A Appendix

A.1 Additional definition and proofs

Let us first recall the optimal transport problem associated with the minimization of $\mathcal{L}^{hKR}_{\lambda,m}$:

$$\inf_{f\in Lip_1(\Omega)} \mathcal{L}^{hKR}_{\lambda(f),m} = \inf_{\pi\in\Pi^p_\lambda(\mu,\nu)} \int_{\Omega\times\Omega} |\boldsymbol{x}-\boldsymbol{z}|d\pi + \pi_{\boldsymbol{x}}(\Omega) + \pi_{\boldsymbol{z}}(\Omega) - 1 \tag{3}$$

Where $\Pi^p_\lambda(\mu,\nu)$ is the set consisting of positive measures $\pi \in \mathcal{M}_+(\Omega\times\Omega)$ which are absolutely continuous with respect to the joint measure $d\mu \times d\nu$ and $\frac{d\pi_{\mathbf{x}}}{d\mu} \in [p, p(m+\lambda)]$, $\frac{d\pi_{\mathbf{z}}}{d\nu} \in [1-p, (1-p)(m+\lambda)]$. We name π^* the optimal transport plan according to Eq.3 and and f^* the associated potential function.

Proof of proposition 1: According to [53], we have

$$||\nabla_x f^*(\boldsymbol{x})|| = 1$$

almost surely and

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\pi^*}\left(|f^*(\boldsymbol{x}) - f^*(y)| = ||\boldsymbol{x}-y||\right) = 1$$

Following the proof of proposition 1 in [26] and [3] we have :
Given $\boldsymbol{x}_\alpha = \alpha * x + (1-\alpha)y$, $0 \leq \alpha \leq 1$

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\pi^*}\left(\nabla_x f^*(\boldsymbol{x}_\alpha) = \frac{\boldsymbol{x}_\alpha - y}{||\boldsymbol{x}_\alpha - y||}\right) = 1.$$

So for for $\alpha = 1$ whe have

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\pi^*}\left(\nabla_x f^*(\boldsymbol{x}) = \frac{\boldsymbol{x} - y}{||\boldsymbol{x} - y||}\right) = 1$$

and then

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\pi^*}\left(y = \boldsymbol{x} - \nabla_x f^*(\boldsymbol{x}).||\boldsymbol{x}-y||\right) = 1$$

This prove the proposition 1 by choosing $t = ||\boldsymbol{x}-y||$.■

Proof of proposition 2: Let μ and ν two distributions with disjoint support with minimal distance ϵ and f^* an optimal solution minimizing the $\mathcal{L}^{hKR}_{\lambda,m}$ with $m < 2\epsilon$. According to [53], f^* is 100% accurate. Since the classification is based on the sign of f we have : $\forall \boldsymbol{x} \in \mu, f^*(\boldsymbol{x}) \geq 0$ and $\forall y \in \nu, f^*(y) \leq 0$. Given $\boldsymbol{x} \in \mu$ and $y = tr_\pi(\boldsymbol{x}) = \boldsymbol{x} - t\nabla_x f^*(\boldsymbol{x})$ and $y \in \nu$. According to the previous proposition we have :

$$\begin{aligned}
|f^*(\boldsymbol{x}) - f^*(y)| &= ||\boldsymbol{x} - y|| \\
|f^*(\boldsymbol{x}) - f^*(y)| &= ||\boldsymbol{x} - (\boldsymbol{x} - t\nabla_x f^*(\boldsymbol{x}))|| \\
|f^*(\boldsymbol{x}) - f^*(y)| &= ||t\nabla_x f^*(\boldsymbol{x}))|| \\
|f^*(\boldsymbol{x}) - f^*(y)| &= t.||\nabla_x f^*(\boldsymbol{x}))|| && (t \geq 0) \\
|f^*(\boldsymbol{x}) - f^*(y)| &= t && (\nabla_x f^*(\boldsymbol{x}) = 1) \\
f^*(\boldsymbol{x}) - f^*(y) &= t && (f^*(\boldsymbol{x}) \geq 0, f^*(y) \leq 0) \\
f^*(y) &= f^*(\boldsymbol{x}) - t
\end{aligned}$$

since $f^*(y) \leq 0$ we obtain :

$$f^*(\boldsymbol{x}) \leq t$$

Since f^* is continuous, $\exists t' > 0$ such that $\boldsymbol{x}_\delta = \boldsymbol{x} - t'\nabla_x f^*(\boldsymbol{x})$ and $f^*(\boldsymbol{x}_\delta) = 0$. We have :

$$\begin{aligned}
|f^*(\boldsymbol{x}) - f^*(\boldsymbol{x}_\delta|) &\leq ||\boldsymbol{x} - \boldsymbol{x}_\delta|| \\
f^*(\boldsymbol{x}) &\leq ||\boldsymbol{x} - (\boldsymbol{x} - t'\nabla_x f^*(\boldsymbol{x}))|| \\
f^*(\boldsymbol{x}) &\leq t'
\end{aligned}$$

and

$$
\begin{aligned}
|f^*(\boldsymbol{x}_\delta) - f^*(y)| &\leq ||\boldsymbol{x}_\delta - y|| \\
-f^*(y) &\leq ||(\boldsymbol{x} - t'\nabla_x f^*(\boldsymbol{x})) - (\boldsymbol{x} - t\nabla_x f^*(\boldsymbol{x}))|| \\
-f^*(y) &\leq t - t' \\
-f^*(y) &\leq ||\boldsymbol{x} - y|| - t' \qquad\qquad)
\end{aligned}
$$

Then, if $f^*(\boldsymbol{x}) < t'$ we have

$$
\begin{aligned}
f^*(\boldsymbol{x}) - f^*(y) <& t' + ||\boldsymbol{x} - y|| - t' \\
f^*(\boldsymbol{x}) - f^*(y) <& ||\boldsymbol{x} - y||
\end{aligned}
$$

which is a contradiction so $f^*(\boldsymbol{x}) = t'$ and

$$\boldsymbol{x}_\delta = \boldsymbol{x} - f^*(\boldsymbol{x})\nabla_x f^*(\boldsymbol{x})$$

■

A.2 Parameters and architectures

A.2.1 Datasets

FashionMNIST has 50,000 images for training and 10,000 for test of size $28 \times 28 \times 1$, with 10 classes.

CelebA contains 162,770 training samples, 19,962 samples for test of size $218 \times 178 \times 3$. We have used a subset of 22 labels: *Attractive*, *Bald*, *Big_Nose*, *Black_Hair*, *Blond_Hair*, *Blurry*, *Brown_Hair*, *Eyeglasses*, *Gray_Hair*, *Heavy_Makeup*, *Male*, *Mouth_Slightly_Open*, *Mustache*, *Receding_Hairline*, *Rosy_Cheeks*, *Sideburns*, *Smiling*, *Wearing_Earrings*, *Wearing_Hat*, *Wearing_Lipstick*, *Wearing_Necktie*, *Young*.

Note that labels in CelebA are very unbalanced (see Table 2, with less than 5% samples for *Mustache* or *Wearing_Hat* for instance). Thus we will use Sensibility and Specificity as metrics.

Table 2: CelebA label distribution: proportion of positive samples in training set (testing set) [bold: very unbalanced labels]

Attractive	*Bald*	*Big_Nose*	*Black_Hair*	*Blond_Hair*
0.51 (0.50)	**0.02 (0.02)**	**0.24 (0.21)**	**0.24 (0.27)**	**0.15 (0.13)**
Blurry	*Brown_Hair*	*Eyeglasses*	*Gray_Hair*	*Heavy_Makeup*
0.05 (0.05)	**0.20 (0.18)**	**0.06 (0.06)**	**0.04 (0.03)**	0.38 (0.40)
Male	*Mouth_Slightly_Open*	*Mustache*	*Receding_Hairline*	*Rosy_Cheeks*
0.42 (0.39)	0.48 (0.50)	**0.04 (0.04)**	**0.08 (0.08)**	**0.06 (0.07)**
Sideburns	*Smiling*	*Wearing_Earrings*	*Wearing_Hat*	*Wearing_Lipstick*
0.06 (0.05)	0.48 (0.50)	**0.19 (0.21)**	**0.05 (0.04)**	0.47 (0.52)
Wearing_Necktie	*Young*			
0.12 (0.14)	0.78 (0.76)			

Cat vs Dog contains 17400 training samples, 5800 test samples of various size.

Imagenet contains 1M training samples, 100 000 samples for test of various size.

preprocessing: For FashionMNIST Images are normalized between $[0, 1]$ with no augmentation. For CelebA dataset, data augmentation is used with random crop, horizontal flip, random brightness, and random contrast. For imagenet and cat vs dog we use the standart preprocessing of resnet (with no normalization in $[0, 1]$)

A.2.2 Architectures

As indicated in the paper, linear layers for OTNN and unconstrained networks are equivalent (same number of layers and neurons), but unconstrained networks use batchnorm and ReLU layer for activation, whereas OTNN only use GroupSort2 [5, 53] activation. OTNN are built using *DEEL.LIP*[2] library.

1-Lipschitz networks parametrization. Several soltutions have been proposed to set the Lipschitz constant of affine layers: Weight clipping [6] (WGAN), Frobenius normalization [51] and spectral normalization [43]. In order to avoid vanishing gradients, orthogonalization can be done using Björck algorithm [9]. DEEL.LIP implements most of these solutions, but we focus on layers called *SpectralDense* and *SpectralConv2D*, with spectral normalization [43] and Björck algorithm [9]. Most activation functions are Lipschitz, including ReLU, sigmoid, but we use GroupSort2 proposed by [5], and defined by the following equation:

$$\text{GroupSort2}(x)_{2i,2i+1} = [\min(x_{2i}, x_{2i+1}), \max(x_{2i}, x_{2i+1})]$$

Network architectures used for CelebA dataset are described in Table 3.

