A Supplementary material

A.1 Proofs

In this section of the appendix, we show how current robust training methods can be derived from our FAR objectives. The notation is kept as in Chapter 3.1 from the paper.

A.1.1 Proof of Equation (7)

Setting $\lambda = 0$ in Equation (6) straightforwardly results as follows.

$$\begin{aligned}
\theta^* &= \arg\min_\theta \sum_{\mathbf{x}\in D} \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} \left\{ l(\tilde{\mathbf{x}},y,f) + \lambda \cdot d_s\left[\mathrm{S}(\tilde{\mathbf{x}},f), \mathrm{S}^T(\mathbf{x},f)\right] \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} \left\{ l(\tilde{\mathbf{x}},y,f) + 0 \cdot d_s\left[\mathrm{S}(\tilde{\mathbf{x}},f), \mathrm{S}^T(\mathbf{x},f)\right] \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} l(\tilde{\mathbf{x}},y,f)
\end{aligned}$$

A.1.2 Proof of Equation (8)

In order to derive the IG-NORM objective from Equation (5), we use IG as saliency maps, set the distance metric d_s to be the L_1-norm induced distance and choose the baseline for IG to be the unperturbed input to the network, i.e. $\mathbf{b} = \mathbf{x}$. Then, Equation (5) becomes as follows.

$$\begin{aligned}
\theta^* &= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} d_s\left[\mathrm{S}(\tilde{\mathbf{x}},f), \mathrm{S}^T(\mathbf{x},f)\right] \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} \|\mathrm{S}(\tilde{\mathbf{x}},f) - \mathrm{S}^T(\mathbf{x},f)\|_1 \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} \|\mathrm{IG}(\tilde{\mathbf{x}},\mathbf{x}) - \mathrm{IG}(\mathbf{x},\mathbf{x})\|_1 \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} \|\mathrm{IG}(\tilde{\mathbf{x}},\mathbf{x})\|_1 \right\}
\end{aligned}$$

Note that $\mathrm{IG}(\mathbf{x}, \mathbf{x}) = 0$ holds due to the completeness axiom of IG. The IG-SUM-NORM objective can analogously be derived from Equation (6).

A.1.3 Proof of Equation (9)

The training regularization of the Align method considers the scalar product between input gradients and the original input image. To derive their objective from our framework, we have to set S to the expression given in Equation (9), with the dissimilarity $d_s(\mathbf{x},\mathbf{y}) = \log\left\{1+\exp\left[-\sum_{i\in\dim(\mathbf{x})}(x_i - y_i)\right]\right\}$ and $\mathrm{S}^T = \mathbf{0}$. Equation (5) then becomes as follows.

$$\begin{aligned}
\theta^* &= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} d_s\left[\mathrm{S}(\tilde{\mathbf{x}},f), \mathrm{S}^T(\mathbf{x},f)\right] \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} d_s\left[\cos\left[g^y_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}),\mathbf{x}\right] - \cos\left[g^{\bar{y}}_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}),\mathbf{x}\right]\right] \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} d_s\left[\cos\left[g^y_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}),\mathbf{x}\right] - \cos\left[g^{\bar{y}}_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}),\mathbf{x}\right]\right] \right\} \\
&= \arg\min_\theta \sum_{\mathbf{x}\in D} \left\{ l(\mathbf{x},y,f) + \lambda \cdot \max_{\|\tilde{\mathbf{x}}-\mathbf{x}\|_p<\varepsilon} \log\left\{1+\exp\left[\cos[g^{\bar{y}}_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}),\mathbf{x}] - \cos[g^y_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}),\mathbf{x}]\right]\right\} \right\}
\end{aligned}$$

A.2 Parameters and architectures

Dataset		MNIST	Fashion-MNIST	CIFAR-10	GTSRB	Restr. Imagenet
Architecture		CNN [12]	CNN [12]	ResNet [8]	ResNet [8]	ResNet [8]
AA	*Attack*	PGD				
	Steps	40				
	Rel. stepsize	0.03				
AR	*Attack*	IFIA				
	Explainer	Integrated Gradients with baseline 0				
	d_s	Sum-Top-K				
	Steps	7				
	Rel. stepsize	1.2/7				
	β	1.0				
	k	50	50	100	100	300
ε		0.3	0.1	0.03	0.03	0.01
Number of restarts		3				

