arXiv:1903.02709v1 [stat.ML] 7 Mar 2019

ADVERSARIAL MIXUP RESYNTHESIZERS

Christopher Beckham[1,3], **Sina Honari**[1,2], **Alex Lamb**[1,2], **Vikas Verma**[†,1,6], **Farnoosh Ghadiri**[1,3], **R Devon Hjelm**[1,2,5] & **Christopher Pal**[1,3,4,‡*]
[1]Mila - Québec Artificial Intelligence Institute, Montréal, Canada
[2]Université de Montréal, Canada
[3]Polytechnique Montréal, Canada
[4]Element AI, Montréal, Canada
[5]Microsoft Research, Montréal, Canada
[6]Aalto University, Finland
firstname.lastname@mila.quebec
[†] vikas.verma@aalto.fi, [‡] christopher.pal@polymtl.ca

ABSTRACT

In this paper, we explore new approaches to combining information encoded within the learned representations of autoencoders. We explore models that are capable of combining the attributes of multiple inputs such that a resynthesised output is trained to fool an adversarial discriminator for real versus synthesised data. Furthermore, we explore the use of such an architecture in the context of semi-supervised learning, where we learn a mixing function whose objective is to produce interpolations of hidden states, or masked combinations of latent representations that are consistent with a conditioned class label. We show quantitative and qualitative evidence that such a formulation is an interesting avenue of research.

1 INTRODUCTION

The autoencoder is a fundamental building block in unsupervised learning, found in numerous seminal works such as variational autoencoders (Kingma & Welling, 2013). Autoencoders are trained to reconstruct their inputs after being processed by two neural networks: an encoder which encodes the input to a high-level representation or *bottleneck*, and a decoder which performs the reconstruction using the representation as input. One primary goal of the autoencoder is to learn representations of the input data which are useful (Bengio, 2012), which may help in downstream tasks such as classification (Zhang et al., 2016; Hsu et al., 2018) or reinforcement learning (van den Oord et al., 2017; Ha & Schmidhuber, 2018). The representations of autoencoders can be encouraged to contain more 'useful' information by restricting the size of the bottleneck, through the use of input noise (e.g., in denoising autoencoders, Vincent et al., 2008), through regularisation of the encoder function (Rifai et al., 2011), or by introducing a prior (Kingma & Welling, 2013). Another goal is in learning interpretable representations (Chen et al., 2016; Jang et al., 2016).

In unsupervised learning, learning often involves qualitative objectives on the representation itself, such as disentanglement of latent variables (Liu et al., 2017; Thomas et al., 2017) or maximisation of mutual information (Chen et al., 2016; Belghazi et al., 2018; Hjelm et al., 2018).

Mixup (Zhang et al., 2017) and *manifold mixup* (Verma et al., 2018) are regularisation techniques that encourage linear behavior on a deep neural network, such as a classifier. These methods artificially augment the training set by producing random convex combinations between pairs of examples and their corresponding labels and training the network on these combinations. This has the effect of creating smoother decision boundaries, which can have a positive effect on generalization performance. In Verma et al. (2018), the random convex combinations are computed in the *hidden space* of the network. This procedure can be viewed as using the high-level representation of the network to produce novel training examples and provides improvements over strong baselines in the supervised learning.

[*]Canada CIFAR AI Chair

In this paper we explore the use of manifold mixup for *unsupervised learning* by performing the mixup operation in the bottleneck of an autoencoder. In order to ensure that the output of the decoder given the mixed representation resembles the input at the pixel level, we leverage adversarial learning (Goodfellow et al., 2014), training a discriminator to distinguish between decoded mixed and unmixed representations. This technique allows the ability to simulate novel data points (such as those corresponding to combinations of annotations not present in the training set). Furthermore, we explore our approach in the context of semi-supervised learning, where we learn a mixing function whose objective is to produce interpolations of hidden states consistent with a conditioned class label.

1.1 Related work

Our method can be thought of as an extension of autoencoders that allows for sampling through interpolations. Variational autoencoders (VAEs, Kingma & Welling, 2013) can also be thought of as a similar extension of autoencoders, using the outputs of the encoder as parameters for an approximate posterior $q(\mathbf{z}|\mathbf{x})$ which is matched to a prior distribution $p(\mathbf{z})$ through the evidence lower bound objective (ELBO). At test time, new datapoints are sampled by passing samples from the prior, $\mathbf{z} \sim p(\mathbf{z})$, through the decoder. In contrast, our we sample a random mixup operation between the representations of two inputs from the encoder.