Network architectures used for FashionMNIST dataset are described in Table 4. The same OTNN architecture is used for MNIST expermentation presented in Fig. 1.

[2] `https://github.com/deel-ai/deel-lip` distributed under MIT License (MIT)

Table 3: CelebA Neural network architectures: Sconv2D is SpectralConv2D, GS2 is GroupSort2, L2Pool is L2NormPooling, SDense is SpectralDense, BN is BatchNorm, AvgPool is AveragePooling

Dataset	OTNN	Unconstrained NN	
	Layer	Layer	Output size
CelebA	Input	Input	$218 \times 178 \times 3$
	SConv2D, GS2	Conv2D, BN, ReLU	$218 \times 178 \times 16$
	SConv2D, GS2	Conv2D, BN, ReLU	$218 \times 178 \times 16$
	L2Pool	AvgPool	$109 \times 89 \times 16$
	SConv2D, GS2	Conv2D, BN, ReLU	$109 \times 89 \times 32$
	SConv2D, GS2	Conv2D, BN, ReLU	$109 \times 89 \times 32$
	L2Pool	AvgPool	$54 \times 44 \times 32$
	SConv2D, GS2	Conv2D, BN, ReLU	$54 \times 44 \times 64$
	SConv2D, GS2	Conv2D, BN, ReLU	$54 \times 44 \times 64$
	SConv2D, GS2	Conv2D, BN, ReLU	$54 \times 44 \times 64$
	L2Pool	AvgPool	$27 \times 22 \times 64$
	SConv2D, GS2	Conv2D, BN, ReLU	$27 \times 22 \times 128$
	SConv2D, GS2	Conv2D, BN, ReLU	$27 \times 22 \times 128$
	SConv2D, GS2	Conv2D, BN, ReLU	$27 \times 22 \times 128$
	L2Pool	AvgPool	$13 \times 11 \times 128$
	SConv2D, GS2	Conv2D, BN, ReLU	$13 \times 11 \times 128$
	SConv2D, GS2	Conv2D, BN, ReLU	$13 \times 11 \times 128$
	SConv2D, GS2	Conv2D, BN, ReLU	$13 \times 11 \times 128$
	L2Pool	AvgPool	$6 \times 5 \times 128$
	Flatten, SDense, GS2	Flatten, Dense, BN, ReLU	256
	SDense, GS2	Dense,BN, ReLU	256
	SDense	Dense	22

The 1-Lipschitz version of resnet50 is described in Table 5. As the unconstrained version, It has around 25M parameters. For the large version, we simply multiply the number channels in hidden layers by 1.5. The unconstrained version is the standart resnet50 architecture. In the case of imagenet we use the pretrained version provided by tensorflow.

Table 4: FashionMNIST Neural network architectures: Sconv2D is SpectralConv2D, GS2 is GroupSort2, SDense is SpectralDense, BN is BatchNorm, AvgPool is AveragePooling, SGAvgPool is ScaledGlobalAveragePooling (DEEL.LIP), GAvgPool is GlobalAveragePooling

Dataset	OTNN	Unconstrained NN	
	Layer	Layer	Output size
FashionMNIST	Input	Input	$28 \times 28 \times 1$
	SConv2D, GS2	Conv2D, BN, ReLU	$28 \times 28 \times 96$
	SConv2D, GS2	Conv2D, BN, ReLU	$28 \times 28 \times 96$
	SConv2D, GS2	Conv2D, BN, ReLU	$28 \times 28 \times 96$
	SConv2D (stride=2), GS2	Conv2D (stride=2), BN, ReLU	$14 \times 14 \times 96$
	SConv2D, GS2	Conv2D, BN, ReLU	$14 \times 14 \times 192$
	SConv2D, GS2	Conv2D, BN, ReLU	$14 \times 14 \times 192$
	SConv2D, GS2	Conv2D, BN, ReLU	$14 \times 14 \times 192$
	SConv2D (stride=2), GS2	Conv2D (stride=2), BN, ReLU	$7 \times 7 \times 192$
	SConv2D, GS2	Conv2D, BN, ReLU	$7 \times 7 \times 384$
	SConv2D, GS2	Conv2D, BN, ReLU	$7 \times 7 \times 384$
	SConv2D, GS2	Conv2D, BN, ReLU	$7 \times 7 \times 384$
	SGAvgPool	GAvgPool	384
	SDense	Dense	10

Table 5: 1-lip resnet architecture for Imagenet and cat vs dog: Sconv2D is SpectralConv2D, GS2 is GroupSort2, SDense is SpectralDense, BC is Batchcentering (centeing without normalization), SL2npool is ScaledL2NormPooling2D, SGAvgl2Pool is ScaledGlobalL2NormPooling2D, GAvgPool is GlobalAveragePooling

Layer				output
Input				$224 \times 224 \times 3$
SConv2D 7-64 (stride=2), BC, GS2				$112 \times 112 \times 64$
InvertibleDownSampling				$56 \times 56 \times 256$
SConv2D 1×1	64	BC, GS2	$\times 3$	$56 \times 56 \times 256$
SConv2D 3×3	64	BC, GS2		
SConv2D 1×1	256	BC		
add-lip		BC, GS2		
SL2npool				$28 \times 28 \times 256$
SConv2D 1×1	128	BC, GS2	$\times 4$	$28 \times 28 \times 512$
SConv2D 3×3	128	BC, GS2		
SConv2D 1×1	512	BC		
add-lip		BC, GS2		
SL2npool				$14 \times 14 \times 512$
SConv2D 1×1	256	BC, GS2	$\times 6$	$14 \times 14 \times 1024$
SConv2D 3×3	256	BC, GS2		
SConv2D 1×1	1024	BC		
add-lip		BC, GS2		
SL2npool				$7 \times 7 \times 1024$
SConv2D 1×1	256	BC, GS2	$\times 3$	$7 \times 7 \times 2048$
SConv2D 3×3	256	BC, GS2		
SConv2D 1×1	1024	BC		
add-lip		BC, GS2		
SGAvgl2Pool				2048
SDense				1 cat vs dog 1000 imagenet

A.2.3 Losses and optimizer

An extension of $\mathcal{L}^{hKR}$ to the multiclass case with q classes. has also been proposed in [53] The idea is to learn q 1-Lipschitz functions $f_1, \dots, f_q$, each component f_i being a *one-versus-all* binary classifier. The loss proposed was the following

$$\mathcal{L}^{hKR}_{\lambda,m}(f_{1,\dots,q}) = \sum_{k=1}^{q} \left[\mathop{\mathbb{E}}_{\mathbf{x}\sim\neg P_k} [f_k(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}\sim P_k} [f_k(\mathbf{x})] \right] + \lambda \mathop{\mathbb{E}}_{\mathbf{x},y\sim\bigcup_{k=1}^{q} P_k} (H\left(f_1(\boldsymbol{x}), \dots, f_q(\boldsymbol{x}), y, m\right) \tag{4}$$

with :

$$H\left(f_1(\boldsymbol{x}), \dots, f_q(\boldsymbol{x}), y, m\right) = (m - f_y(\boldsymbol{x}))_+ + \sum_{k \neq y} (m + f_k(\boldsymbol{x}))_+$$

This formulation has three main drawbacks: (i) for large number of classes several outputs may have few or no positive sample within a batch leading to slow convergence, (ii) weight of $f_y(\boldsymbol{x})$ (the function of the true class) with respect to the other decreases when the number of classes increases, (iii) the expectancy has to be evaluated through the batch, making the loss dependant of the size of the batch.

To overcome these drawbacks, we propose first to replace the Hinge term H with a softmax weighted version. The softmax on all but true class is defined by:

$$\sigma(f_k(\boldsymbol{x}), y, \alpha) = \frac{e^{\alpha * f_k(\boldsymbol{x})}}{\sum_{j \neq y} e^{\alpha * f_j(\boldsymbol{x})}}$$

We can define a weighted version of H function:

$$H_\sigma\left(f_1(\boldsymbol{x}),\ldots,f_q(\boldsymbol{x}),y,m,\alpha\right) = (m - f_y(\boldsymbol{x}))_+ + \sum_{k\neq y} \sigma(f_k(\boldsymbol{x}),y,\alpha) * (m + f_k(\boldsymbol{x}))_+$$

In this function, the value of $f_y(\boldsymbol{x})$ for the true class maintains consistent weight relative to the values of other functions, regardless of the number of classes. α is a temperature parameter. Initially, the softmax behaves like an average as all the values of f_k are close. However, during the learning process, as the values of $|f_k|$ increase, the softmax transitions to function like a maximum. Similarly, if a low value is chosen for α , the softmax behaves as an average, resulting in a one vs all hKR loss. By choosing a higher value for α, the softmax unbalances the weights. Thus the loss persists as a one vs all hKR but incorporates a re-weighting of the opposing classes for each targeted class.

