Debiasing-VAE for Facial Recognition

Puck de Haan
11305150
pdehaan274@gmail.com

Paul ten Kaate
10743367
paultenkaate@outlook.com

Tim van Loenhout
10741577
timvanloenhout@gmail.com

Jan Erik van Woerden
11033711
janerikvw2006@gmail.com

Supervised by:
Simon Passenheim

ABSTRACT

An attempt was made to reproduce the method described in "Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure" (Amini & Soleimany et al., 2019). Since there was no legitimate open source implementation, the method had to be implemented from scratch. The algorithm is an in-process debiasing model for computer vision, applied on a facial recognition task. Debiasing is done with the help of a debiasing VAE (DB-VAE) that learns the latent space representing the data, which is subsequently used to adaptively resample more diverse batches from the data during training. We found that the specifications provided by the authors were not enough to accurately reproduce their model. Furthermore, these settings did result in a posterior collapse of the VAE, which we solved by shifting weight from the regularization loss to the reconstruction loss. In the end, we did obtain similar debiasing results, however at the expense of a slightly less accurate classifier. Altogether, the reproducability of the main paper is deemed insufficient.

1 INTRODUCTION

Since Artificial Intelligence (AI) is increasingly common in our society and many decisions are being made by Machine Learning (ML) systems, fairness in AI is a rapidly expanding field of research. ML systems are, among other purposes, already being used to control autonomous vehicles [15], to predict recidivism rates [4] and even to make medical diagnoses [14]. Such systems could have a significant impact on our society, thus it is important that the choices made by these systems are fair and unbiased. There are many different definitions of "fairness", but most research into this field seeks to mitigate the algorithmic discrimination of individuals based on protected attributes such as race or gender.

The paper by Amini & Soleimany et al. [1], that we will review and whose method we will reproduce for this assignment, focuses on fairness in computer vision – specifically facial recognition. This task has been shown to be subject to algorithmic bias for specific demographics. For instance, the facial recognition software used by the US law enforcement shows discrepancies in accuracy between groups of different race and/or gender [8]. To try and tackle this fairness problem in facial recognition, [1] proposes an in-process method to mitigate the bias. The method's debiasing capabilities are directly integrated in the training process of the model – both automatically and unsupervised – to compensate for the implicit bias embedded in training data. To uncover this implicit bias, the model simultaneously learns a classification task and the underlying latent space of the data.

FACT '20, January, MSc AI, UvA
2020.

2 METHOD

Background

Facial recognition can be implemented as a binary classification task: an image should be classified either as a face or not a face. Thus, for every datapoint $\boldsymbol{x}$ the model has to predict a class $\hat{y}$ that matches the label y. Instead of manually defining variables for each datapoint, it is assumed that every datapoint has a corresponding latent variable vector $\boldsymbol{z} \in \mathbb{R}^k$ that captures its implicit properties. The authors of the paper define a notion of algorithmic fairness that takes these latent variables into account. Namely, if the outcome of a classification model does not change when a set of latent features $\boldsymbol{z}$ is introduced, the model can be considered fair. Thus, a classifier model (f_θ) is fair if: $f_\theta(x) = f_\theta(x, z)$.

With this notion of fairness, it is only possible to ensure a fair classifier if the samples in the dataset are (approximately) uniformly distributed over the latent space. Furthermore, this notion allows us to analytically measure the bias of the classifier by calculating the classification accuracy over different latent variables and measuring the variance between these accuracies.

Algorithm

The proposed model to accomplish the debiasing task by using latent variables, is the debiasing-VAE (DB-VAE, Figure 1). This extended version of a VAE [7] can be used for classification, while simultaneously debiasing the dataset during training. The latter is done by adaptively resampling the training batches, unsupervised and dependent on the distribution of the learned latent variables.

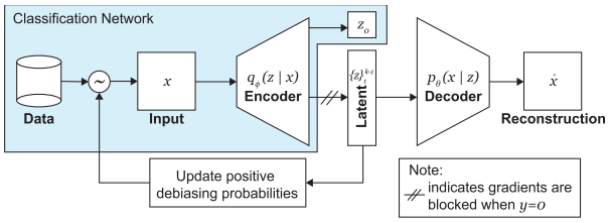

Figure 1: Schematic overview of the DB-VAE framework.