There are other works which sample interpolations as part of an unsupervised objective. Berthelot et al. (2018) uses a discriminator network to predict the mixing coefficient from the decoder output of the mixed representation, and the autoencoder tries to 'fool' the discriminator, making interpolated points indistinguishable from real ones. Sainburg et al. (2018) uses a BEGAN framework with an additional interpolation-based adversarial objective. Some elements which differentiates our work from theirs is that we perform an exploration into different kinds of mixing functions, including a semi-supervised variant which can produce mixes consistent with a class label.

2 Formulation

Let us consider an autoencoder model $F(\cdot)$, with the encoder part denoted as $f(\mathbf{x})$ and the decoder $g(\mathbf{x})$. In an autoencoder we wish to minimise the reconstruction, which is simply:

$$\min_F \mathbb{E}_{\mathbf{x}\sim\mathbf{p}(\mathbf{x})} ||\mathbf{x} - g(f(\mathbf{x}))||_2 \tag{1}$$

Because autoencoders trained in this fashion tend to produce images which are slightly blurry, one can train an adversarial autoencoder (Makhzani et al., 2015), but instead of putting the adversary on the bottleneck, we put it on the reconstruction, and the discriminator (denoted D) tries to distinguish between real and reconstructed $\mathbf{x}$, and the autoencoder (which is analogous to the generator) tries to construct 'realistic' reconstructions so as to fool the discriminator. Because of this, we coin the term 'ARAE' (adversarial reconstruction autoencoder). This can be written as:

$$\begin{aligned} &\min_F \ \mathbb{E}_{\mathbf{x}\sim\mathbf{p}(\mathbf{x})} \ \lambda||\mathbf{x} - g(f(\mathbf{x}))||_2 + \ell_{GAN}(g(f(\mathbf{x})), 1) \\ &\min_D \ \mathbb{E}_{\mathbf{x}\sim\mathbf{p}(\mathbf{x})} \ \ell_{GAN}(D(\mathbf{x}), 1) + \ell_{GAN}(D(g(f(\mathbf{x}))), 0), \end{aligned} \tag{2}$$

where ℓ_{GAN} is a GAN-specific loss function. In our case, ℓ_{GAN} is the binary cross-entropy loss, which corresponds to the Jenson-Shannon GAN (Goodfellow et al., 2014).

One way to use the autoencoder to generate novel samples would be to encode two inputs $\mathbf{h}_1 = f(\mathbf{x_1})$ and $\mathbf{h}_2 = f(\mathbf{x_2})$ into their latent representation, perform some combination between them, and then run the result through the decoder $\mathbf{g}(\cdot)$. There are many ways one could combine the two latent representations, and we denote this function $\text{Mix}(\mathbf{h_1}, \mathbf{h_2})$. Earlier we mentioned 'manifold mixup' (Verma et al., 2018), in which the authors propose a method to learn more discriminative features (which resulted in improved classification performance) by mixing in the hidden space through convex combinations:

$$\text{Mix}_{\text{mixup}}(\mathbf{h}_1, \mathbf{h}_2) = \lambda\mathbf{h}_1 + (1-\lambda)\mathbf{h}_2, \tag{3}$$

where $\lambda \in [0,1]^{(bs,)}$ is sampled from either a Beta or Uniform$(0,1)$ distribution and bs denotes the minibatch size. Alternatively, one could randomly retain some latent variables from $\mathbf{h}_1$ and use the

rest from $\mathbf{h}_2$, and in this case we would randomly sample a binary mask $\mathbf{m} \in \{0,1\}^{(bs \times f)}$ (where f denotes the number of feature maps) and perform the following operation:

$$\text{Mix}_{\text{Bernoulli}}(\mathbf{h}_1, \mathbf{h}_2) = \mathbf{m}\mathbf{h}_1 + (1 - \mathbf{m})\mathbf{h}_2, \tag{4}$$

where $\mathbf{m}$ is sampled from a Bernoulli(p) distribution (p can simply be sampled uniformly).