We also propose a sample-wise and weighted version of the KR part (left term in Eq 4). to get the proposed loss:

$$\begin{aligned}\mathcal{L}^{hKR}_{\lambda,m,\alpha}(f_{1,\ldots,q},x,y) = &\left[\sum_{k\neq y}\left[f_k(\boldsymbol{x}) * \sigma(f_k(\boldsymbol{x}),y,\alpha)\right] - f_y(\mathbf{x})\right] \\ &+ \lambda * H_\sigma\left(f_1(\boldsymbol{x}),\ldots,f_q(\boldsymbol{x}),y,m,\alpha\right)\end{aligned} \tag{5}$$

It's important to note that this definition only applys to the balanced multiclass case (as in FashionMnist and ImageNet). In the unbalanced scenario, the weight must be rescaled according to the a priori distribution of the classes.

For CelebA, with hyperparameters λ is set to 20, and $m = 1$. For FashionMNIST, we use Eq. 5, λ is set to 5, $\alpha = 10$ and $m = 0.5$. For cat vs dog λ is set to 10 and $m = 18$. For imagenet λ is set to 500, $\alpha = 200$ and $m = 0.05$.

We train all networks with ADAM optimizer [36]. We use a batch size of 128, 200 epochs , and a fixed learning rate $1e{-2}$ for CelebA. For FashionMNIST we perform 200 epochs with a batch size of 128. We fix the learing rate to $5e{-4}$ for the 50 first epochs, $5e{-5}$ for the epochs 50-75, $1e{-6}$ for the last epochs. For cat vs dog we perform 200 epochs with a batch size of 256. We fix the learing rate to $1e{-2}$ for the 100 first epochs, $1e{-3}$ for the epochs 100-150, $1e{-4}$ for the epochs 150-180 and $1e{-9}$ for the last epochs. For imagenet we perform 40 epochs with a batch size of 512. We fix the learing rate to $5e{-4}$ for the 30 first epochs, $5e{-5}$ for the epochs 30-35, $1e{-5}$ for the epochs 35-38 and $1e{-9}$ for the last epochs.

A.3 Complementary results

A.3.1 FashionMNIST performances and ablation study

Table 6 presents different performance resuts on FashionMNIST. First line is the reference unconstrained network. Second line shows the performances of the new version of $\mathcal{L}^{hKR}_{\lambda,\alpha}$. Table 6 also shows that the new version of the $\mathcal{L}^{hKR}_{\lambda,m,\alpha}$ in the multiclass case (Eq. 5) outperforms the $\mathcal{L}^{hKR}_{\lambda,m}$defined in [53] (Eq. 4). Obviously, the accuracy enhancement is obtained at the expense of the robustness. The main interest of this new loss is to provide a wider range in the accuracy/robustness trade-off.

Table 6: FashionMNIST accuracy comparison with the different version of multiclass $\mathcal{L}^{hKR}_{\lambda,m}$. For the fixed margin, we use the one that performs best by parameter tuning (i.e. $m = 0.5$)

Model	Accuracy
Unconstrained	88.5
OTNN $\mathcal{L}^{hKR}_{\lambda,m}$multiclass version [53] ($\lambda = 10$, $m = 0.5$)	72.2
(Ours) OTNN $\mathcal{L}^{hKR}_{\lambda,m,\alpha}$ ($\lambda = 10$, $m = 0.5$,$\alpha = 10$)$(Eq.\ 5)$	**88.6**

A.3.2 CelebA performances

Table 7 presents the Sensibility and Specificity for each label reached by Unconstrained network and OTNN.

As a reminder, given True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN) samples, Sensitivity (true positive rate or Recall) is defined by:

$$Sens = \frac{TP}{TP + FN}$$

Specificity (true negative rate) is defined by:

$$Spec = \frac{TN}{TN + FP}$$

Table 7: CelebA performance results for unconstrained and OTNN networks

Model	Metrics: Sensibility/Specificity			
	Attractive	*Bald*	*Big_Nose*	*Black_Hair*
Unconstrained	**0.83 / 0.81**	0.64 / 1.00	**0.65 / 0.87**	**0.74 / 0.95**
OTNN	0.80 / 0.75	**0.87 / 0.83**	0.73 / 0.70	0.78 / 0.84
	Blond_Hair	*Blurry*	*Brown_Hair*	*Eyeglasses*
Unconstrained	**0.86 / 0.97**	**0.49 / 0.99**	**0.80 / 0.88**	**0.96 / 1.00**
OTNN	0.86 / 0.89	0.66 / 0.72	0.81 / 0.73	0.80 / 0.89
	Gray_Hair	*Heavy_Makeup*	*Male*	*Mouth_Slightly_Open*
Unconstrained	0.62 / 0.99	**0.84 / 0.95**	**0.98 / 0.98**	**0.93 / 0.94**
OTNN	**0.84 / 0.83**	0.89 / 0.83	0.92 / 0.89	0.80 / 0.89
	Mustache	*Receding_Hairline*	*Rosy_Cheeks*	*Sideburns*
Unconstrained	0.47 / 0.99	0.47 / 0.98	0.46 / 0.99	**0.79 / 0.98**
OTNN	**0.86 / 0.76**	**0.81 / 0.79**	**0.82 / 0.80**	0.79 / 0.82
	Smiling	*Wearing_Earrings*	*Wearing_Hat*	*Wearing_Lipstick*
Unconstrained	**0.90 / 0.95**	**0.84 / 0.90**	**0.89 / 0.99**	**0.90 / 0.96**
OTNN	0.84 / 0.88	0.78 / 0.72	0.86 / 0.90	0.90 / 0.89
	Wearing_Necktie	*Young*		
Unconstrained	0.75 / 0.98	**0.95 / 0.65**		
OTNN	**0.87 / 0.86**	0.79 / 0.69		

A.4 Complementary explanations metrics

A.4.1 Explanation attribution methods

An attribution method provides an importance score for each input variables x_i in the output $f(x)$. The library used to generate the attribution maps is Xplique [21].

For a full description of attribution methods, we advise to read [18], Appendix B. We will only remind here the equations of

- Saliency: $g(\boldsymbol{x}) = |\nabla_{\boldsymbol{x}} f(\boldsymbol{x})|$
- SmoothGrad: $g(\boldsymbol{x}) = \underset{\delta \sim \mathcal{N}(0, \mathbf{I}\sigma)}{\mathbb{E}} (\nabla f(\boldsymbol{x} + \delta))$

SmoothGrad is evaluated on $N = 50$ samples on a normal distribution of standard deviation $\sigma = 0.2$ around x. Integrated Gradient [59], noted IG, is also evaluated on $N = 50$ samples at regular intervals. Grad-CAM [52], noted GC, is classically applied on the last convolutional layer.

A.4.2 XAI metrics

For the experiments we use four fidelity metrics, evaluated on 1000 samples of test datasets:

- Deletion [47]: it consists in measuring the drop of the score when the important variables are set to a baseline state. Formally, at step k, with u the k most important variables according to an attribution method, the Deletion$^{(k)}$ score is given by:

  $$\text{Deletion}^{(k)} = f(\boldsymbol{x}_{[\boldsymbol{x}_u = \boldsymbol{x}_0]})$$

  The AUC of the Deletion scores is then measured to compare the attribution methods ($\downarrow$ is better). The baseline x_0 can either be a zero value (*Deletion-zero*), or a uniform random value (*Deletion-uniform*).

- Insertion [47]: this metric is the inverse of Deletion, starting with an image in a baseline state and then progressively adding the most important variables. Formally, at step k, with u the most important variables according to an attribution method, the Insertion$^{(k)}$ score is given by:

  $$\text{Insertion}^{(k)} = f(\boldsymbol{x}_{[\boldsymbol{x}_{\overline{u}} = \boldsymbol{x}_0]})$$

  The AUC is also measured to compare attribution methods ($\uparrow$ is better). The baseline is the same as for Deletion.

- μFidelity [8]: this metric measures the correlation between the fall of the score when variables are put at a baseline state and the importance of these variables. Formally:

  $$\mu\text{Fidelity} = \underset{\substack{u \subseteq \{1, \dots, d\} \\ |u| = k}}{\text{Corr}} \left(\sum_{i \in u} g(\boldsymbol{x})_i, f(\boldsymbol{x}) - f(\boldsymbol{x}_{[\boldsymbol{x}_u = \boldsymbol{x}_0]}) \right)$$

  For all experiments, k is equal to 20% of the total number of variables, and cutting the image in a grid of 20×20 (9×9 for cat vs dog and imagenet). The baseline is the same as the one used by Deletion. Being a correlation score, we can either compare attribution methods, or different neural networks on the same attribution method ($\uparrow$ is better).

- Robustness-Sr [31]: this metric evaluate the average adversarial distance when the attack is done only on the most relevant features. Formally, given the u most important variables:

  $$\text{Robustness-Sr} = \left\{ \underset{\delta}{min} ||\delta|| s.t. f(\boldsymbol{x} + \delta) \neq {}_x, \delta_{\overline{u}} = 0 \right\}$$

  where $\delta_{\overline{u}} = 0$ indicates that adversarial attack is authorized only on the set u. The AUC is measured to compare attribution methods ($\downarrow$ is better). Note this metric cannot be used to compare different networks, since it depends on the robustness of the network.

We use also several other metrics:

- Distances between explanations: to compare two explanation $f(x)$, we use either L_2 distance, or $1-\rho$ where ρ is the Spearman rank correlation [2, 20, 60] ($\downarrow$ is better).
- Explanation complexity: we use the JPEG compression size as a proxy of the Kolmogorov complexity ($\downarrow$ is better).
- Stability: As proposed in [68], the Stability is evaluated by the average distance of explanations provided for random samples drawn in a ball of radius 0.3 (0.15 for cat vs dog and imagenet) around x. As before, the distance can be either L_2 or $1-\rho$ ($\downarrow$ is better).

