held constant. Empty rows correspond to "unused" latent features.

## References

1. Chen, R.T., Li, X., Grosse, R.B., Duvenaud, D.K.: Isolating sources of disentanglement in variational autoencoders. Advances in neural information processing systems **31** (2018)
2. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
3. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework (2016)

4. Kim, H., Mnih, A.: Disentangling by factorising. In: International Conference on Machine Learning. pp. 2649–2658. PMLR (2018)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
7. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Proceedings of the 31st international conference on neural information processing systems. pp. 972–981 (2017)
8. Kumar, A., Sattigeri, P., Balakrishnan, A.: Variational inference of disentangled latent concepts from unlabeled observations. arXiv preprint arXiv:1711.00848 (2017)
9. Lee, S.Y.: Accelerator physics. World Scientific Publishing Company (2018)
10. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (December 2015)
11. Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., Bachem, O.: Challenging common assumptions in the unsupervised learning of disentangled representations. In: international conference on machine learning. pp. 4114–4124. PMLR (2019)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32**, 8026–8037 (2019)

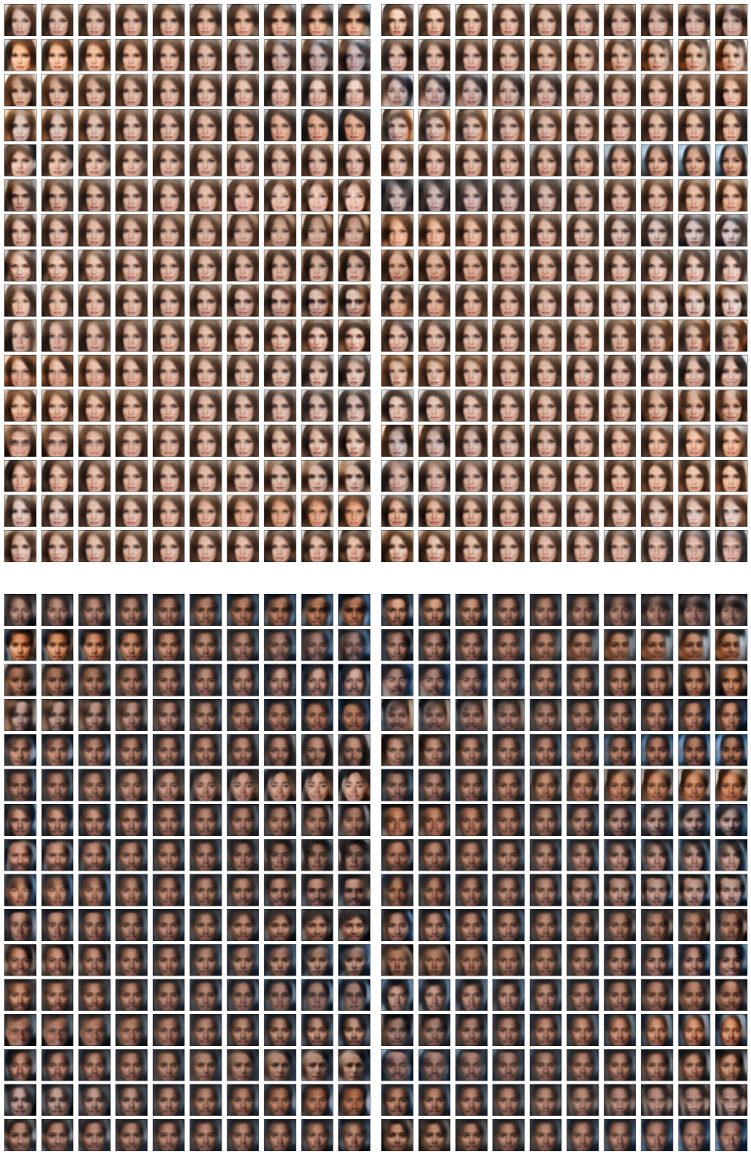

Fig. 2: Depiction of the latent space for NashAE $\lambda = 0.0$ (Part A)

