Deep Image Representations with Explainable Features

Vasily Tolkachev

ZHAW IDP

vasily.tolkachev@gmail.com

www.github.com/vastol

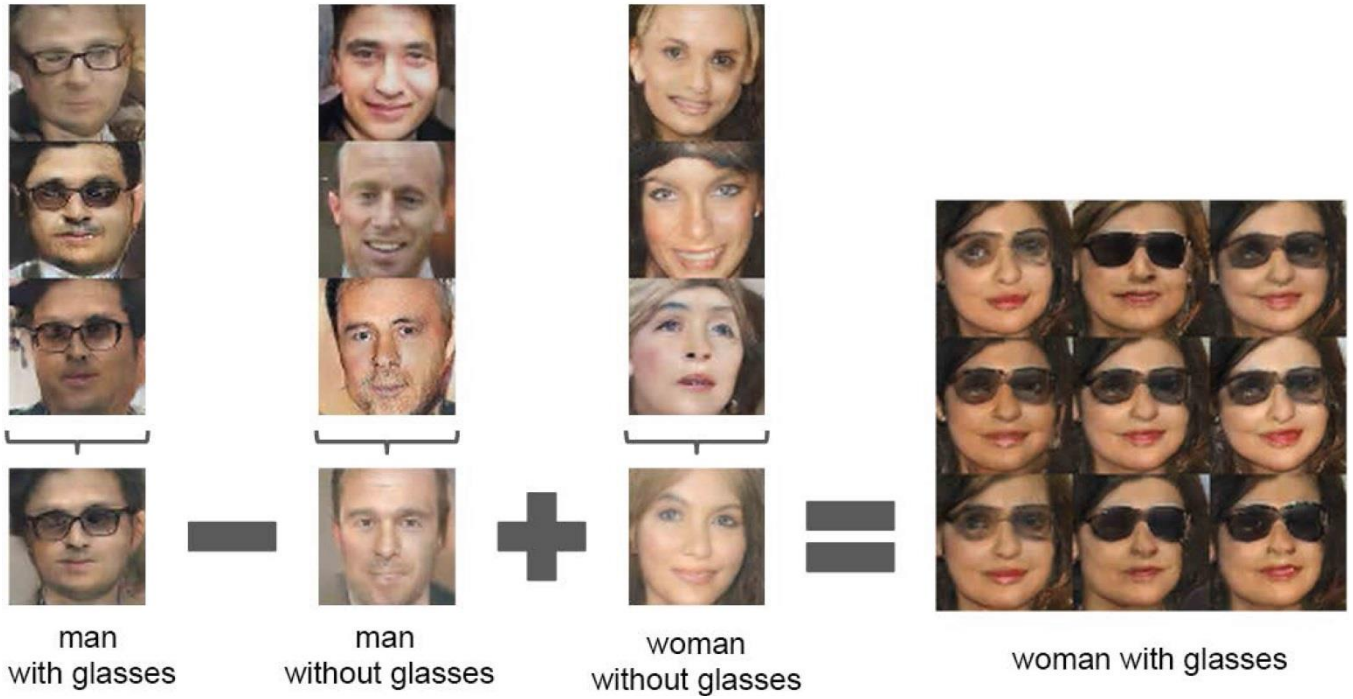
www.idp.zhaw.ch

ZHAW Datalab Seminar

19.06.2018

Motivation

- In the context of autoencoders, do we need features in the bottleneck layer (representations) to be explanable? In general, is it needed for efficient classification/clustering/transfer learning?
- How do we make a network learn explainable features?
- How do we avoid cases when just one feature group is used to reconstruct all images, while other features are not fully exploited?
- Can we do image arithmetic with explainable features?



Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016

Disentangling Factors of Variation by Mixing Them

Paper review

Qiyang Hu, Attila Szabó, Tiziano Portenier, Matthias Zwicker, Paolo Favaro

Overview

- Goal: separate image features into semantically interpretable properties (factors of variation). In case of face recognition these can be hair style, color, glasses, smile etc.