A.4.3 Supplementary metric results

In this section we present several experiments and metrics that we were not able to insert in the core of the paper.

Deletion-zero and Insertion-zero are evaluated on CelebA and FashionMNIST dataset. It is known that the baseline value can be a bias for these metrics, and we are convinced that it has a higher influence with 1-Lipschitz networks. Even if results for Deletion-zero and Insertion-zero are less obvious than for Deletion and Insertion Uniform, we can see in Table 8, that for these metrics, the rank of Saliency is most of the time higher for OTNN.

Table 8: Insertion and Deletion metrics evaluation; GC: GradCam, GI: Gradient.Input, IG: Integrated Gradient, Saliency Rk : Rank (comparison by line only : in bold best score)

Dataset	Network	Deletion-Zero ($\downarrow$ is better)					
		GC	GI	IG	Rise	Saliency	SmoothGrad
		Deletion-Zero					
CelebA	OTNN	8.01	7.04	7.05	7.09	6.98 (Rk2)	**6.96**
	Unconstrained	5.77	4.56	4.38	5.07	**4.13** (Rk1)	4.51
Fashion-	OTNN	0.24	0.16	**0.15**	0.26	0.20 (Rk4)	0.19
MNIST	Unconstrained	0.33	0.28	0.23	**0.16**	0.38 (Rk5)	0.39
		Insertion-zero ($\uparrow$ is better)					
CelebA	OTNN	10.26	11.63	11.58	**15.50**	10.06 (Rk6)	10.10
	Unconstrained	14.24	11.71	12.37	**15.70**	6.67 (Rk6)	7.65
Fashion-	OTNN	0.31	0.46	**0.47**	0.36	0.36 (Rk4)	0.39
MNIST	Unconstrained	0.53	0.59	0.68	**0.73**	0.45 (Rk6)	0.46

To leverage the bias of the baseline value, as proposed in [31] we evaluated the Robustness-SR metric, Saliency map on OTNN achieves top-ranking scores. One might argue that scores for unconstrained networks are lower, but this is directly linked to the higher intrinsic robustness of OTNNand thus cannot be compared.

Table 9: Robustness-SR metrics evaluation; GC: GradCam, GI: Gradient.Input, IG: Integrated Gradient, Saliency Rk : Rank (comparison by line only : in bold best score)

Dataset	Network	Robustness-SR ($\downarrow$ is better)					
		GC	GI	IG	Rise	Saliency	SmoothGrad
CelebA	OTNN	28.54	14.01	13.28	30.54	**11.64** (Rk1)	12.65
	Unconstrained	11.11	9.19	10.00	15.15	7.38 (Rk2)	**7.20**
Fashion-	OTNN	**1.69**	3.31	3.36	3.27	2.29 (Rk3)	2.01
MNIST	Unconstrained	1.17	1.36	1.17	**1.15**	1.21 (Rk4)	1.25

The full results for the explanation complexity is given on Table 10. The complexity is still lower for OTNN on FashionMNIST, even if the gap with Unconstrained networks is narrower than for CelebA.

Table 10: Complexity of Saliency map by JPEG compression (kB): lower is better

	CelebA	FashionMNIST
OTNN	9.48	0.92
Unconstrained	16.84	0.94

A.5 Complementary qualitative results

In this section, we provide more samples of qualitative results and couterfactual exlanations for OTNN, based on the gradient, i.e. $\boldsymbol{x} - t * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x})$ for $t > 1$.

A.5.1 FashionMNIST

Fig. 6 gives more results on FashionMNIST.

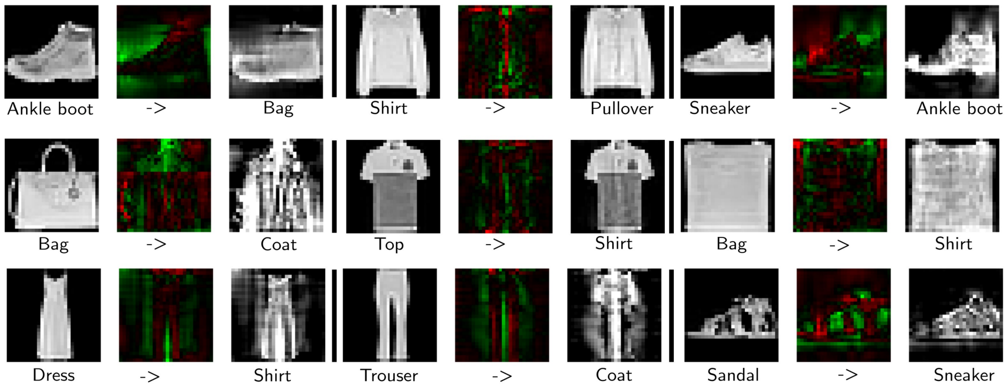


Figure 6: FashionMNIST samples

A.5.2 CelebA

We presents results for other labels of CelebA. For ethic concerns we have hidden labels that can be subject to misinterpretation, such as *Attractive*, *Male*, *Big_Nose*. Fig. 7 to 25 present more results on the labels presented in the core of the paper, *Mouth_Slightly_Open*, *Mustache*,*Wearing_Hat*.

A.5.3 Cat vs Dog

We present some supplementary comparison of Saliency Maps and counterfactual examples for cat vs dog(Fig. 26 and 27).

A.5.4 Imagenet

We present some supplementary comparison of Saliency Maps Imagenet (Fig. 28). As pointed out previously, our model doesn't produce significant counterfacutal explanations on Imagenet.

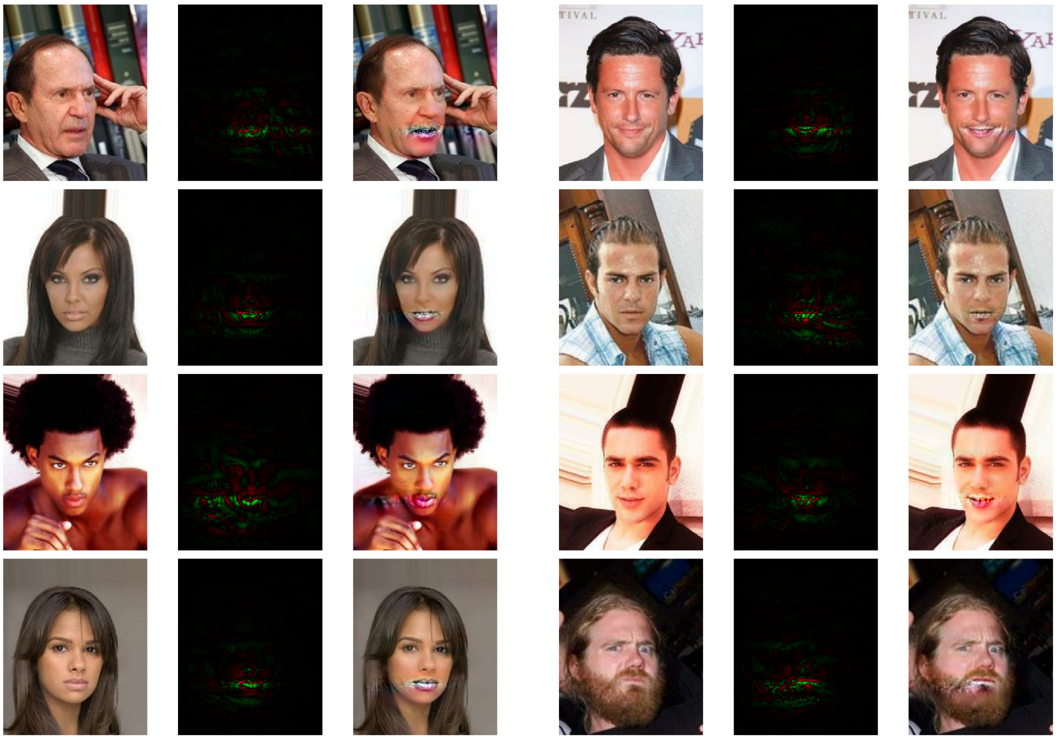


Figure 7: Samples from label Mouth_slightly_open: left source image (closed) , center difference image, right counterfactual (open) of form $\boldsymbol{x} - 10 * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x})$