With this in mind, we propose the 'adversarial mixup resynthesiser' (AMR), where part of the autoencoder's objective is to produce mixes which, when decoded, are indistinguishable from real images:

$$\begin{aligned}
&\min_F \ \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim \mathbf{p}(\mathbf{x})} \ \underbrace{\lambda ||\mathbf{x} - g(f(\mathbf{x}))||_2}_{\text{reconstruction}} + \underbrace{\ell_{GAN}(g(f(\mathbf{x})), 1)}_{\text{fool D with reconstruction}} + \underbrace{\ell_{GAN}(g(\text{Mix}(f(\mathbf{x}), f(\mathbf{x}'))), 1)}_{\text{fool D with mixes}} + \\
&\qquad\qquad \underbrace{\beta ||\text{Mix}(f(\mathbf{x}), f(\mathbf{x}')) - f(g(\text{Mix}(f(\mathbf{x}), f(\mathbf{x}'))))||_2}_{\text{mixing consistency}} \\
&\min_D \ \mathbb{E}_{\mathbf{x},\mathbf{x}' \sim \mathbf{p}(\mathbf{x})} \ \underbrace{\ell_{GAN}(D(\mathbf{x}), 1)}_{\text{label x as real}} + \underbrace{\ell_{GAN}(D(g(f(\mathbf{x}))), 0)}_{\text{label reconstruction as fake}} + \underbrace{\ell_{GAN}(g(\text{Mix}(f(\mathbf{x}, f(\mathbf{x}'))), 0)}_{\text{label mixes as fake}}.
\end{aligned} \tag{5}$$

Note that the mixing consistency loss is simply the reconstruction between the mix and the re-encoding of it. This is necessary as the decoder may simply output an image which is not semantically consistent with the two images which were mixed. (Refer to Section 5.2 for more justification of why such a consistency loss is necessary.)

2.1 USING LABELS

While it is particular interesting to generate new examples via random mixing in the hidden states, we also explore a supervised formulation in which we learn a mixing function which can produce mixes between two examples consistent with a particular class label. We make this possible by backpropagating through a classifier network $p(\mathbf{y}|\mathbf{x})$ which branches off the end of the discriminator, i.e., an auxiliary classifier GAN (Odena et al., 2017).

Let us assume that for some image $\mathbf{x}$, we have a set of binary attributes associated with it $\mathbf{y}$ where $\mathbf{y} \in \{0,1\}^k$ (and $k \geq 1$). We introduce a mixing function $\text{Mix}_{\text{sup}}(\mathbf{h}_1, \mathbf{h}_2, \mathbf{y})$, which is an MLP that maps $\mathbf{y}$ to a mask $\mathbf{m} \in [0,1]^{bs \times f}$. This mask is used to produce a new convex combination $\tilde{\mathbf{h}}_{\text{mix}} = \mathbf{m}\mathbf{h}_1 + (1 - \mathbf{m})\mathbf{h}_2$, which is consistent with the class label $\mathbf{y}$. Note that the conditioning class label should be semantically meaningful with respect to both of the conditioned hidden states. For example, if we're producing mixes based on the gender attribute and both $\mathbf{h}_1$ and $\mathbf{h}_2$ are male, it would not make sense to condition on the 'female' label. To enforce this constraint, we simply make the conditioning label a convex combination $\tilde{\mathbf{y}}_{\text{mix}} = \alpha\mathbf{y}_1 + (1 - \alpha)\mathbf{y}_2$ as well, using $\alpha \sim \text{Uniform}(0, 1)$.

To make this more concrete, the autoencoder and discriminator, in addition to their losses described in Equation 5, try to minimise their respective losses:

$$\begin{aligned}
&\min_F \ \mathbb{E}_{\mathbf{x}_1,\mathbf{y}_2 \sim p(\mathbf{x},\mathbf{y}),\mathbf{x}_2,\mathbf{y}_2 \sim p(\mathbf{x},\mathbf{y}),\alpha \sim U(0,1)} \ \underbrace{\ell_{\text{GAN}}(D(g(\tilde{\mathbf{h}}_{\text{mix}})), 1)}_{\text{fool D with mix}} + \underbrace{\ell_{\text{cls}}(p(\mathbf{y}|\mathbf{g}(\tilde{\mathbf{h}}_{\text{mix}})), \tilde{\mathbf{y}}_{\text{mix}})}_{\text{make mix's class consistent}} \\
&\qquad\qquad + \underbrace{\beta ||\tilde{\mathbf{h}}_{\text{mix}} - f(g(\tilde{\mathbf{h}}_{\text{mix}}))||_2}_{\text{consistency loss}} \\
&\min_D \ \mathbb{E}_{\mathbf{x}_1,\mathbf{y}_2 \sim p(\mathbf{x},\mathbf{y}),\mathbf{x}_2,\mathbf{y}_2 \sim p(\mathbf{x},\mathbf{y}),\alpha \sim U(0,1)} \ \underbrace{\ell_{\text{GAN}}(D(g(\tilde{\mathbf{h}}_{\text{mix}})), 0)}_{\text{label mixes as fake}} \\
&\text{where } \tilde{\mathbf{h}}_{\text{mix}} = \text{Mix}_{\text{sup}}(f(\mathbf{x}_1), f(\mathbf{x}_2), \tilde{\mathbf{y}}_{\text{mix}}) \text{ and } \tilde{\mathbf{y}}_{\text{mix}} = \alpha\mathbf{y}_1 + (1 - \alpha)\mathbf{y}_2
\end{aligned} \tag{6}$$