- Data for evaluation: MNIST, Sprites (game avatars), CelebA (celebrities)
- Usage:
  - transfer attributes from one image to another (man without glasses $\rightarrow$ man with glasses)
  - image retrieval/search and classification using the feature space

- Feature representation is considered disentangled if sufficiently accurate classification can be achieved by simple linear classifier
- Novel invariance and classification loss types

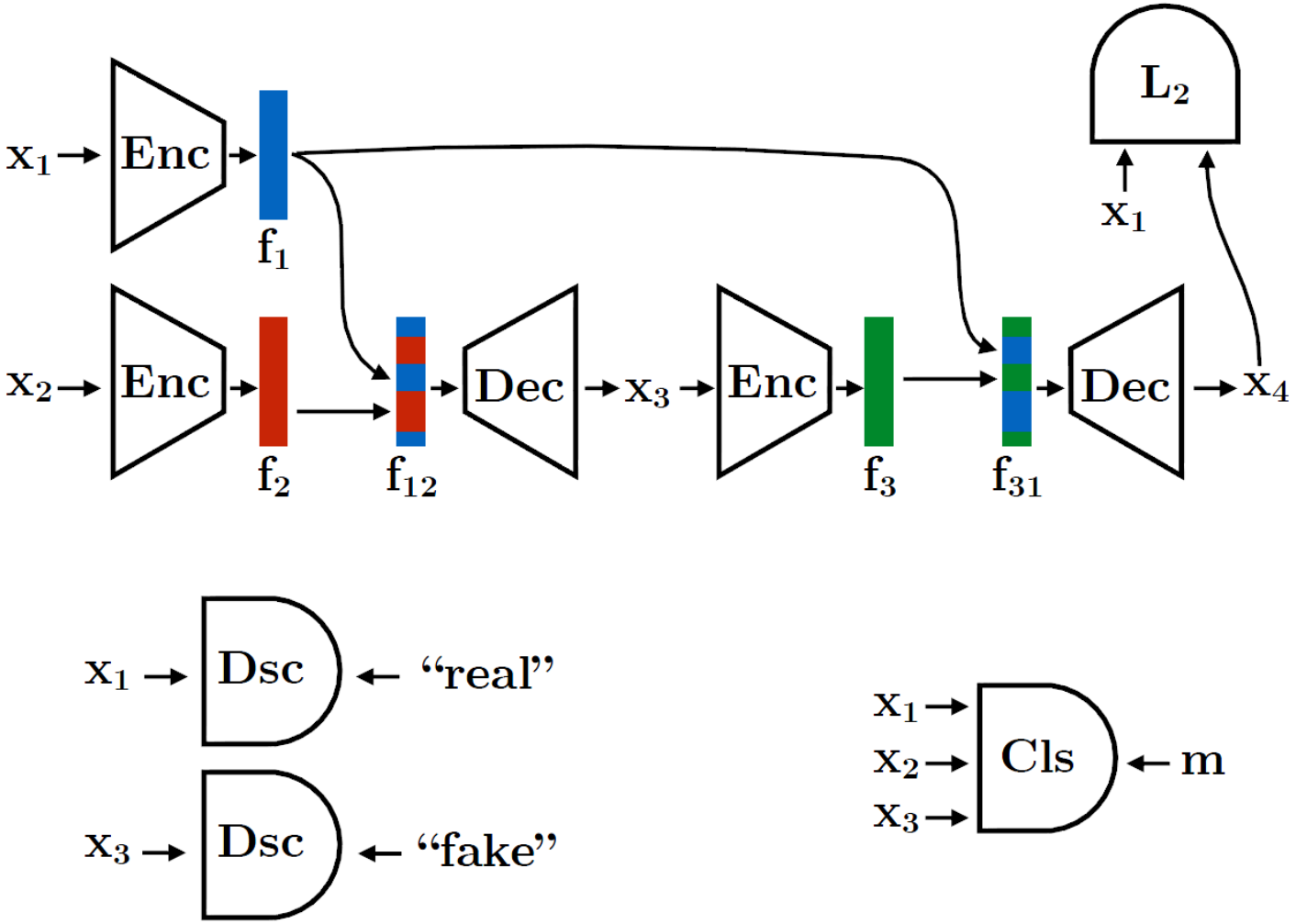
Details

- Assumption: each factor of variation is encoded using its own feature vector, which is called a feature chunk

- Invariance property:
  - encoding of each image attribute into its feature chunk should be invariant to transformations of any other image property.
  - decoding of each chunk into its corresponding attribute should be invariant to changes of other chunks.
- Invariance is achieved by a sequence of two mixing and unmixing autoencoders.