Recovering Latent Variables. Identical to the original VAE, the encoder of the DB-VAE learns the approximate posterior distribution $q_\phi(z|x)$ of the latent space. However, whereas the encoder of the original VAE produces $2k$ latent variables as output, the encoder of the DB-VAE produces d additional output variables for classification where $\hat{y} \in \mathbb{R}^d$. This results in a total of $2k + d$ variables, respectively consisting of the mean and standard deviation of $z \in \mathbb{R}^k$ for a datapoint and its predicted class. The decoder architecture is the same as in the original VAE model and is used to reconstruct samples from the latent space.

The DB-VAE loss function is slightly different from the loss function of a regular VAE model. It is a weighted combination of three different components: the reconstruction loss (1), the regularization loss (2) and the classification loss (3). These losses are defined by the L_p norm between input x and reconstruction $\hat{x}$, the KL-divergence between the approximate posterior and a unit Gaussian [7] and the cross-entropy loss.

$$\mathcal{L}_{\text{recon}} = \|x - \hat{x}\|_p \tag{1}$$

$$\mathcal{L}_{\text{reg}} = \frac{1}{2}\sum_{j=0}^{k-1}(\sigma_j + \mu_j^2 - 1 - \log\sigma_j) \tag{2}$$

$$\mathcal{L}_{\text{class}} = \sum_{i\in 0,1} y_i \log\frac{1}{\hat{y}_i} \tag{3}$$

One important detail to take into account when training the model, is that one should only backpropagate the gradients of the decoder and latent space for the classes that require debiasing. In the case of facial recognition, we are only interested in debiasing the images of faces. Hence, when a non-face (irrelevant class) is encountered during training, only the gradients from classification should be backpropagated.

Unsupervised Debiasing. To ensure that the samples used for training are uniformly distributed across the latent space, the algorithm adaptively resamples datapoints. To do so, the latent space $Q(z|X)$ over the entire distribution of the dataset is approximated with the encoder of the DB-VAE. This distribution is then used to identify the over- and under-represented regions in latent space, which indicate the frequency of the samples in the dataset. Subsequently, the algorithm drops the over-represented regions in latent space, as a result increasing the sampling probability of the less frequent samples in under-represented regions. Consequently, the distribution of the samples used for training is evened out and becomes more uniform.

In order to simplify this debiasing approach, we approximate $Q(z|X)$ with histogram $\hat{Q}(z|X)$, whose dimensions are defined by the dimension of z. Since a high-dimensionality of z could still result in a very complex histogram, it is simplified even further by using independent histograms to approximate the joint distribution. This distribution is then used, together with the debiasing parameter α, to calculate the final probability of sampling a datapoint x (Equation 4). Furthermore, the debiasing parameter can be tuned to adjust the level of debiasing. For instance, when $\alpha \to 0$ the debiasing factor increases and when $\alpha \to \infty$ the debiasing is nullified.

$$\mathcal{W}(z(x)|X) \approx \prod_{i=0}^{k} \frac{1}{\hat{Q}_i(z_i(x)|X) + \alpha} \tag{4}$$

Thus the complete debiasing process consists of every epoch feeding the entire training dataset through the model to retrieve $z(x)$, updating the probability histograms accordingly. Subsequently, during training, every new batch is drawn from X with likelihood $\mathcal{W}(z(x)|X)$. As a result, the model trains on a subset that is not manually chosen, but learned from the latent space.

Implementation

The paper [1] mentions an open-source implementation of the algorithm, however were not able to localize it. Instead we consulted a publicly available iPython notebook written by one of the authors (A. Amini) that implements the algorithm [1]. Although it was a helpful source, there were quite some issues: a different sampling method as in the paper was used and the algorithm had to be transferred to Pytorch, since the authors used Tensorflow as their framework. Altogether, the implementation basically had to be written from scratch.