While we do not explore this, having the mixer function Mix_{sup} internally produce a mask $\mathbf{m}$ may be interesting from a model explainability point of view – one could potentially leverage them to determine what feature maps contribute the most to explaining a particular set of attributes.

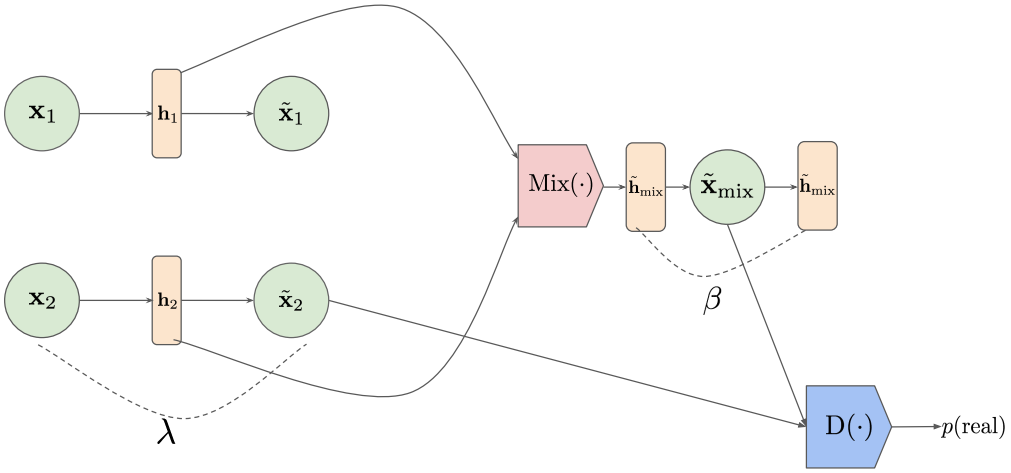


Figure 1: The mixing autoencoder. In addition to the autoencoder loss functions, we have a mixing function Mix which creates some combination between the latent variables $\mathbf{h}_1$ and $\mathbf{h}_2$, which is subsequently decoded into an image intended to be realistic-looking and semantically consistent with the two constituent images. This is achieved through the consistency loss (weighted by β) and the discriminator.