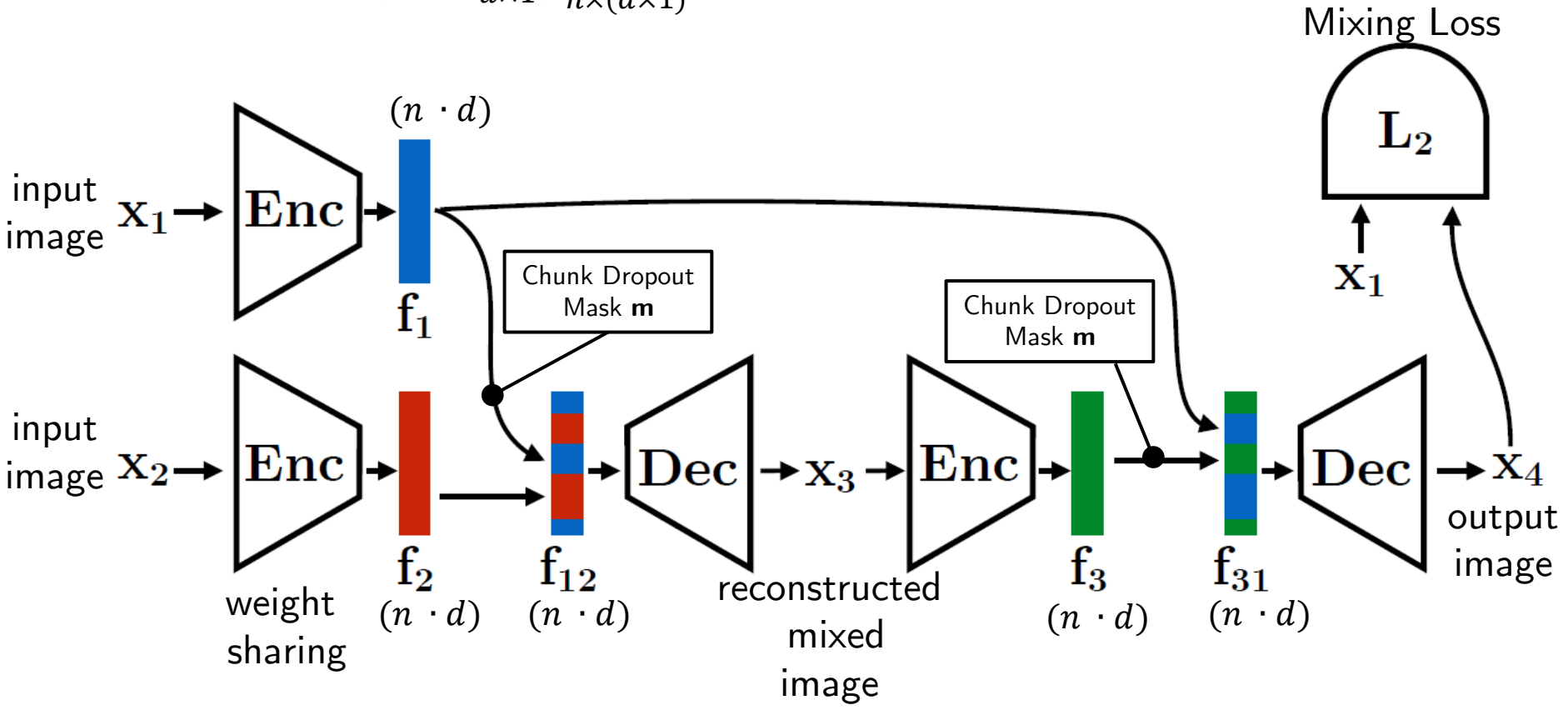
- Need to avoid the Shortcut problem, when an encoder utilizes just one feature chunk to reconstruct all images, not providing the meaningful feature decomposition.

