

A APPENDIX

A.1 DATASETS

For the bulk of our experiments in Sections 5 and 4 we use the MNIST (LeCun et al., 1998), EMNIST (Cohen et al., 2017) and CelebFaces Attributes (CelebA) (Liu et al., 2015) datasets. Few examples from each dataset and corresponding concepts are presented in Figure 5

MNIST. MNIST is a handwritten digits classification dataset with 10 digits. There are 60,000 training examples and 10,000 testing examples. It is a subset of a larger set from National Institute of Standards and Technology (NIST) where each image is size-normalized and centered in a fixed-size image. We introduce concept of *rotation* to these images by rotating 75% (unless otherwise specified) of images by one of 4 possible values $\{90, 180, 270\}$. Concept of *color* is added to images by multiplying each image channel with corresponding [R,G,B] values drawn from a uniform distribution.

EMNIST. EMNIST is a set of handwritten characters derived from NIST Special Database 19. For our experiments we use the EMNIST Letters dataset, which is a 26 class classification of english letters. There are 88,800 training examples and 14,800 testing examples. We introduce concept of *rotation* to these images by rotating 75% (unless otherwise specified) of images by one of 4 possible values $\{90, 180, 270\}$. Concept of *color* is added to images by multiplying each image channel with corresponding [R,G,B] values drawn from a uniform distribution.

CelebFaces Attributes. CelebA is a large-scale face attributes prediction dataset with more than 200,000 images. Each image has 40 different attribute annotations. There are 162,770 training examples, 19962 test examples and 19867 validation examples. For our experiments, in Sections 5 we use concepts of $\{Eyeglasses, No_Beard, Smiling\}$ and for experiments in 4 we use $\{Smiling, Wearing_Lipstick, Heavy_Makeup, High_Cheekbones\}$.

A.2 MODELS

To describe the architecture used in our experiments, we use the following notation:

- $\text{Conv2d}(c_{in}, c_{out}, k, s, p)$: A two dimensional convolution operation that takes c_{in} input channels and produce c_{out} output channels. A square kernel of size k is used. s is the stride and p is the padding.
- $\text{ConvT2d}(c_{in}, c_{out}, k, s, p)$: A two dimensional transposed convolution operation that takes c_{in} input channels and produce c_{out} output channels. A square kernel of size k is used. s is the stride and p is the padding.
- $\text{Linear}(c_{in}, c_{out})$: A linear layer that maps an input vector $v_1 \in \mathbb{R}^{c_{in}}$ to an output vector $v_2 \in \mathbb{R}^{c_{out}}$.

Autoencoder Architecture

The architecture for our Autoencoder experiment presented in 5 is based on Lucic et al. (2018). Similar to Esser et al. (2020), we replace the batch normalization Ioffe & Szegedy (2015) by activation normalization Kingma & Dhariwal (2018). Details regarding the architecture can be found in Table 3.

Table 3: Architecture of autoencoder model. Input images have size of $h \times w \times c$ and are quadratic in nature $h = w$

Encoder	Decoder
Conv2d(3, 64, 4, 2, 1), ActNorm, LeakyReLU(0.2)	ConvT2d(z , 512, $h/16$, 1, 0), ActNorm, LeakyReLU(0.2)
Conv2d(64, 128, 4, 2, 1), ActNorm, LeakyReLU(0.2)	ConvT2d(512, 256, 4, 2, 1), ActNorm, LeakyReLU(0.2)
Conv2d(128, 256, 4, 2, 1), ActNorm, LeakyReLU(0.2)	ConvT2d(256, 128, 4, 2, 1), ActNorm, LeakyReLU(0.2)
Conv2d(256, 512, 4, 2, 1), ActNorm, LeakyReLU(0.2)	ConvT2d(128, 64, 4, 2, 1), ActNorm, LeakyReLU(0.2)
Conv2d(512, $2 \cdot z$, $h/16$, 1, 0)	ConvT2d(64, 3, 4, 2, 1), Tanh

Classifier Architecture



Figure 5: Example images for rotated-colored-MNIST, rotated-colored-EMNIST and CelebFaces Attributes.

We use two different architectures for our transfer learning setup. A larger 6 layer network for source model which is the same as the encoder block of autoencoder presented in Table 3 with a fully connected layer as classifier attached at the end (see Table 4). A smaller 3 layer network for target model (see Table 5).

Table 4: Architecture of source model

Encoder	Classification Head
Conv2d(3, 64, 4, 2, 1), ActNorm, LeakyReLU(0.2)	Linear(z^s , $n_{classes}$)
Conv2d(64, 128, 4, 2, 1), ActNorm, LeakyReLU(0.2)	
Conv2d(128, 256, 4, 2, 1), ActNorm, LeakyReLU(0.2)	
Conv2d(256, 512, 4, 2, 1), ActNorm, LeakyReLU(0.2)	
Conv2d(512, z^s , $h/16$, 1, 0)	

Table 5: Architecture of source model

Encoder	Classification Head
Conv2d(3, 64, 4, 2, 1), ActNorm, LeakyReLU(0.2)	Linear(z^t , $n_{classes}$)
Conv2d(64, z^t , $h/16$, 1, 0)	

Invertible Neural Network

As described in Section 3.1, we use the Invertible Neural Network proposed in (Esser et al., 2020) - Invertible Interpretable Network. In this implementation, the network consists of three invertible

layers stacked one on to of another to create an invertible *block*: coupling blocks (Dinh et al., 2016), actnorm (Kingma & Dhariwal, 2018) blocks and shuffling blocks. After the input representation z^s is passed through several invertible blocks, the output is split into K factors $(z_k^s)_{k=0}^K$. We refer readers to (Esser et al., 2020) for further details.