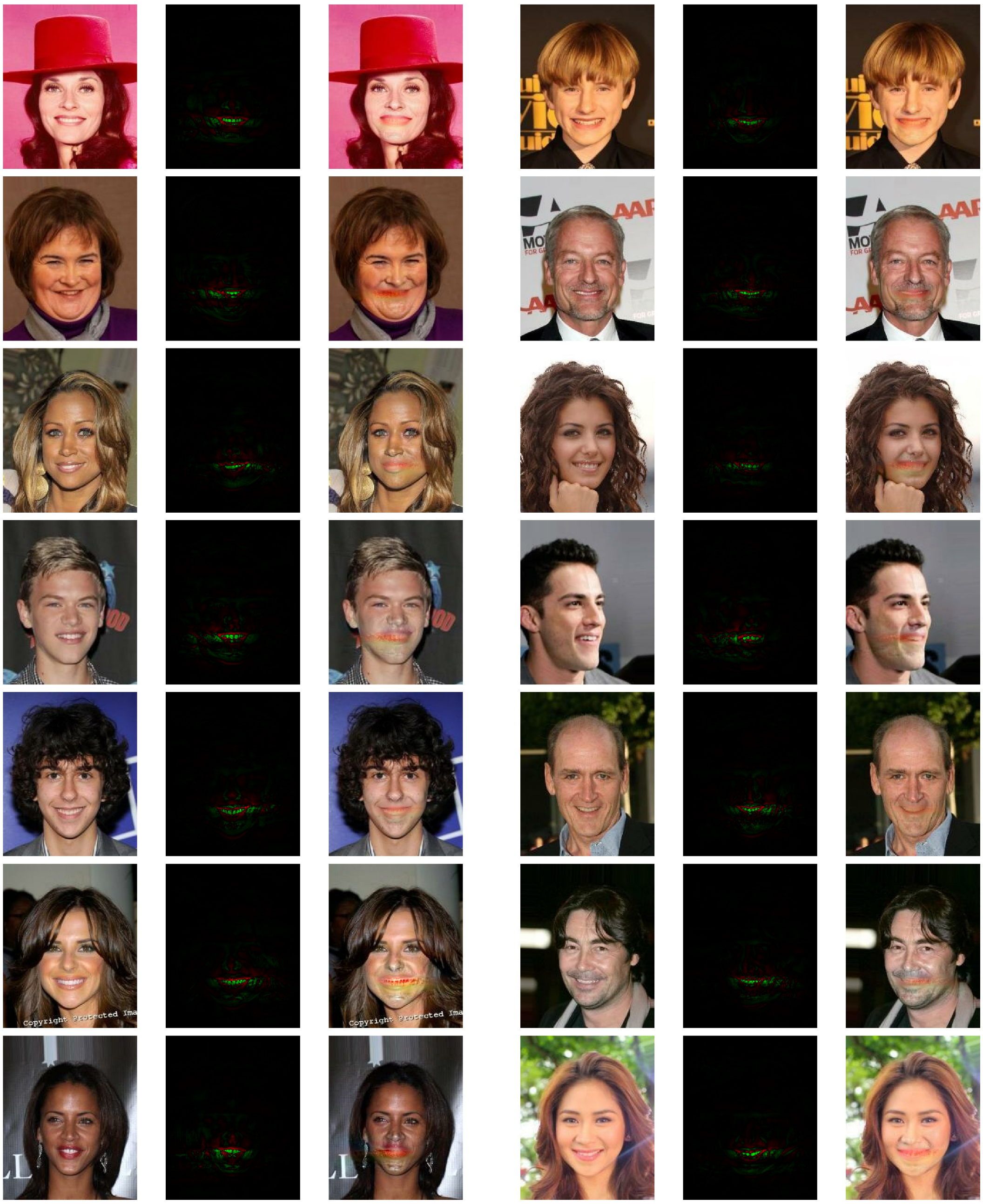


Figure 8: Samples from label Mouth_slightly_open: left source image (open) , center difference image, right counterfactual (close) of form $\boldsymbol{x} - 10 * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x})$

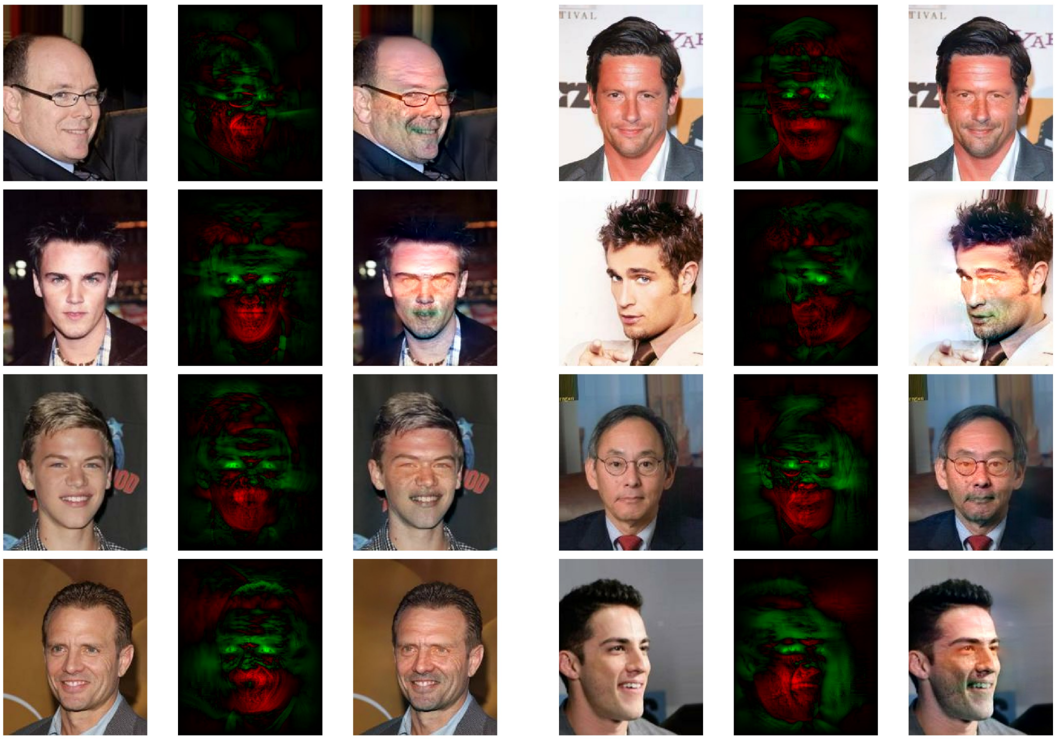


Figure 9: Samples from label Mustache: left source image (no mustache) , center difference image, right counterfactual (mustache) of form $\boldsymbol{x} - t * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x})$ with $t \in \{5, 10, 20\}$

Figure 10: Samples from label Mustache: left source image (Mustache) , center difference image, right counterfactual (Non Mustache) of form $\boldsymbol{x} - t * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x})$, $t \in 5, 10$

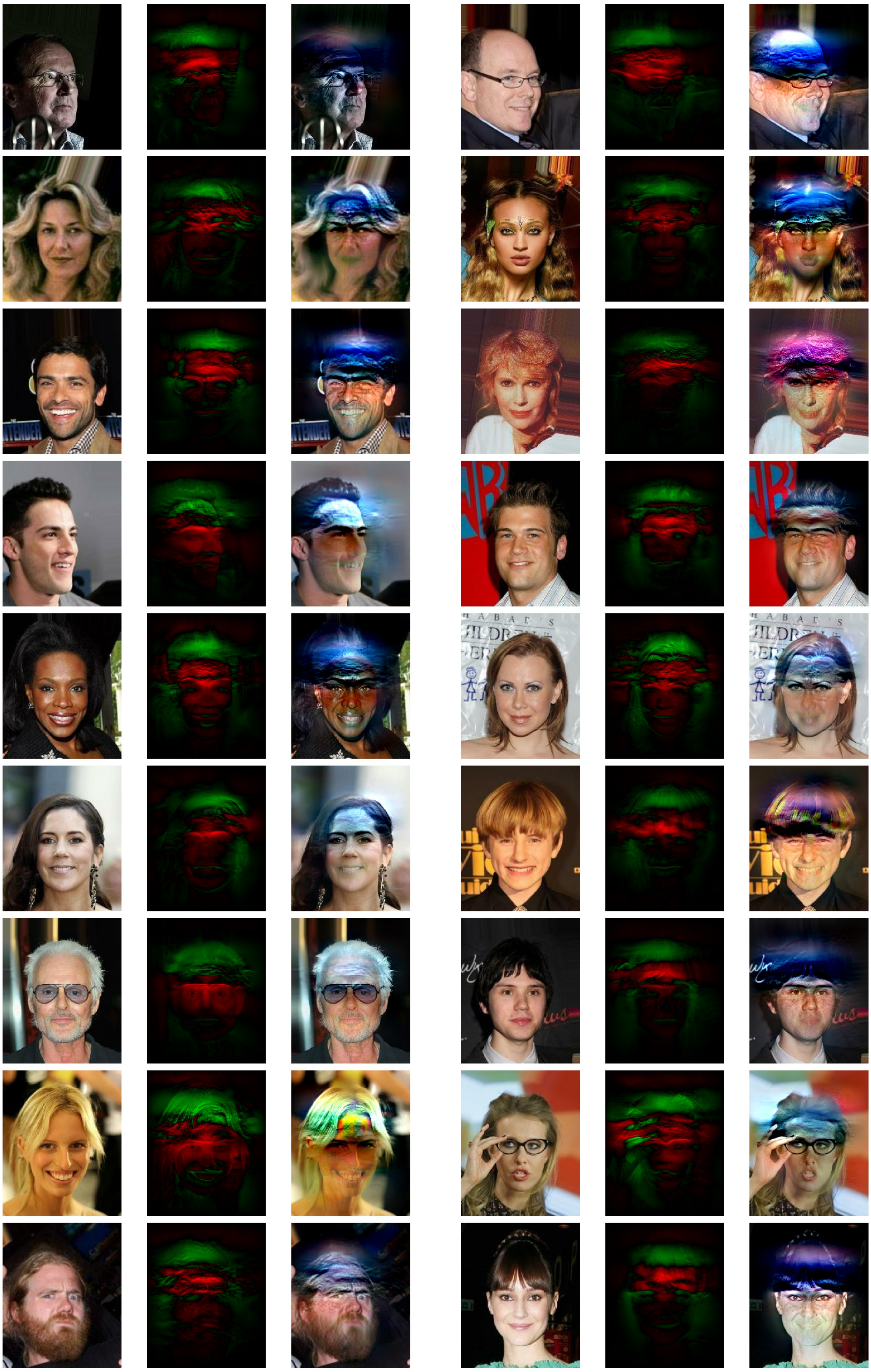
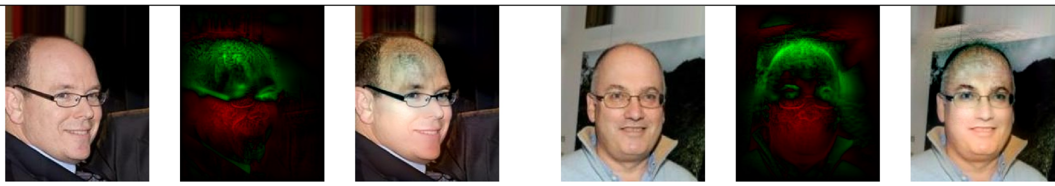