Network Architecture Overview



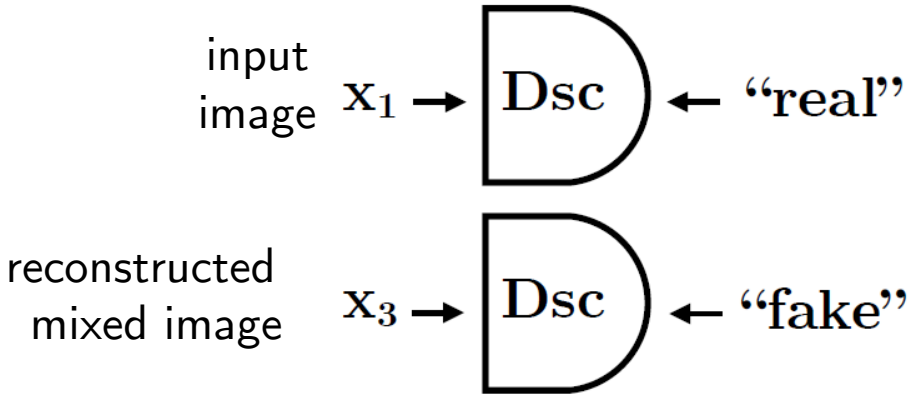
1. Sequence of Autoencoders

feature vector $f_j = \begin{pmatrix} [f_j^1]_{d\times 1} \\ \cdots \\ [f_j^n]_{d\times 1} \end{pmatrix}_{n\times(d\times 1)}$ where $[f_j^i]_{d\times 1}$ is a i^{th} chunk of feature vector j



Mixing Loss $\mathcal{L}_M(\theta_{\text{Enc}}, \theta_{\text{Dec}}) = E_{\mathbf{x}_1,\mathbf{x}_2}\left[\sum_{\mathbf{m}} |\mathbf{x}_4 - \mathbf{x}_1|^2\right]$

2. Adversarial (Discriminator) Loss

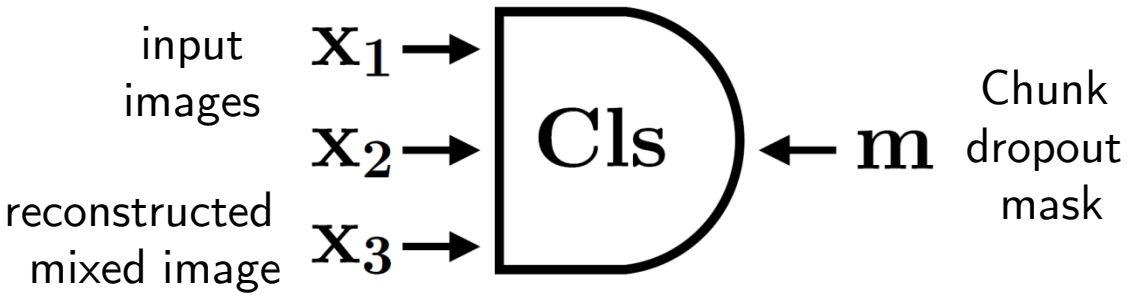


$$\mathcal{L}_G(\theta_{\text{Enc}}, \theta_{\text{Dec}}, \theta_{\text{Dsc}}) = \sum_{\mathbf{m}} E_{\mathbf{x}_1, \mathbf{x}_2}\Big[\log(\text{Dsc}(\mathbf{x}_1)) + \log(1 - \text{Dsc}(\mathbf{x}_3))\Big]$$

- When the GAN objective reaches the global optimum, the distribution of 'fake' images should match the real image distribution.