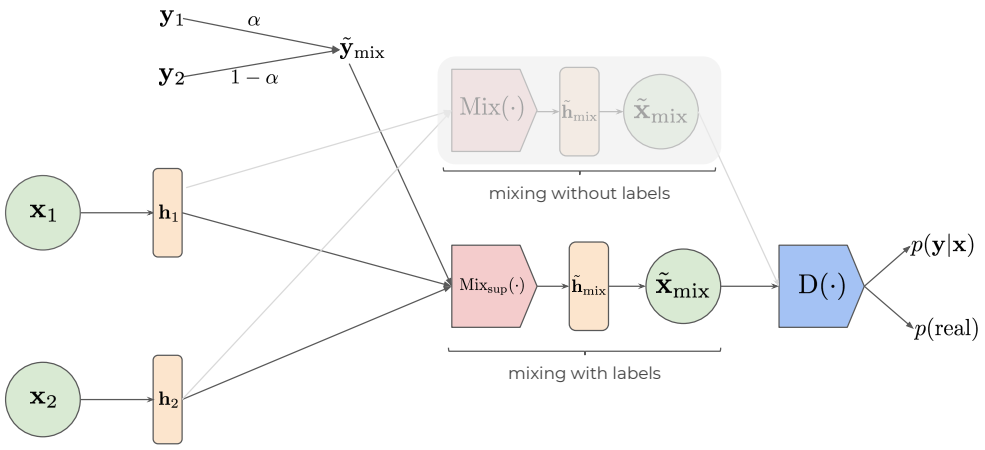


Figure 2: The supervised mixing autoencoder using an auxiliary classifier GAN (ACGAN). The mixer function, denoted in this figure as Mix_{sup}, takes $\mathbf{h}_1$, $\mathbf{h}_2$ and a convex combination of $\mathbf{y}_1$ and $\mathbf{y}_2$ (denoted $\tilde{\mathbf{y}}_{\text{mix}}$) and internally produces a Bernoulli mask which is then used to produce an output combination $\mathbf{h}_{\text{mix}} = \mathbf{m}\mathbf{h}_1 + (1-\mathbf{m})\mathbf{h}_2$. In addition to fooling the discriminator, it also has to make sure the class prediction by the auxiliary classifier is consistent with the mixed class $\tilde{\mathbf{y}}_{\text{mix}}$. Note that in this formulation, we still perform the kind of mixing which was shown in Figure 1, and this is shown in the diagram with the component noted 'mixing without labels'.

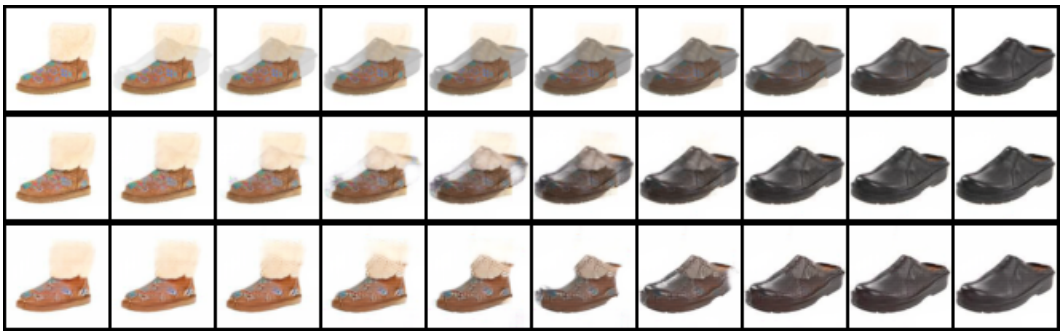


Figure 3: Interpolations between sheepskin boots and croc shoes for the Zappos dataset. For each image, from top to bottom, the rows denote: (a) linearly interpolating in pixel space; (b) the adversarial reconstruction autoencoder (ARAE); and (c) the adversarial mixup resynthesiser (AMR). For more images, consult the appendix section.



Figure 4: Interpolations between two images for the CelebA dataset, using the mixup technique (Equation 3). The left-most image is $\mathbf{x}_1$ and the right-most image is $\mathbf{x}_2$, and we create mixes between them by linearly increasing α from 0 to 1. For each image, from top to bottom, the rows denote: (a) linearly interpolating in pixel space; (b) the adversarial reconstruction autoencoder (ARAE); and (c) the adversarial mixup resynthesiser (AMR). For more images, consult the appendix section.

3 RESULTS

We use ResNets (He et al., 2015) for generator and discriminators. The precise architectures for generator and discriminator can be found here.[1] The datasets evaluated on are:

- UT Zappos50K (Yu & Grauman, 2014; 2017): a large dataset comprising 50k images of shoes, sandals, slippers, and boots. Each shoe is centered on a white background and in the same orientation, which makes it convenient for generative modelling purposes.
- CelebA (Liu et al., 2015): a large-scale and highly diverse face dataset consisting of 200K images. We use the aligned and cropped version of the dataset downscaled to 64px, and only consider (via the use of a keypoint-based heuristic) frontal faces. It is worth noting that despite this, there is still quite a bit of variation in terms of the size and position of the faces, which can make mixing between faces a more difficult task since the faces are not completely aligned.

As seen in Figure 4, both ARAE and the mixup variant produce more realistic-looking interpolations than in pixel space. Due to background details however, it is slightly harder to distinguish the quality between the two adversarial methods. Therefore, we also use the Frechet Inception Distance (FID)

[1] Link to the code will be added!



Figure 5: Interpolations between two images for the CelebA dataset, using the Bernoulli mixup technique (Equation 4). $\mathbf{m}$ is sampled from a Bernoulli(p), with zero denoting retain feature maps from $\mathbf{x}_1$ and one denoting retain feature maps from $\mathbf{x}_2$. (Top: ARAE, bottom: mixup ARAE)

by Heusel et al. (2017) (based on features extracted from a pre-trained CelebA classifier) to compute the distance between samples from the dataset and ones from our autoencoders[2]. Concretely, we compute (on the validation set) two scores: the FID between validation samples and their reconstructions, and the FID between validation samples and randomly sampled interpolations. For the latter, we repeat this five times (over different randomly sampled interpolations) and also over two repeat runs, resulting in $5 \times 3 = 15$ FID scores which are averaged over. These results are shown in Tables 1 and 2 for the mixup and Bernoulli mixup formulations, respectively. As we can see, mixes produced by our autoencoder have significantly lower scores, indicating they are more representative samples to the data distribution than that of the autoencoder baseline.