Figure 11: Samples from label Wearing Hat: left source image (No Hat) , center difference image, right counterfactual (Hat) of form $\boldsymbol{x} - t * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x})$, $t \in 5, 10$

Figure 12: Samples from label Wearing Hat: left source image (Hat) , center difference image, right counterfactual (No Hat) of form $\boldsymbol{x} - t * \hat{f}(\boldsymbol{x})\nabla_x \hat{f}(\boldsymbol{x}), t \in 5, 10$

Bald $\rightarrow$ "not" Bald



"not" Bald $\rightarrow$ Bald

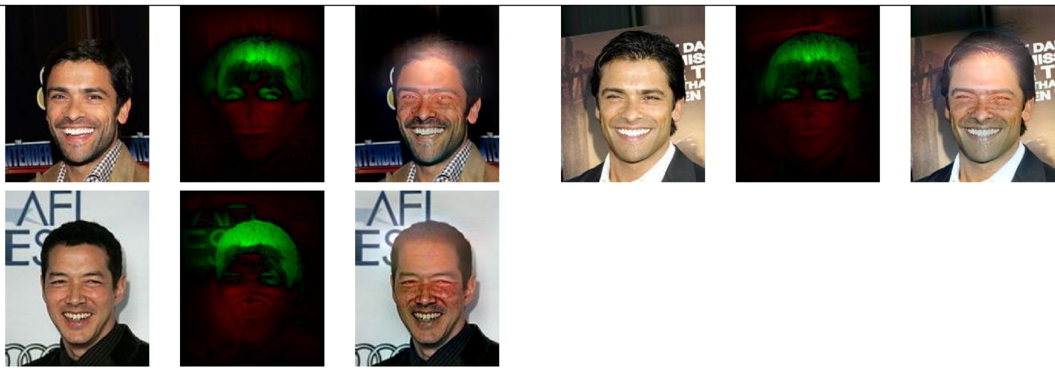


Figure 13: Samples from label Bald

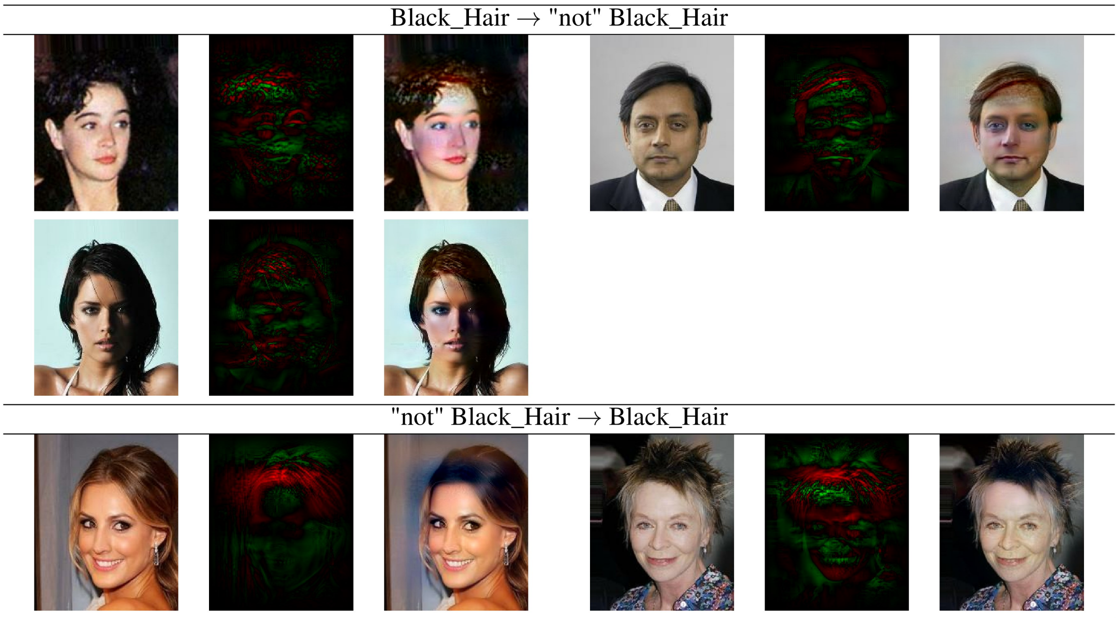


Figure 14: Samples from label Black_Hair

Figure 15: Samples from label Blond_Hair

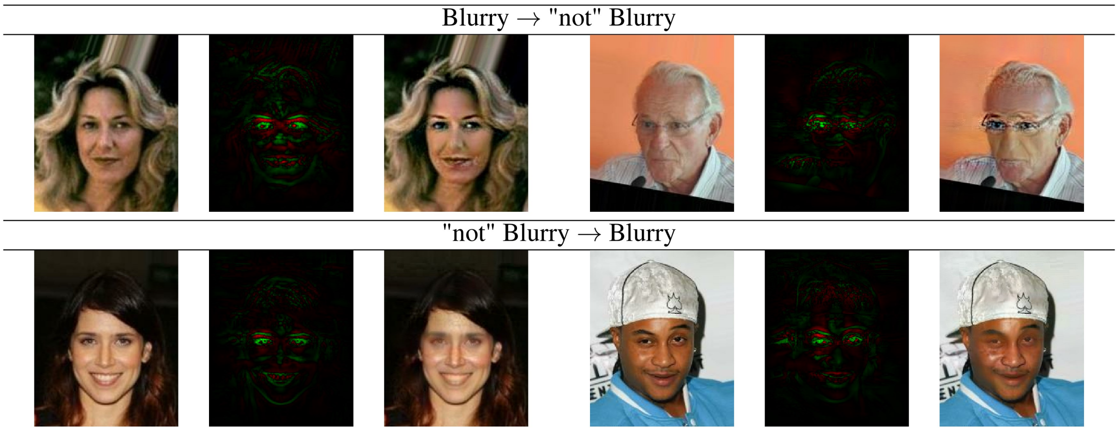


Figure 16: Samples from label Blurry

Brown_Hair → "not" Brown_Hair

"not" Brown_Hair → Brown_Hair

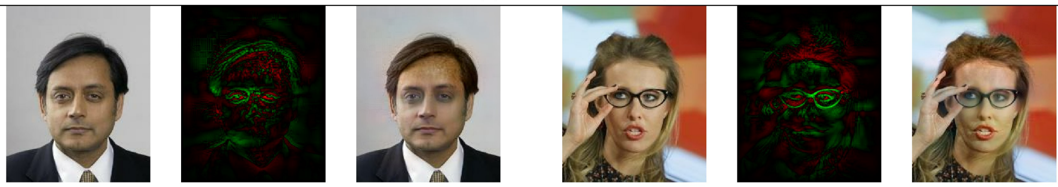
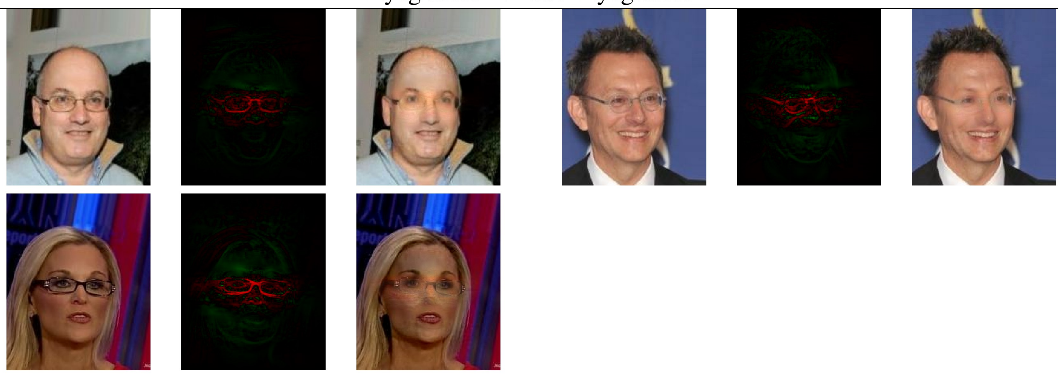