- Hence, the adversarial loss is used to ensure that the mixed image $\mathbf{x_3}$, which is reconstructed by the first autoencoder, comes from the same distribution as the original input image $\mathbf{x_1}$

3. Classifier (cross-entropy) Loss



$$\mathcal{L}_C(\theta_{\text{Enc}}, \theta_{\text{Dec}}, \theta_{\text{Cls}}) =$$
$$E_{\mathbf{x}_1,\mathbf{x}_2}\Big[-\sum_{\mathbf{m}}\sum_i m^i \log(y^i) + (1-m^i)\log(y^i))\Big]$$

- A binary classifier takes input images $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$ and for every feature chunk i decides if $\mathbf{x_3}$ was generated using the corresponding chunk from $\mathbf{x_1}$ or $\mathbf{x_2}$.
- Combining the classifier and the chunk dropout mask $\mathbf{m}$ avoid the shortcut problem

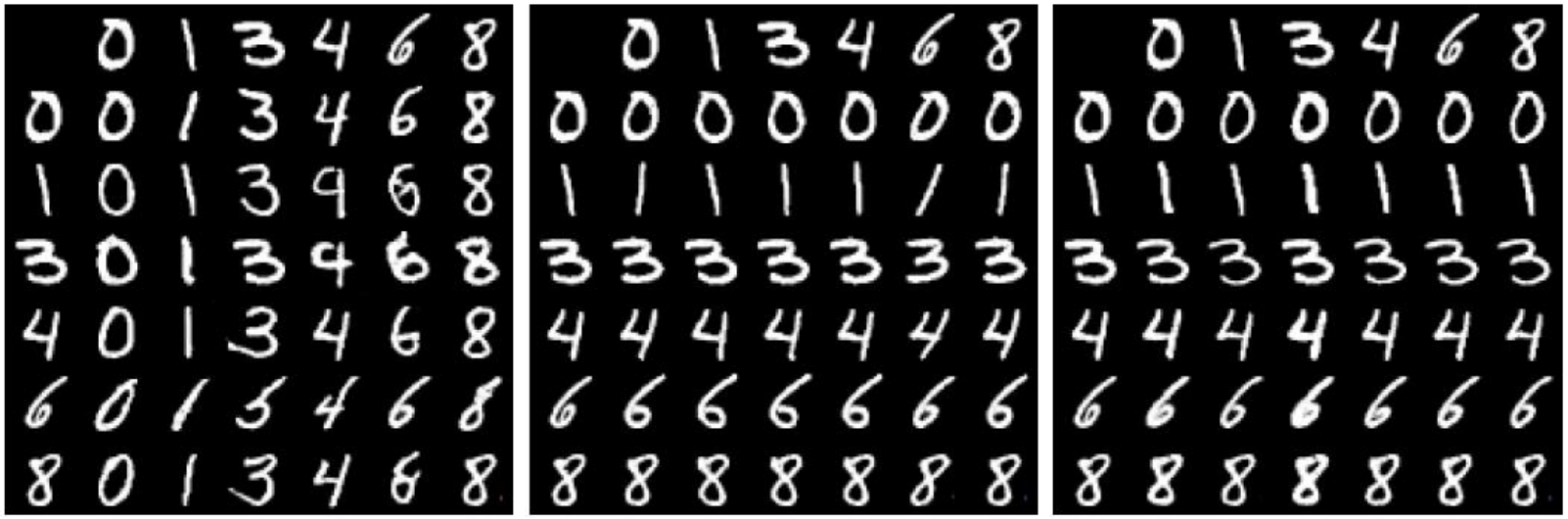
Overall Loss

$$\min_{\theta_{\text{Enc}},\theta_{\text{Dec}},\theta_{\text{Cls}}} \max_{\theta_{\text{Dsc}}} \quad \lambda_M \mathcal{L}_M + \lambda_G \mathcal{L}_G + \lambda_C \mathcal{L}_C$$

Experiments

- DCGAN was used for encoder, decoder and discriminator
- AlexNet with batchnorm without dropout was used as the classifier
- The last fully connected layer of the encoder was taken as a feature vector, then manually split into chunks.
- For evaluation on MNIST, 8 chunks were used
- For Sprites and CelebA, 64 chunks were used (otherwise lower rendering quality)
- For CelebA the mixing loss had a greater weight, possibly to achieve better rendering due to a semantically richer dataset