Furthermore, the authors neglected to specify some crucial design choices such as the latent dimension, loss weights, reduction of the reconstruction loss and the used padding for the convolutional layers. As a result, we had to make decisions on the model architecture ourselves. However, since the model turned out to be very sensitive to changes in the architecture, extensive experimentation was required to produce a working model.

On top of these unspecified design choices, several other design choices mentioned in the paper did not work in practice. Primarily, some issues came up with the adaptive resampling: the product over the latent variable probabilities,

[1] https://github.com/aamini/introtodeeplearning/tree/master/lab2

as described in Equation 5, resulted in numerical underflow. Therefore, we decided to adapt the resampling method as used in the aforementioned iPython Notebook. Instead of combining all features, this method takes the maximum probability over these features as the overall resampling probability. By only taking the feature with maximum probability into account we compromise on complexity, but end up with a numerically stable method. Furthermore, we had to discard the ReLU activation in the output of the encoder (see Section 3).

3 EXPERIMENTAL SETUP

To evaluate the proposed method we trained several DB-VAE models with different settings to classify whether an image contains a face or not. For testing we used a separate dataset from the training and validation dataset, containing faces with variation in gender and skin color. After deciding on the final settings, we tested several different values for α to see which value results in the lowest measure of bias and highest overall accuracy on the test set.

Datasets

The classifier was trained on a subset of the dataset used in the orginal paper, which consists of 109.914 samples. 50% of the images are of faces and 50% of non-faces, which are respectively retrieved from the CelebA dataset [9] and randomly sampled from all catergories on ImageNet [3]. The face images are cropped to 64x64 with a focus on the face bounding box and the non-face images are randomly cropped to 64x64. Finally, the complete dataset is split into 80% training data and 20% validation data.

For testing, as in the original paper, the PBB dataset [2] is used, which contains 1270 images of faces that are annotated with the corresponding gender (male and female) and skin tone (lighter and darker). Since the PBB dataset does not contain any face bounding box information, we extracted differently sized patches from every image. Subsequently, the image is classified as a face if any of its patches is classified as a face.

Layer	Input	Kernel Size	Filters	Stride	Padding	Activation	Batchnorm
Conv 1	Dataset	5x5	12	2	2	ReLU	Yes
Conv 2	Conv 1	5x5	24	2	2	ReLU	Yes
Conv 3	Conv 2	5x5	48	2	2	ReLU	Yes
Conv 4	Conv 3	5x5	72	2	2	ReLU	Yes

Flatten output size [batch x 4 x 4 x 72] to [batch x 1152]

Layer	Input	Size	Activation	Batchnorm
Fully Connected (FC) 1	Conv 4	1152 x 1000	ReLU	Yes
FC mean	FC 1	1000 x 100	-	-
FC logvar	FC 1	1000 x 100	-	-
FC class.	FC 1	1000 x 1	-	-

Table 1: Encoder architecture

Layer	Input	Size	Activation	Batchnorm
FC 1	z	100 x 1000	ReLU	Yes
FC 2	FC 1	1000 x 1152	ReLU	Yes

Shape [batch x 1152] output to [batch x 4 x 4 x 72]

Layer	Input	Kernel	Filters	Stride	Pad.	Output Pad.	Activation	Batchnorm
Deconv 1	FC 1	5x5	72	2	2	1	ReLU	Yes
Deconv 2	Deconv 1	5x5	48	2	2	1	ReLU	Yes
Deconv 3	Deconv 2	5x5	24	2	2	1	ReLU	Yes
Deconv 4	Deconv 3	5x5	12	2	2	1	Sigmoid	

Table 2: Decoder architecture

Model

The DB-VAE model consists of two successive networks: an encoder and a decoder of which the architectures can be found in Tables 1 and 2. A latent variable vector $\boldsymbol{z}$ is sampled from the produced mean and log variance vectors by applying the reparameterization trick [7]. Subsequently, the decoder reconstructs the original output given this latent vector.