Figure 17: Samples from label Brown_Hair

Eyeglasses → "not" Eyeglasses



"not" Eyeglasses → Eyeglasses

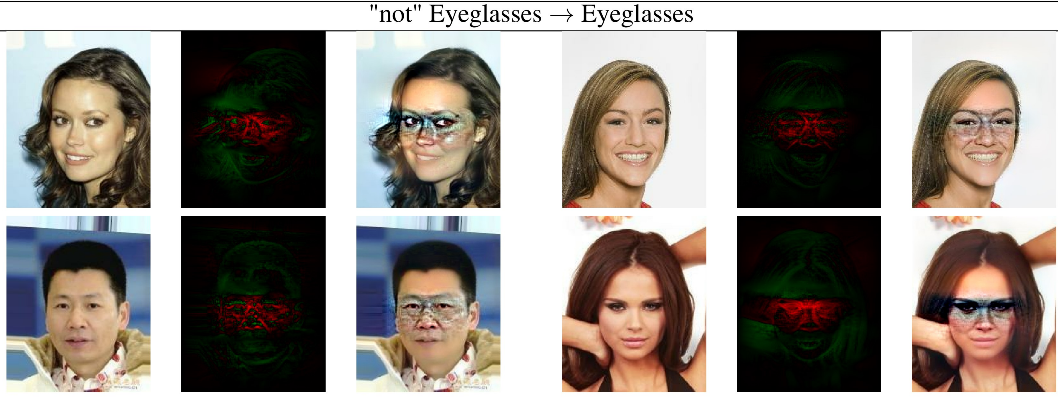


Figure 18: Samples from label Eyeglasses

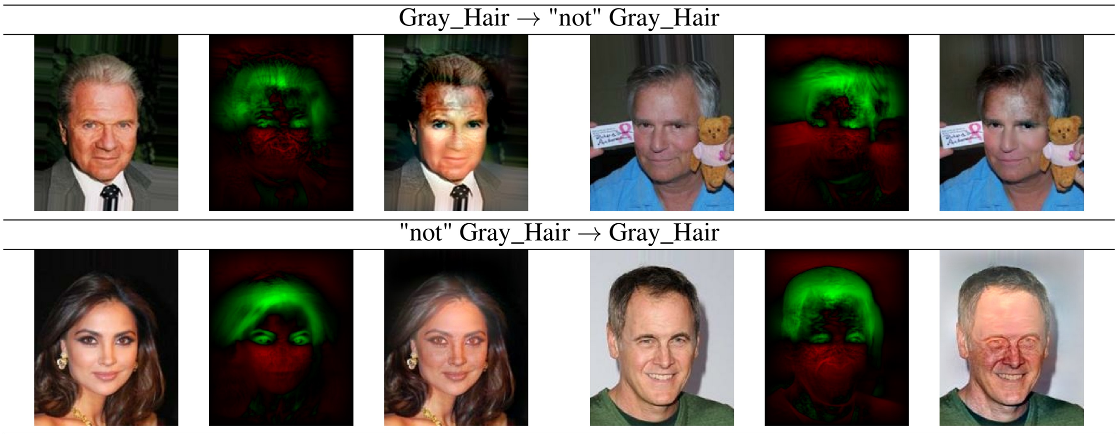


Figure 19: Samples from label Gray_Hair

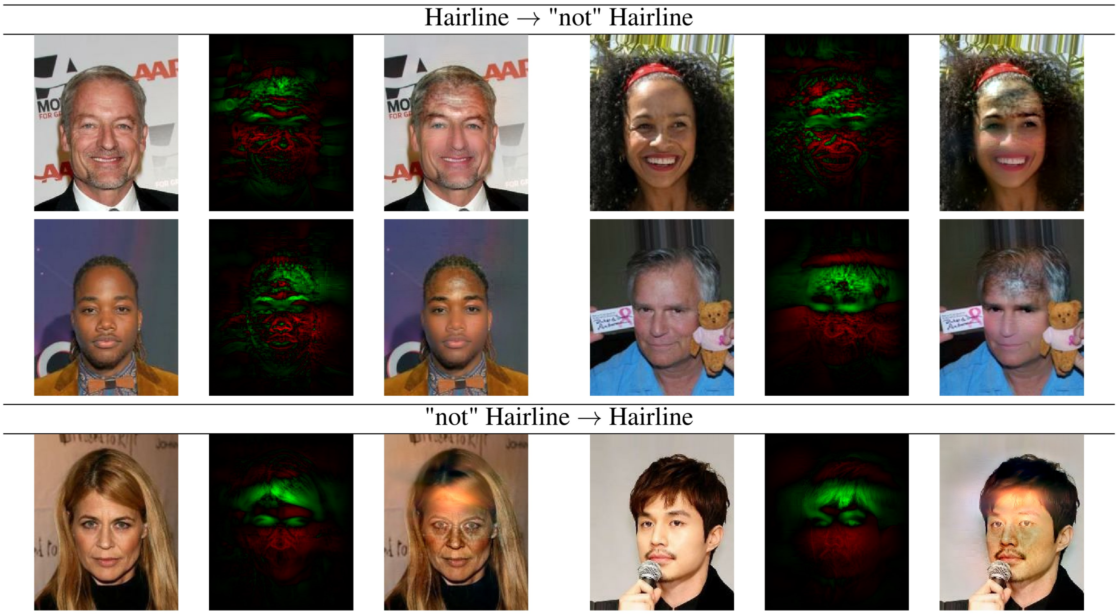


Figure 20: Samples from label Hairline

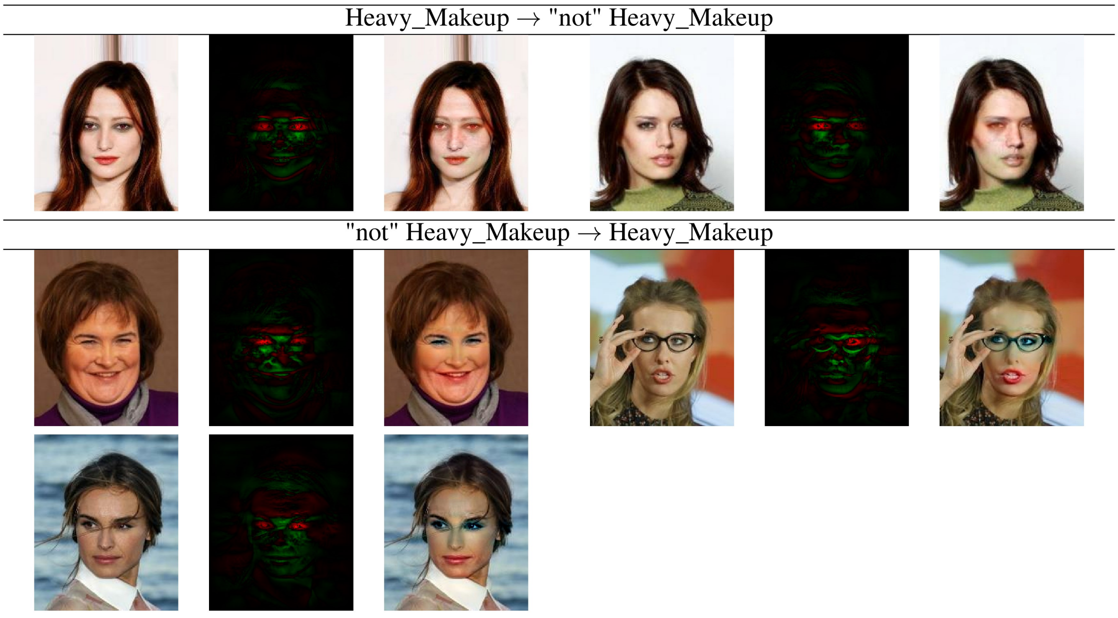


Figure 21: Samples from label Heavy_Makeup

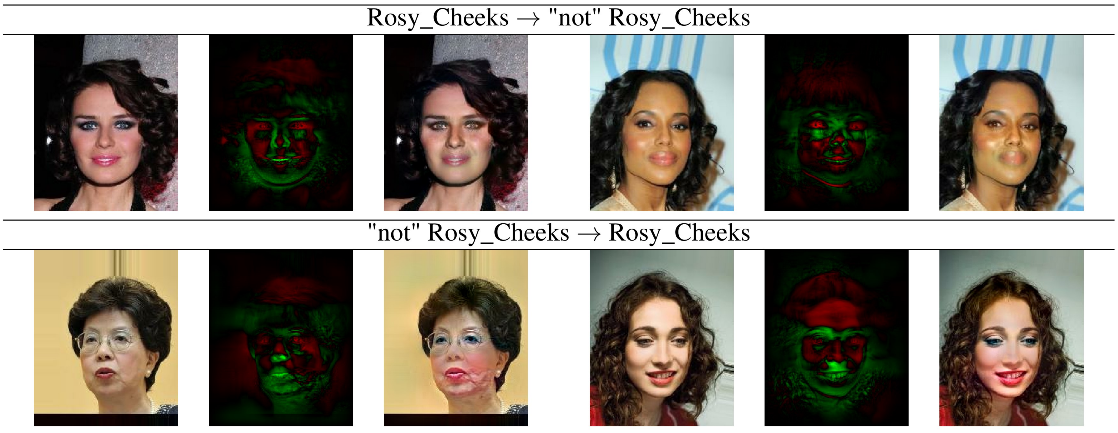


Figure 22: Samples from label Rosy_Cheeks