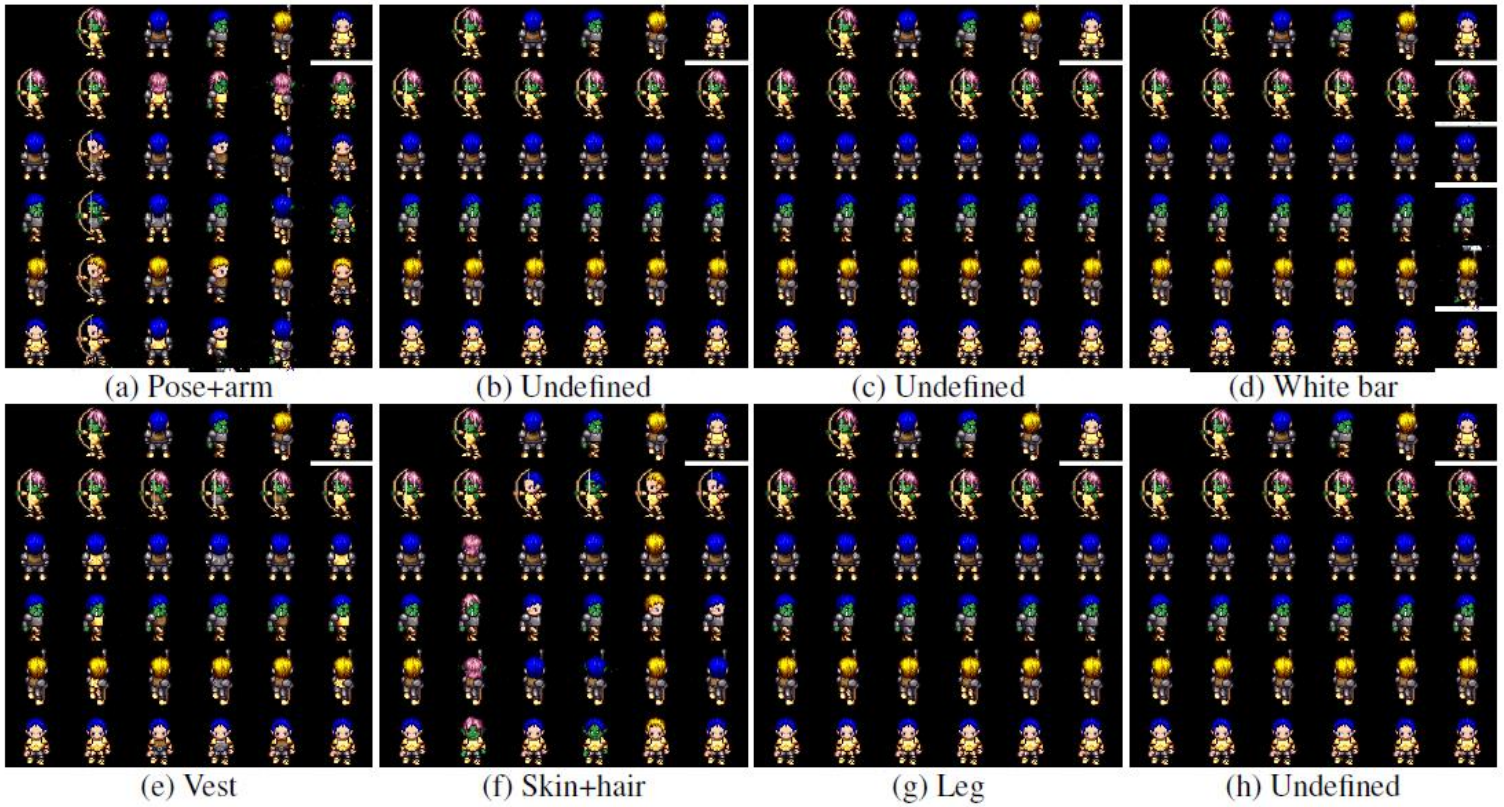
MNIST (60K images)



(a) Digit class (b) Rotation angle (c) Stroke width

- The method was able to disentangle the labels and non-labelled attributes, like rotation angle and stroke width (assigned by manual inspection)
- All recognizable variations seem to have been encoded in the three chunks

Sprites (120K images)



- Many body parts labels available (body shape, skin color hairstyle, etc.)
- Mixing autoencoder was able to disentangle 2 chunks, while adding just the GAN loss improved rendering.