Table 1: Frechet Inception Distances computed between samples from the validation set and autoencoder for the mixup loss. λ and β denote respectively the reconstruction and consistency coefficients (see Equation 3), and were tuned to produce the best results.

Method	λ	β	FID(data, mixes)	FID(data, reconstruction)
ARAE	20	-	5.935591 $\pm$ 0.304821	1.385260 $\pm$ 0.133630
Mixup ARAE	100	50	**4.754007** $\pm$ **0.261206**	**1.374413** $\pm$ **0.111096**

Table 2: Frechet Inception Distances computed between samples from the validation set and autoencoder for the Bernoulli mixup technique. λ and β denote respectively the reconstruction and consistency coefficients (see Equation 4), and were tuned to produce the best results.

Method	λ	β	FID(data, mixes)	FID(data, reconstruction)
ARAE	20	-	16.643667 $\pm$ 0.588725	**1.385262** $\pm$ **0.133633**
Mixup ARAE	50	50	**10.360476** $\pm$ **0.842708**	3.728347 $\pm$ 0.606376

A more substantial exploration over hyperparameters (as well as existing techniques) will indeed be necessary, and we will defer this to the next version of this preprint.

3.0.1 SUPERVISED CELEBA

We present some qualitative results with the supervised formulation. We train our supervised mixing autoencoder variant using a subset of the attributes in CelebA ('is male', 'is wearing heavy makeup', and 'is wearing lipstick'). We consider pairs of examples $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2)\}$ (where one example is male and the other female) and produce random convex combinations of the attributes $\tilde{\mathbf{y}}_{\text{mix}} = \alpha\mathbf{y}_1 + (1-\alpha)\mathbf{y}_2$ and decode their resulting mixes $\text{Mix}_{\text{sup}}(f(\mathbf{x}_1), f(\mathbf{x}_2), \tilde{\mathbf{y}}_{\text{mix}})$. This can be seen in Figure 6.

[2]This is somewhat of a misnomer because the FID implies the use of an Inception network that has been pre-trained on ImageNet. Instead, we use a ResNet which has been trained on CelebA.



Figure 6: Interpolations produced by the class mixer function for the set of binary attributes {`male`, `heavy_makeup`, `lipstick`}. For each image, the left-most face is $\mathbf{x}_1$ and the right-most face $\mathbf{x}_2$, with faces in between consisting of mixes $\text{Mix}_{\text{sup}}(f(\mathbf{x}_1), f(\mathbf{x}_2), \tilde{\mathbf{y}}_{\text{mix}})$ of a particular attribute mix $\tilde{\mathbf{y}}_{\text{mix}}$, shown in the top-left corner of each face (where red denotes '0' and green denotes '1').

4 Conclusion

In this paper, we proposed the adversarial mixup resynthesiser and showed that it can be used to produce realistic-looking combinations of examples by performing mixing in the bottleneck of an autoencoder. Furthermore, we presented a semi-supervised variant in which one can leverage class labels to produce more specific kinds of mixes. While one of our motivations behind this technique was to essentially leverage an autoencoder as a generative model, we conjecture that our technique may have positive implications on downstream tasks and the resulting representation, though this is yet to be substantiated. Furthermore, since we have proposed a few different mixing functions, it will be useful to examine more fundamentally what their implications are on the resulting representation that is learned. As such, we would like to run further experiments in these issues, and also run comparisons against other similar and existing techniques in the literature.

Acknowledgments

We thank Huawei for supporting this research. We also thank Compute Canada for GPUs used in this work. Vikas Verma was supported by Academy of Finland project 13312683 / Raiko Tapani AT kulut.

REFERENCES

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 531–540, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/belghazi18a.html.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver (eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pp. 17–36, Bellevue, Washington, USA, 02 Jul 2012. PMLR. URL http://proceedings.mlr.press/v27/bengio12a.html.

David Berthelot, Colin Raffel, Aurko Roy, and Ian J. Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *CoRR*, abs/1807.07543, 2018. URL http://arxiv.org/abs/1807.07543.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016. URL http://arxiv.org/abs/1606.03657.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL http://arxiv.org/abs/1706.08500.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *CoRR*, abs/1810.02334, 2018. URL http://arxiv.org/abs/1810.02334.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. URL http://arxiv.org/abs/1703.00848.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015. URL http://arxiv.org/abs/1511.05644.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. URL http://arxiv.org/abs/1802.05957.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2642–2651. JMLR. org, 2017.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 833–840, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104587.