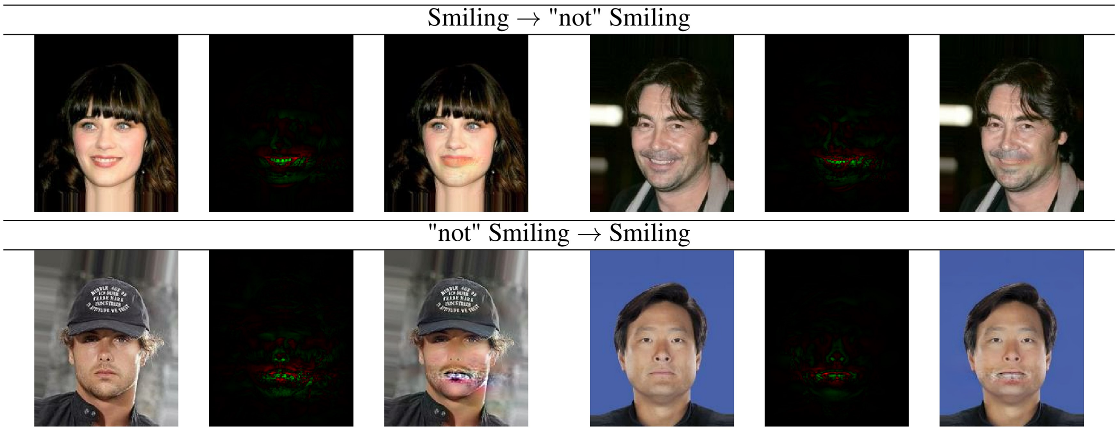


Figure 23: Samples from label Smiling

Figure 24: Samples from label Wearing_Lipstick

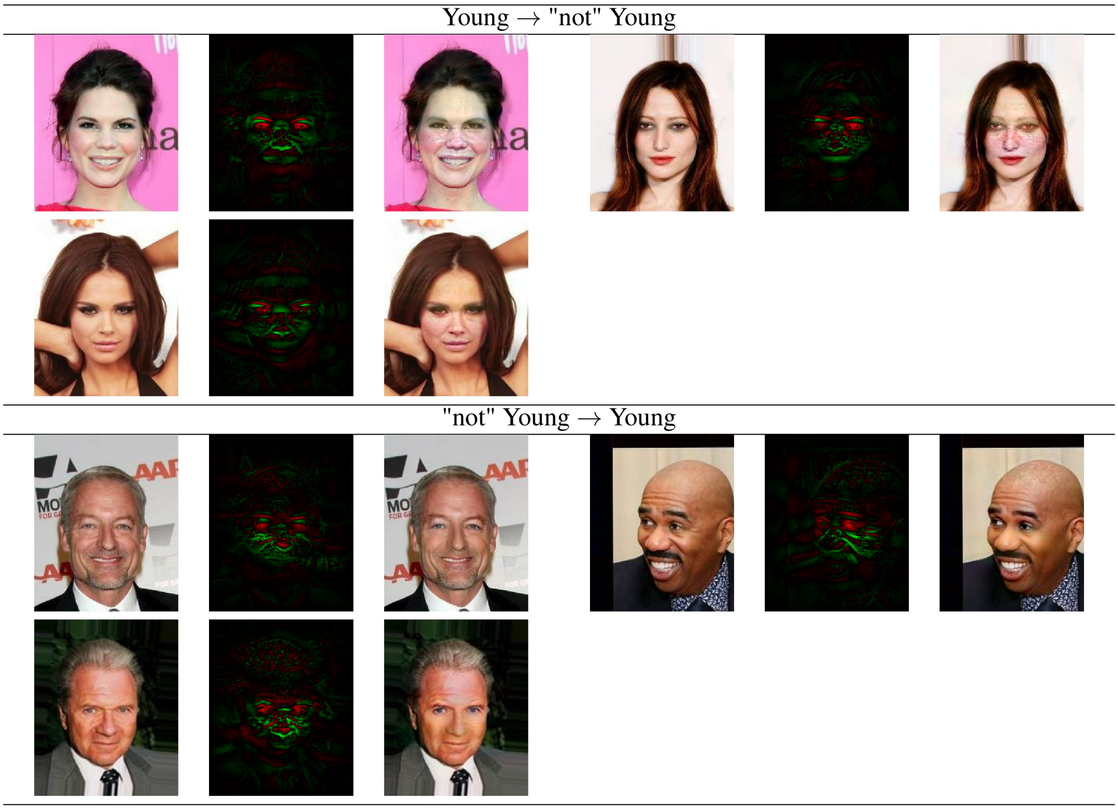
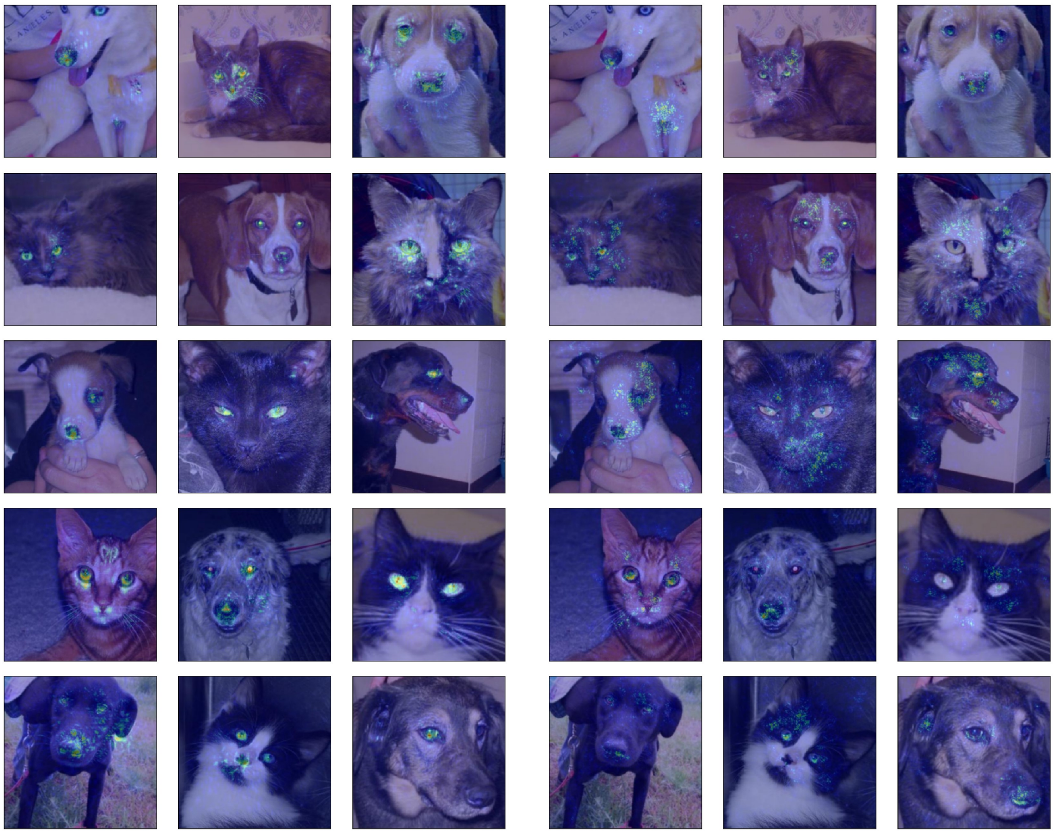


Figure 25: Samples from label Young



(a) OTNN (b) Unconstrained

Figure 26: Cat vs Dog Saliency Map samples

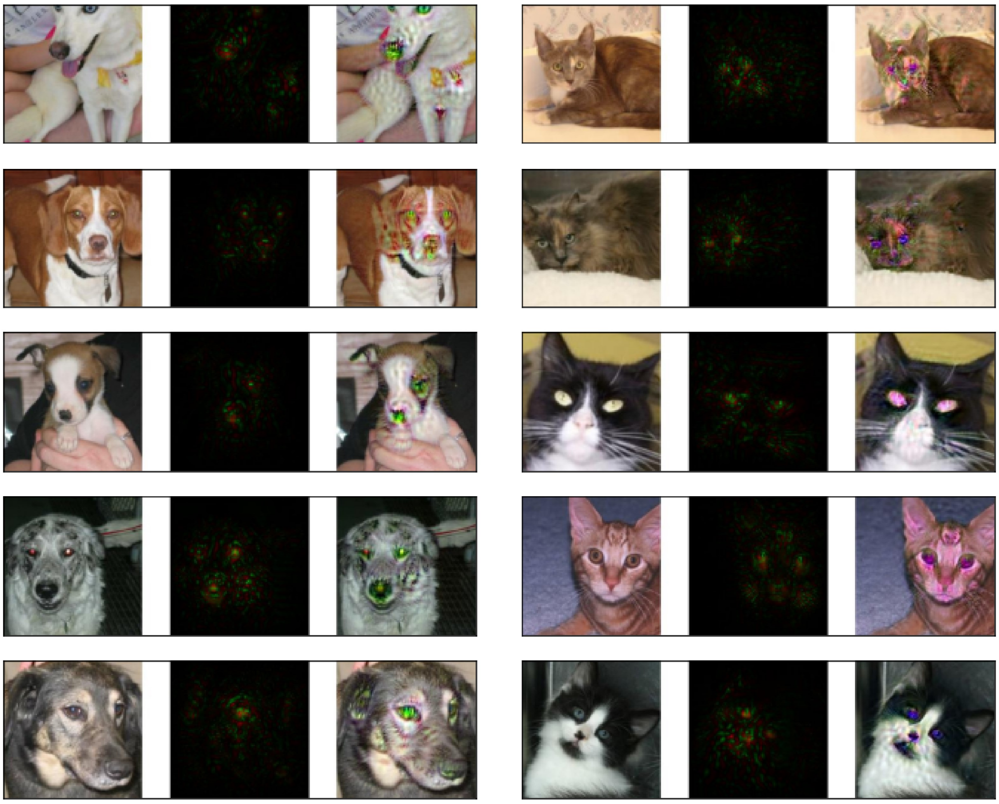


Figure 27: Cat vs Dog Saliency counterfactual samples. Left dog to cat, right cat to dog

(a) OTNN (b) Unconstrained

Figure 28: Imagenet Saliency Map samples