- The full loss is illustrated to improve performance, eliminates artefacts and solves the shortcut problem

Sprites (120K images)

- Nearest neighbor classification was done on a chunk of the features and mean average precision(mAP) was used to compare it with the true labels.

- Comparison to Autoencoder and other restricted versions of the model shows a significant improvement in mAP:

Method	body	skin	vest	hair	arm	leg	pose	average
Random	0.5	0.25	0.33	0.17	0.5	0.5	0.006	0.32
C+G	0.53	0.31	0.41	0.24	0.51	0.52	0.06	0.37
AE	0.56	0.37	0.40	0.31	0.54	0.56	0.46	0.46
AE+C+G	0.59	0.50	0.53	0.46	0.56	0.54	0.44	0.52
MIX	0.57	0.61	0.51	0.62	0.54	0.94	**0.53**	0.62
MIX + C	0.57	0.65	0.43	**0.63**	0.55	0.58	0.51	0.56
MIX + G	**0.59**	0.31	0.44	0.24	0.54	0.96	0.47	0.51
MIX + C + G	0.58	**0.80**	**0.94**	0.49	**0.58**	**0.96**	0.52	**0.70**

CelebA (200K images)

CelebA (200K images)

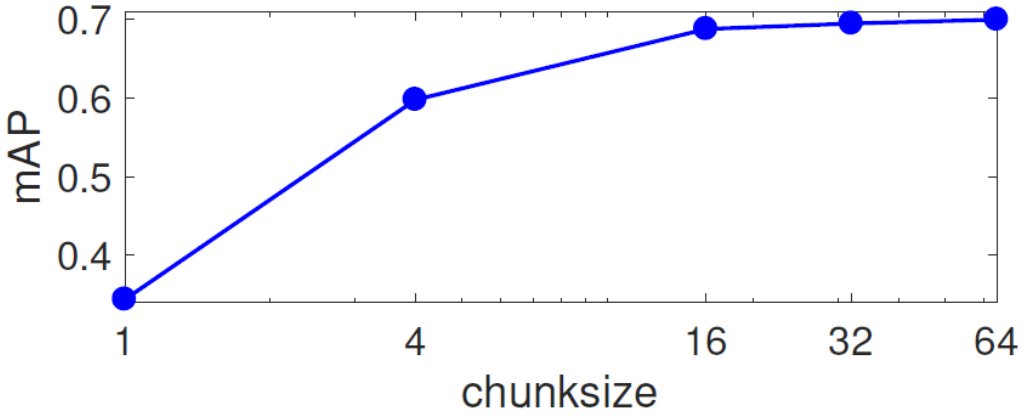
- 40 labeled binary attributes (gender, hair color, facial hair, etc.)
- DCGAN showed a more pronounced attribute transfer, while BEGAN blurred out the changes
- The method recovered 5 semantically meaningful attributes: brightness, glasses, hair color, hair style and pose/style.
- Since a class depends only on one chunk in the disentangled representation, a linear classification in the whole disentangled (chunked) feature space was evaluated. The results were slightly worse, but comparable with the latest architectures DIP-VAE and beta-VAE.

Advantages

- No manual labeling required
- No need to isolate factors of variation beforehand or sample images where only one factor changes
- Novel idea of classification into feature chunks
- Shortcut problem is solved with a classification loss forcing each feature chunk to have a discernable effect
- Feature chucks can be high-dimensional in contrast with other papers
- In the disentangled feature space:
  - linear classifier should yield high precision and recall
  - Nearest neighbor search could successfully be used for image retrieval

Limitations

- How to choose the number of chunks(n) and their size? Not enough heuristics/experiments/justification.



- To make feature chunks 'meaningful', each chunk was manually assigned to a class (subjective!), making the procedure not completely unsupervised.
- Needs further evaluation on more semantically rich datasets (medicine, self-driving cars).
- Feature space is only designed for attribute transfer and can't be used for sampling.
- What about datasets with artefacts (errors in the classes, strongly unbalanced classes)?

Limitations

- More generally: is feature decomposition into chunks needed for precise and efficient transfer learning?

- Because of the manual interpretation of chunks, the same argument as SIFT/SURF features vs. end-to-end neural networks.

Thank you
for your attention!