Design Choices

While we made an effort to follow the model architecture as described in [1], we had to make multiple design choices by ourselves. Either due to specifications being left out the paper, or the results not being reproducible with the specified settings. First of all, the authors did not specify the padding they used for the convolutional layers and their iPython Notebook used a padding setting that is not available in Pytorch. We found that a padding of 2 for every convolutional layer was sufficient – together with an additional output padding to the right and bottom for the last two deconvolutional layers – to obtain a reconstructed image of 64 x 64 pixels.

It was stated in the paper [1] that all layers of the model are followed by a ReLU activation. However, a ReLU after the mean and log-var layer forces the mean and the variance of $\boldsymbol{z}$ to be higher than 0 and 1 respectively. This does not seem optimal since the regularization loss tries to force the approximate distribution of the latent space to resemble a unit Gaussian. As this becomes almost impossible when the mean cannot take a value below 0, we did not use an activation function for the mean and log-var layers. Furthermore, after the final deconvolutional layer, a sigmoid activation is added to map the output for every pixel of the reconstructed image to a value between 0 and 1. This is preferred as the network output represents a matrix of float RGB values that should be in this range.

The L_2 norm is used for the reconstruction loss by the authors. However, it is not specified whether the average or sum over the pixel losses for an image should be used. During our exploratory research, we found that the total loss was very sensitive to choices on these settings. Therefore, results for both an L_2 loss with a mean and a sum reduction will be presented.

Furthermore, the paper describes a weighted loss function, yet the values for these weights are never specified. However, the interactions between the different components of the loss have a significant impact on the training process and are thus of fundamental importance to the model outcomes. On top of that, the authors employ a L_2 norm for the reconstruction, but do not detail whether the average or sum over the pixel losses for an image should be used. Although we found in our exploratory research that the total loss was very sensitive to choices on these settings, we did not have the means to do an extensive grid search on this matter. Nevertheless, we discovered that by using equal weights for each component, but shifting some emphasis from the regularization to the reconstruction loss by applying the sum instead of mean, we obtained a correctly working VAE, while preserving sufficient focus on the classification component.

Also, there was no mention of a specific latent dimension in the paper, which during initial tests we found to produce good results with size 100. Finally, the authors also did not specify any optimizer, therefore we used Adam [6] with default learning rate, since this optimizer was also used in the previously mentioned notebook.

Using the above mentioned settings and different values for α, we trained each model 5 times for 60 epochs (5000 iterations). Since the model appeared fairly unstable and occasionally displayed big discrepancies between runs, we used the best 3 accuracies per model for the final results.

4 RESULTS

Training

The loss curve of the model where the mean reconstruction loss is taken, is most similar to the loss curve depicted in the paper. Although we used the summed reconstruction loss for further training to prevent posterior collapse, its loss curve is very unstable and in a different range (Figure 2).

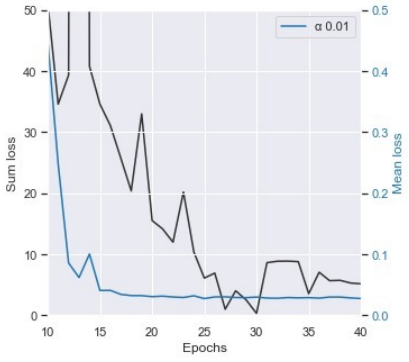

Figure 2: Loss curves during training when using the average vs summed reconstruction error.

Adaptive Resampling

In Figure 3 the sampling probabilities of all samples in the training set after debiasing can be seen. It is clear that the more frequent samples, the majority, have a lower sampling probability than the rarer samples.

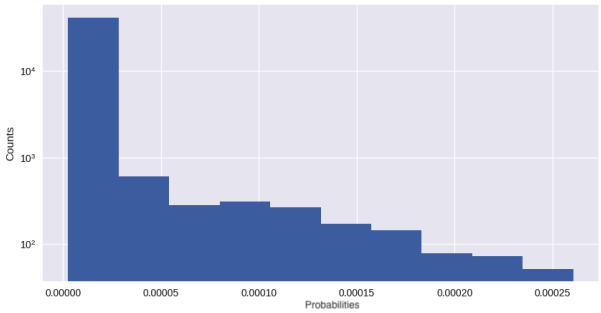

Figure 3: Sampling probabilities over the training data set, while training with $\alpha = 0.001$

Additionally, Figure 4 also show that the adaptive resampling indeed changes the images the model samples for training. The faces with the lowest resampling probability are all uniformly looking faces of (mostly) light skinned women, whereas the faces with the highest resampling probability are very diverse.