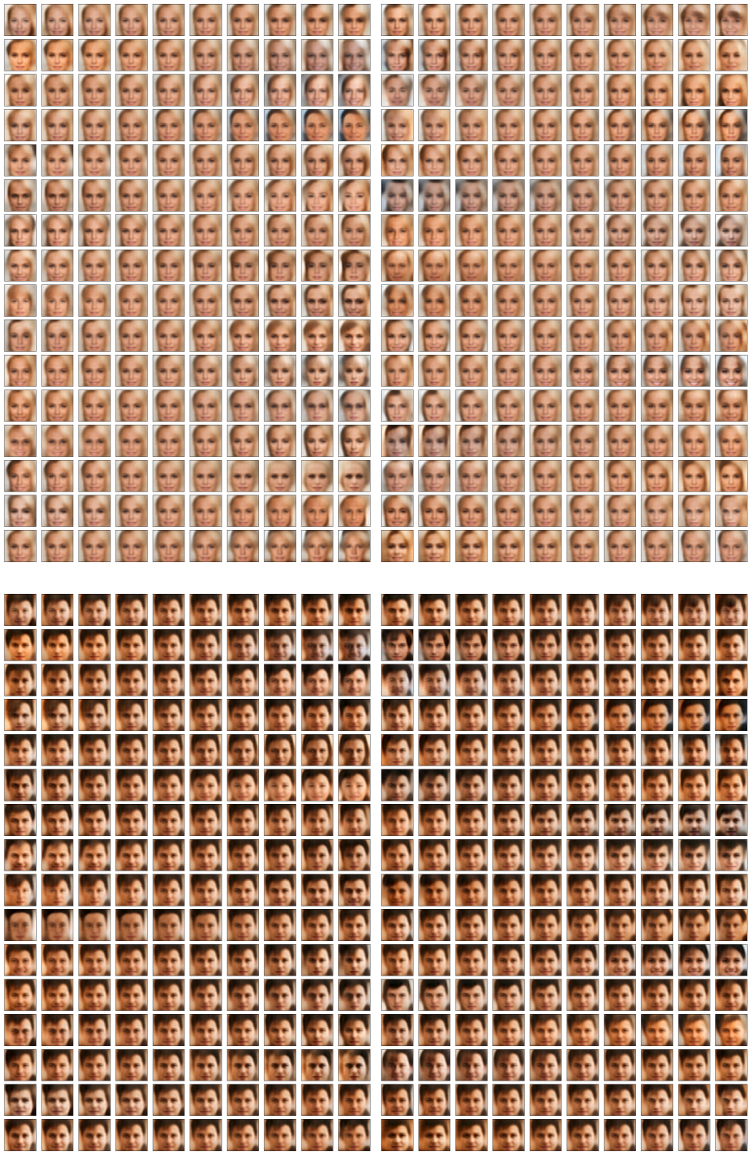

Fig. 3: Depiction of the latent space for NashAE $\lambda = 0.0$ (Part B)