Tim Sainburg, Marvin Thielk, Brad Theilman, Benjamin Migliori, and Timothy Gentner. Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions. *CoRR*, abs/1807.06650, 2018. URL http://arxiv.org/abs/1807.06650.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *CoRR*, abs/1708.01289, 2017. URL http://arxiv.org/abs/1708.01289.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017. URL http://arxiv.org/abs/1711.00937.

Vikas Verma, Alex Lamb, Christopher Beckham, Aaron Courville, Ioannis Mitliagkis, and Yoshua Bengio. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv preprint arXiv:1806.05236*, 2018.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1096–1103, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.

A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *Computer Vision and Pattern Recognition (CVPR)*, Jun 2014.

A. Yu and K. Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *International Conference on Computer Vision (ICCV)*, Oct 2017.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL http://arxiv.org/abs/1710.09412.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *CoRR*, abs/1611.09842, 2016. URL http://arxiv.org/abs/1611.09842.

5 APPENDIX

5.1 EXPERIMENTAL CONFIGURATION

We will provide a summary of our experimental setup here, though we also provide links to (and encourage viewers to look at) various parts of the code such as the networks used for the generator and discriminator and the optimiser hyperparameters.

We use a residual network for both the generator and discriminator. The discriminator uses spectral normalisation (Miyato et al., 2018), with five discriminator updates being performed for each generator update. We use ADAM for our optimiser with $\alpha = 2e^{-4}$, $\beta_1 = 0.0$ and $\beta_2 = 0.99$.

5.2 CONSISTENCY LOSS

Earlier in the paper we mentioned that it was necessary to include a consistency loss (see Equation 5 and Figure 1) since we want the decoded mix to be semantically consistent with its constituent latent variables. That is to say, if we have some mixed latent code $\tilde{\mathbf{h}}$ and decode it into $\tilde{\mathbf{x}}$, we would like that its re-encoding $g(\tilde{\mathbf{x}}) = \tilde{\mathbf{h}}$. In order to show that this loss is indeed necessary, we plot its values during training for both the ARAE baseline and the mixup ARAE. We can see that for the baselines (shown in solid colour) the consistency loss does not naturally go down to zero, hence the need to include this in our loss function.

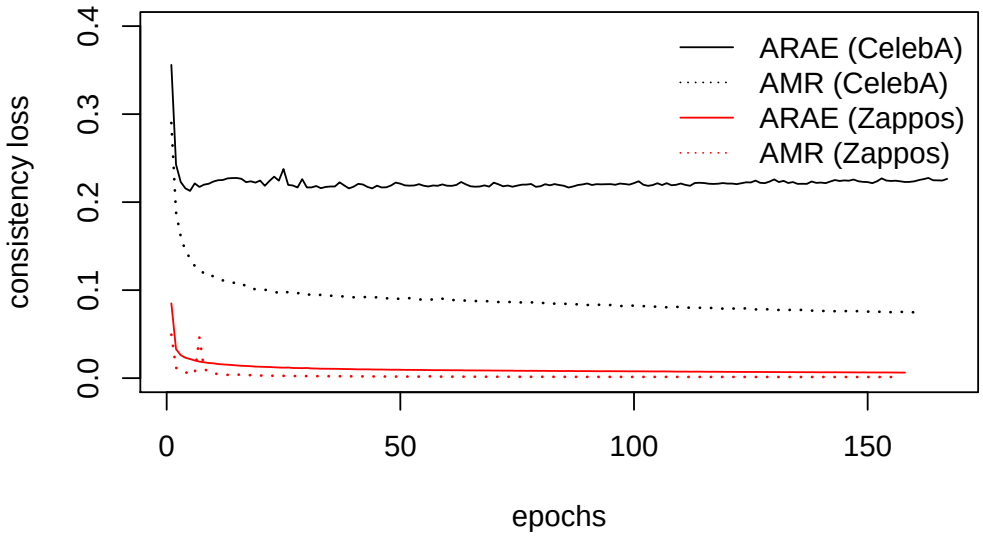


Figure 7: Consistency loss

5.3 ZAPPOS (MIXUP)



Figure 8: Interpolations between two images. For each image, from top to bottom, the rows denote: (a) linearly interpolating in pixel space; (b) the adversarial reconstruction autoencoder (ARAE); and (c) the adversarial mixup resynthesiser (AMR). For more images, consult the appendix section.