Figure 4: Top 12 faces with the lowest and highest sampling probabilities.

Test accuracy

Figure 5 implies that higher values of α do have a debiasing effect. However, they also cause a slight decrease in overall accuracy (indicated by recall, since the test-set only contains faces). This effect is supported in table 3 which shows a decreasing trend for both the overall accuracy and the variance between the accuracies (indicating the bias). $\alpha = 0.001$ seems to be a too strong debiasing factor, since the accuracy decreases vastly when this value is used.

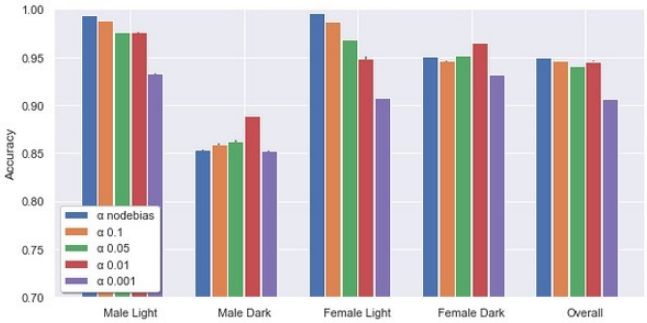

Figure 5: Accuracy on test-set using sum reduction of the reconstruction loss during training.

	$\mathbb{E}[\mathcal{A}]$: Recall	$Var[\mathcal{A}]$: Measure of Bias
No debias	92.39	46.20
$\alpha = 0.1$	91.78	36.39
$\alpha = 0.05$	90.76	32.33
$\alpha = 0.01$	90.52	22.00
$\alpha = 0.001$	85.22	22.42

Table 3: Recall and bias on PPB test-set

5 DISCUSSION

The findings we obtained partly match up with the debiasing results reported in the paper. That is, the sampling probability distribution of the training data is similar to the one depicted in the paper and subsequently the sampled batches during training are indeed more diverse with debiasing. However, although the measure of bias decreases when adaptive resampling is applied, the overall classification accuracy decreases. Hence, we did succeed in promoting fairness through a more debiased classificaton model, but at the cost of a slightly worse classifier.

On top of the fairness-quality trade-toff, we also achieved a slightly lower overall accuracy than the paper. Given this also holds for the biased model, this discrepancy is most likely caused by differences in model architecture. Since the model is fairly unstable and highly dependent on this architecture, all experimental settings that were not mentioned in the paper should be tested more extensively. Moreover, when using settings that produce a similar loss curve as in the paper, the model shows a posterior collapse on the reconstruction part of the model [11] (see Figure 7 in the appendix for illustration). Despite the workings of the decoder being a means to an end, namely that of debiased classification, we believe that a reliable VAE should not always produce the same reconstruction, regardless its input. Drawing conclusions based on a VAE that does not work as intended largely discounts the transparency and implied fairness of the model. Therefore, we shifted some emphasis from the generalization loss to the reconstruction loss by using a sum instead of a mean over the latter. As a result, the model became able to reconstruct more accurate resemblances of the inputs (Figure 8), without compromising on the classification accuracy.

Clearly, the main advantage of the proposed model is that it is indeed capable of debiasing data during training by consulting the learned latent variables. On the other hand, since the exact proposed method made for numerical issues, the adaptive resampling method was not instantly reproducable from the paper. We therefore applied an adjusted resampling method. A major drawback of the model is that the impact of different α values depends strongly on the model settings, including on the formula used for adaptive resampling. A drawback of our alternative resampling method is that by considering only the most extreme feature for the resampling probability, the debiasing mechanism could potentially emphasize mainly on the rare cases. As a result, the rarest cases (e.g. blue hair, hats, etc.) could provide the highest probabilities for the sampling process, albeit maybe not the most relevant for the debiasing objective.