Algorithm for training IIN

Algorithm 2 TRAIN-IIN: Train Invertible Interpretable Network

```

1: Inputs: IIN training dataset  $D_I$ ; IIN loss  $\mathcal{L}_{\mathcal{I}}(\cdot)$ ; Seed weight parameters:  $\mathcal{W}_I[0]$ ; Source pre-
   trained network  $f^s$ ; Number of Epochs  $E$ , Layer  $L$ , Concepts  $K$  to factorize.
2: Randomly shuffle  $D_I$ .
3: for  $epoch \in [1 : E]$  do
4:   for  $batch \in D_I$  do
5:      $(x_a, x_b | concept) \leftarrow D_I[batch]$ . Where  $(x_a, x_b)$  are pairs of samples encoding concept.
6:      $(z_a^s, z_b^s) \leftarrow (f_L^s(x_a), f_L^s(x_b))$ .
7:      $\mathcal{W}_I[batch] \leftarrow \mathcal{W}_I[batch - 1] - \eta_{batch} \nabla_{\mathcal{W}_I} \mathcal{L}_{\mathcal{I}}((z_a^s, z_b^s) | concept)$ 
8:   end for
9: end for
10: Output: Trained model  $I$  with last iterate of  $\mathcal{W}_I$ 

```

Statistics Network

For mutual information based experiments in Section 5, we adapt a statistics network proposed in Belghazi et al. (2018). We use a custom sequence of 4 layer network that takes intermediate representation Z and concept vector C to estimate the mutual information between them. Details about the architecture is presented in Table 6.

Table 6: Architecture of statistics network for mutual information estimation.

Statistics Network
ConcatLayer(Z, C)
Linear($ Z + C , 100$), ReLU()
Linear(100, 100), ReLU()
Linear(100, 1), ReLU()

A.3 EXPERIMENTAL DETAILS

For our experimental analysis in the main paper, we set the number of epochs for training to $E = 50$ for all models. We train all models using a batch size of 25 and a learning rate of 10^{-4} for the Adam optimizer (Kingma & Ba, 2014). All models were randomly initialized before training. While training source classifier, IIN, target classifier, we use different samples to ensure that each model is trained with non-overlapping samples. For example, we train the source network in CelebA experiments with the standard training dataset, IIN and target classifier with a split of validation samples. Testing is done on the standard test dataset. Statistics network is trained by querying pre-trained source model with standard validation dataset and testing on standard test dataset. For creating prototypes for concepts, we use 100 samples.

The models were trained in parallel with the specifications shown in Table 7.

Resource	Setting
CPU	IBM Power 9 CPU @ 3.15GHz
Memory	512GB
GPUs	1 x NVIDIA Tesla V100 16 GB
Disk	1.2 TB
OS	RedHat8

Table 7: Resources used for training



Figure 6: Visualization of *color* concept blocking using prototypes created by different aggregation operators in colored-MNIST images. We find that *median* and *mean* performs the best.



Figure 7: Visualization of *Eyeglasses* concept blocking using prototypes created by different aggregation operators in CelebA images. We find that *median* and *mean* performs the best.

A.4 ADDITIONAL EXPERIMENTS

Choosing the operation for creating prototypes

In order to create prototypes for each concept, we first query the pre-trained source model and IIN with 100 images that don't have a concept, and aggregate the corresponding concept factor in IIN output. To choose the right aggregation factors, we experimented with simple operations such as *mean*, *median*, *mode* and setting the factor to *zero*. In order to visualize how effective each aggregation is at blocking a concept, we visualize a few images reconstructed by using auto-encoder based source network as detailed in Section 5.1. We present these results for colored-MNIST images with *color* concept blocked in 6 and CelebA images with *Eyeglasses* concept blocked in 7. We find the *mean* and *median* performs the best and we chose *mean* for rest of the experiments presented in Sections 5 and 4.

Autoencoder based concept removal images

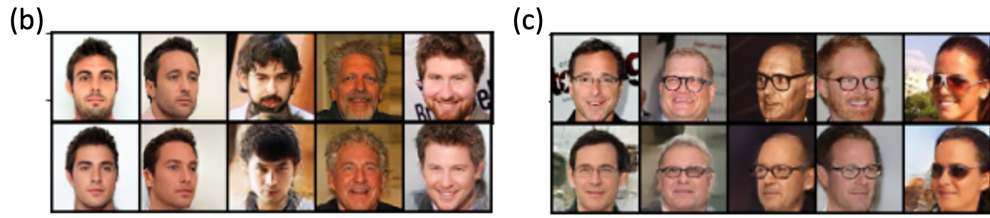


Figure 8: Visualization of concept blocking using an autoencoder as source network. For CelebA dataset, we consider randomly drawn samples with *Beard & Smiling* attributes and proceed to block them individually. These results are presented in (a) and (b) respectively