Table 2: Evaluation parameters

Dataset		MNIST	Fashion-MNIST	CIFAR-10	GTSRB	Restr. Imagenet
Nat	Optimizer	Adam				
	Epochs	50				
	Batch size	50	50	128	128	32
	LR	0.001	0.001	0.01	0.01	0.01
Adv	*Optimizer*	Adam				
	Epochs	50				
	Batch size	50	50	128	128	32
	LR	0.0001	0.001	0.001	0.001	0.001
	Adv. ratio	0.7				
Align	*Optimizer*	Adam				
	Epochs	50				
	Batch size	50	50	-	-	32
	LR	0.0001	0.0001	-	-	0.0001
	λ	0.5	0.5	-	-	0.5
AAT	*Optimizer*	Adam				
	Epochs	50				
	Batch size	50	50	128	128	32
	LR	0.0001	0.0001	0.0001	0.0001	0.0001
	λ	0.5	1.0	2.0	0.5	1.5
AdvAAT	*Optimizer*	Adam				
	Epochs	50				
	Batch size	50	50	128	128	32
	LR	0.0001	0.0001	0.0001	0.0001	0.0001
	λ	0.5	0.5	0.5	0.2	0.5

Table 3: Training parameters

We conduct experiments on five vision datasets (MNIST, Fashion-MNIST, CIFAR-10, GTSRB and Restricted Imagenet) to compare our attributional robustness method to state of the art algorithms. Each model is implemented in PyTorch v1.3.1 and is trained distributedly on six NVIDIA Tesla V100 GPUs with the PyTorch Distributed Data Parallel wrapper. We fix all seeds to 42. Table 2 contains the evaluation parameters of our experiments, Table 3

the training parameters. We finetune the natural model to train our robust methods. If we do not mention a specific parameter, it is set to the default value in PyTorch v1.3.1. Moreover, the parameters values of IFIA during training are kept as the values during evaluation.

A.3 Initialization methods

We use seven different initialization methods for addressing the dependency of attributional robustness on the initialization. These are detailed in the next paragraphs. If a parameter is not mentioned, it is kept as the default value defined in PyTorch. The training setup is kept constant for each initialization, and corresponds to the setup mentioned in the previous section for the different models.

Init.	Model	NA	AA	IN	CO
PTD	Nat	99.1%	0.0%	0.23	0.20
	Adv	99.0%	93.9%	0.35	0.05
	AAT	98.9%	8.7%	0.39	0.28
CUST	Nat	98.8%	0.0%	0.09	0.03
	Adv	98.8%	88.9%	0.21	0.02
	AAT	98.6%	8.7%	0.30	0.18
UNI	Nat	99.2%	0.0%	0.18	0.13
	Adv	98.9%	93.6%	0.40	0.08
	AAT	98.7%	5.5%	0.33	0.24
HU	Nat	99.2%	0.0%	0.13	0.08
	Adv	99.0%	93.6%	0.12	0.01
	AAT	98.3%	7.2%	0.38	0.24
HN	Nat	99.2%	0.0%	0.10	0.06
	Adv	99.1%	93.6%	0.11	0.01
	AAT	98.5%	4.3%	0.36	0.24
GU	Nat	99.2%	0.0%	0.27	0.19
	Adv	99.0%	93.6%	0.21	0.45
	AAT	98.8%	6.4%	0.38	0.27
GN	Nat	99.2%	0.0%	0.26	0.20
	Adv	99.0%	94.0%	0.37	0.55
	AAT	98.7%	9.1%	0.39	0.28

Table 4: Estimated attributional robustness (IN and CO) for several different initialization methods (Init.). The results are reported for models trained naturally (Nat), adversarially (Adv) as well as with our *AAT* objective on MNIST. The natural and adversarial accuracy is given in the NA and AA columns. While accuracies of the models are similar, their estimated attributional robustness varies significantly throughout the initializations.

PTD. Default PyTorch initialization for linear and convolutional layers. This is the He uniform initialization with $a = \sqrt{5}$ for the weights and a uniform initialization with bounds $\pm b = \pm 1/\sqrt{\texttt{fan\_in}}$ for the bias terms.

CUST. Custom initialization method. Weights are initialized utilizing a zero-centered normal distribution with a standard deviation of 0.1, and biases are initialized to be 0.1, both for linear and convolutional layers.

UNI. Uniform initialization method. Weights and biases are initialized utilizing a uniform distribution with bounds $\pm b = \pm 0.1$ for all layers.

HU. He uniform initialization method. Weights are initialized utilizing the default PyTorch He uniform initialization, biases are set to zero.

HN. He uniform initialization method. Weights are initialized utilizing the default PyTorch He normal initialization, biases are set to zero.

GU. Glorot uniform initialization method. Weights are initialized utilizing the default PyTorch Glorot uniform initialization, biases are set to zero.

GN. Glorot normal initialization method. Weights are initialized utilizing the default PyTorch Glorot normal initialization, biases are set to zero.