6 BROADER IMPLICATIONS

There is a comparable fair model that makes use of a VAE, where chosen priors are used that encourage the model to be invariant to sensitive factors in the data [10]. In contrast, the DB-VAE model does not require such manual selection of senstive features. This unsupervised approach does not only contribute to fairness in AI, but also confidentiality, as there are laws emerging stating that sensitive data may not be used at all. For many debiasing models this could pose a problem, that is; how to ensure you are not discriminating when you are not allowed to identify protected groups in your data [5]? However, since the DB-VAE model learns the sensitive features unsupervised, it is able to preserve fairness without requiring any such sensitive information in the data. The only issue is that it might be possible to deduce the sensitive properties of every data point from the latent space learned by the VAE. This would create a new confidentiality problem, while solving another.

Furthermore, accountability and transparency should also be taken into account for further research in AI. These concepts aim at fostering trust in machine learning models [12], which is fundamental when taking actions based on an algorithmic prediction, according to [13]. Although the method we reviewed does not directly contribute to understanding the cause of such predictions, it is able to provide a degree of transparency with regards to its added debiasing feature. That is, the specific attributes that cause higher sampling probabilities could be deduced from the latent space. Knowing which latent features are deemed underrepresented in a dataset by the algorithm and acting on this information, would increase the transparency of a decision making process. Subsequently, this knowledge could help in decreasing the uncertainty about algorithm accountability.

7 CONCLUSION

In the end we were able to produce a model that could do debiasing through adaptively resampling the training data. However, it took extensive testing and some adjustments to the described method to produce similar results. The lack of detail in the description of the method, together with some of the infeasible design choices that were made by the authors, makes it difficult to award an ACM badge to the paper. On the one hand, we did succeed in implementing the adaptive resampling method, but on the other hand this was not possible with just the information as provided in the paper. That is why we decided to grant the paper the reproducibility badge, but only with severe restrictions.

REFERENCES

[1] Alexander Amini, Ava P. Soleimany, Wilko Schwarting, Sangeeta N. Bhatia, and Daniela Rus. 2019. Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure *(AIES '19)*. Association for Computing Machinery, New York, NY, USA, 289–295. https://doi.org/10.1145/3306618.3314243

[2] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. 77–91.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[4] Julia Dressel and Hany Farid. 2018. The accuracy, fairness, and limits of predicting recidivism. *Science Advances* 4, 1 (2018). https://doi.org/10.1126/sciadv.aao5580

[5] Matthew Jagielski, Michael Kearns, Jieming Mao, Alina Oprea, Aaron Roth, Saeed Sharifi-Malvajerdi, and Jonathan Ullman. 2018. Differentially Private Fair Learning. arXiv:cs.LG/1812.02696

[6] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. arXiv:cs.LG/1412.6980

[7] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. arXiv:stat.ML/1312.6114

[8] B. F. Klare, M. J. Burge, J. C. Klontz, R. W. Vorder Bruegge, and A. K. Jain. 2012. Face Recognition Performance: Role of Demographic Information. *IEEE Transactions on Information Forensics and Security* 7, 6 (Dec 2012), 1789–1801. https://doi.org/10.1109/TIFS.2012.2214212

[9] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*. 3730–3738.

[10] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2015. The Variational Fair Autoencoder. arXiv:stat.ML/1511.00830

[11] James Lucas, George Tucker, Roger Baker Grosse, and Mohammad Norouzi. 2019. Understanding Posterior Collapse in Generative Latent Variable Models. In *DGS@ICLR*.

[12] José Mena, Oriol Pujol, and Jordi Vitrià. 2019. Dirichlet uncertainty wrappers for actionable algorithm accuracy accountability and auditability. arXiv:cs.LG/1912.12628

[13] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR* abs/1602.04938 (2016). arXiv:1602.04938 http://arxiv.org/abs/1602.04938

[14] Nguyen Dang Thanh, Mumtaz Ali, et al. 2017. A novel clustering algorithm in a neutrosophic recommender system for medical diagnosis. *Cognitive Computation* 9, 4 (2017), 526–544.