5.4 CELEBA (MIXUP)

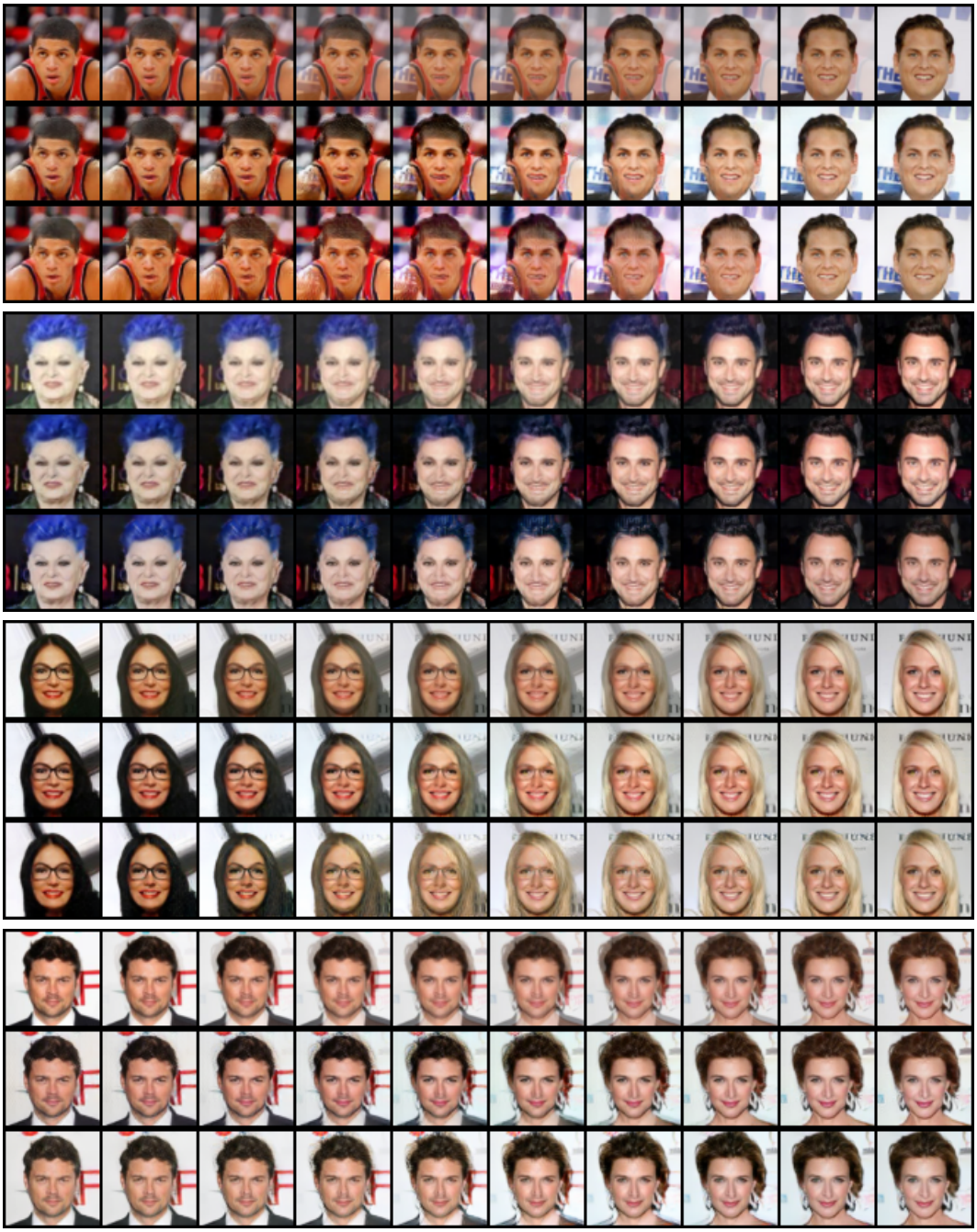


Figure 9: Interpolations between two images using the mixup technique (Equation 3). For each image, from top to bottom, the rows denote: (a) linearly interpolating in pixel space; (b) the adversarial reconstruction autoencoder (ARAE); and (c) the adversarial mixup resynthesiser (AMR). For more images, consult the appendix section.

5.5 CELEBA (BERNOULLI MIXUP)



Figure 10: Interpolations between two images using the Bernoulli mixup technique (Equation 4). (For each image, top: ARAE, bottom: AMR)