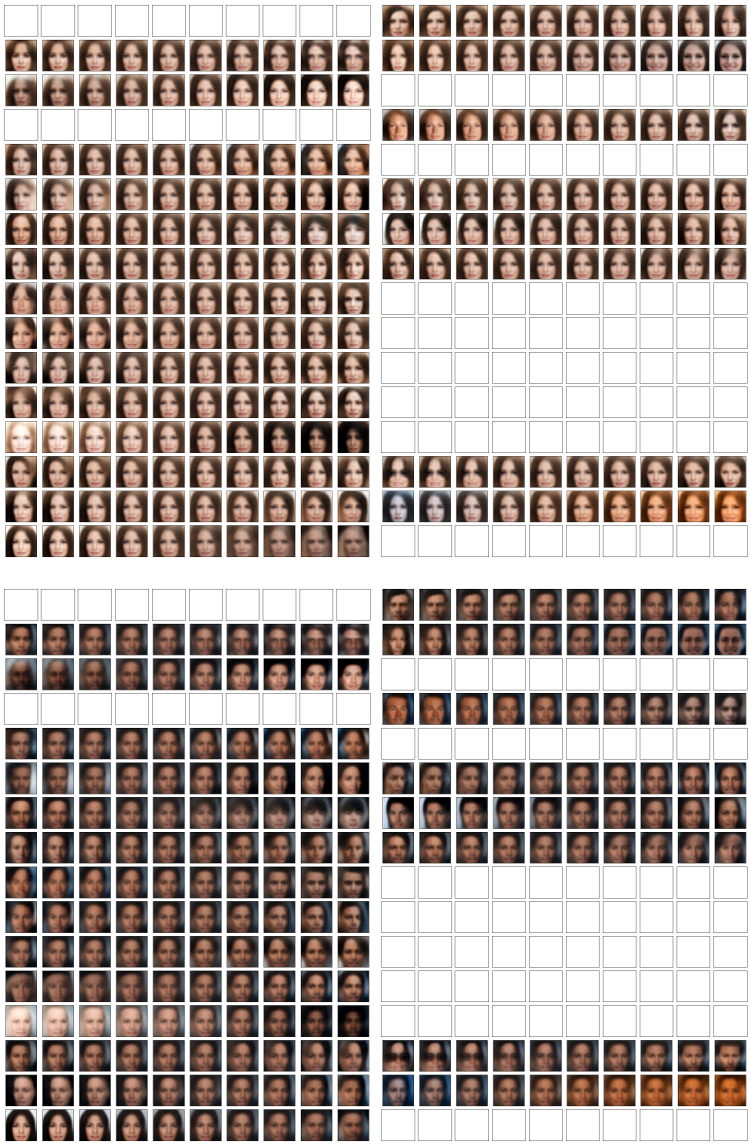

Fig. 4: Depiction of the latent space for NashAE $\lambda = 0.2$ (Part A). Blank rows correspond to unused latent features

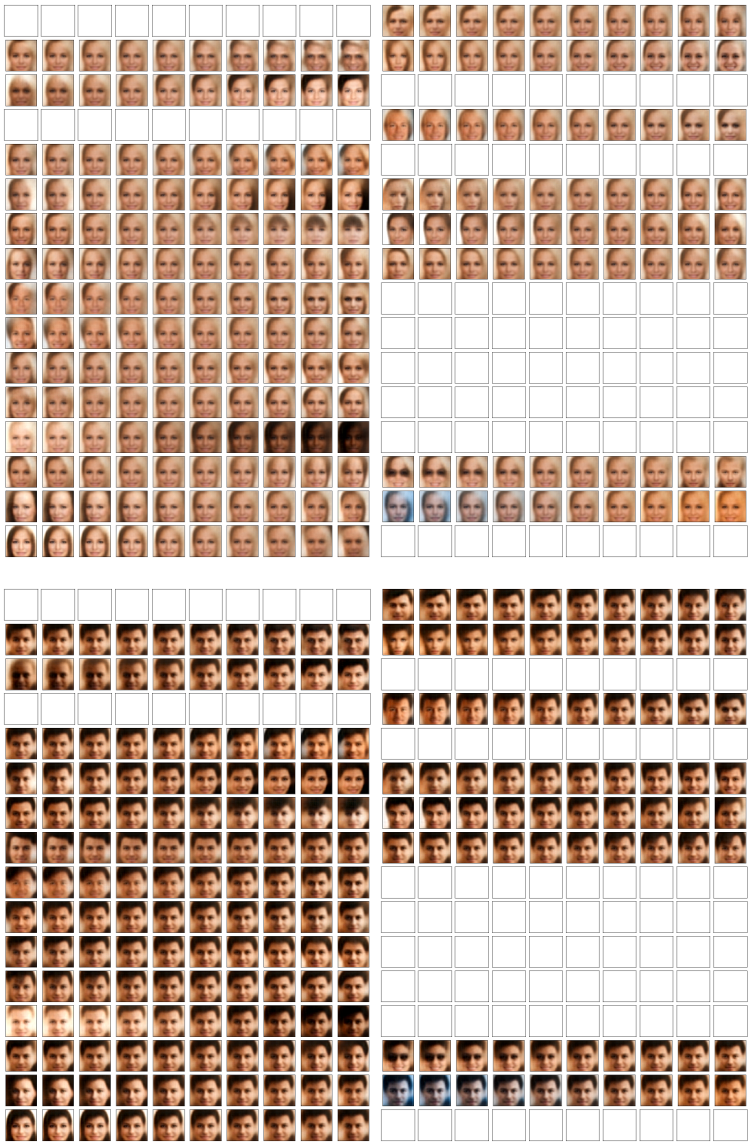

Fig. 5: Depiction of the latent space for NashAE $\lambda = 0.2$ (Part B). Blank rows correspond to unused latent features