[15] Mitch Waldrop. 2015. Autonomous vehicles: No drivers required. *Nature* 518 (02 2015), 20–3. https://doi.org/10.1038/518020a

A WHO DID WHAT

Equal contribution. Everybody worked together on the implementation of the method, since a lot of brainstorming had to be done to come up with a solution for the missing parameters. Towards the end of the project, **Puck** and **Tim** focused more on writing the report, while **Paul** created the iPython Notebook and **Jan Erik** retrieved and visualized the results from the final model.

B ADDITIONAL FIGURES

Additional figures produced by the model.

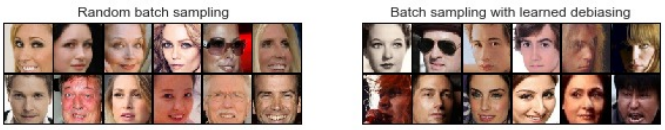

Figure 6: Faces in a batch with random (left) and adaptive (right) resampling.

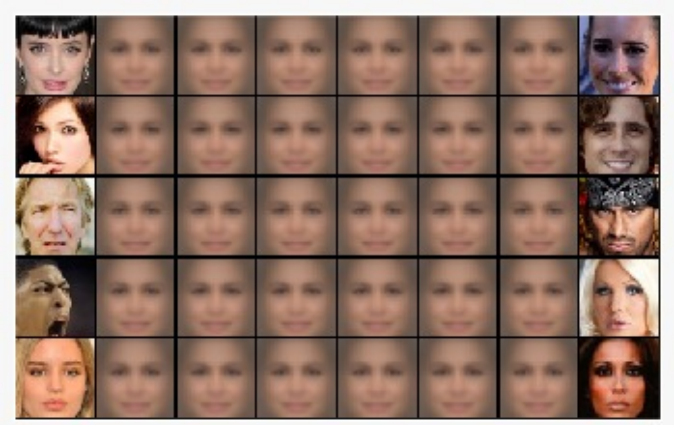

Figure 7: Posterior collapse when interpolating in the latent space when using average reconstruction loss.

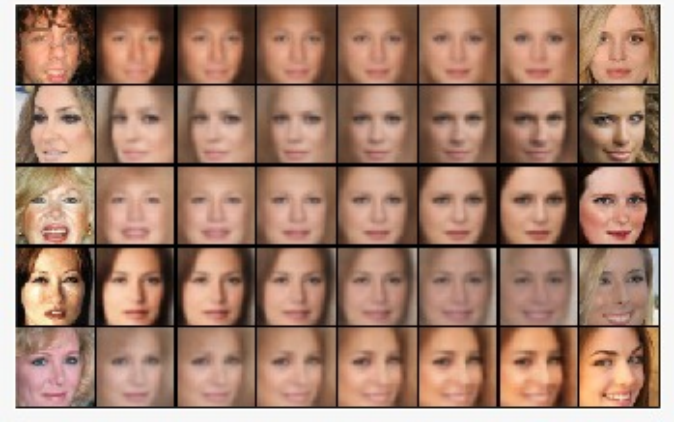

Figure 8: Interpolating in the latent space when using summed reconstruction loss

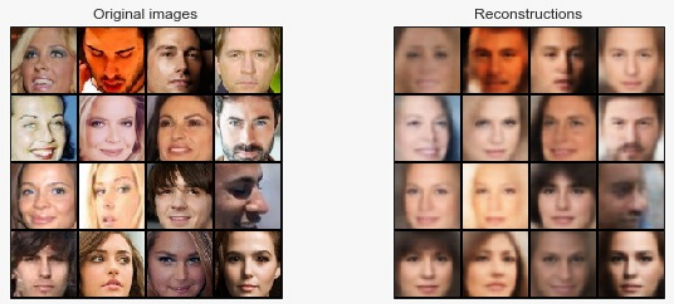

Figure 9: Image reconstruction when using summed reconstruction loss.