Geometric attacks on batch normalization

Amur Ghose[†] Huawei
Apurv Gupta[‡] IBM
Yaoliang Yu[†] UWaterloo
Pascal Poupart[†] UWaterloo

Abstract

Constructing adversarial examples usually requires labels, which provide a loss gradient to construct the example. We show that for batch normalized architectures, intermediate latents that are produced after a batch normalization step suffice to produce adversarial examples using an intermediate loss solely utilizing angular deviations, **without** any label. We motivate our loss through the geometry of batch normed representations and concentration on a known hypersphere. Our losses build on and expand intermediate latent based attacks that usually require labels. The success of our method implies that leakage of intermediate representations may suffice to create a security breach for deployed models, which persist even when the model is transferred to downstream usage. We further show that removal of batch norm weakens our attack significantly, suggesting that batch norm's contribution to adversarial vulnerability may be understood by analyzing such attacks.

1 Introduction

Adversarial examples x_{adv} are commonly defined as data instances which lie only ϵ away in some norm (usually L_∞) from an actual data instance x_{real}. To humans, ϵ is small and x_{real}, x_{adv} share the same label, yet to a classifier neural network, this is not the case. Since their initial discovery [1, 2], adversarial examples have spurred research in both attacking [3, 4] (i.e., generation of adversarial data instances) and defending [5] neural networks against them. Common variants for generating adversarial images rely on gradient steps. One of the earliest effective attacks is Fast Gradient Sign Method (FGSM) [2], which is improved by Projected Gradient Descent (PGD) [6]. We highlight and build on **intermediate-level** attacks, exemplified by members such as Intermediate Level Attack Projection (ILAP) [7] and its variants [8]. In this attack, an initial direction is determined via e.g. FGSM. Then, gradient steps are used to maximally perturb the intermediate hidden representation along the initial direction to find a suitable x_{adv}, unlike directly working with the label. Such a method can often outperform the FGSM itself. ILAP relies on FGSM to set the initial direction, but once the direction is obtained, the layers after the intermediate layer play no role.

With this in mind, we pose a question. Given an unlabeled x, can an adversary - with knowledge only of the intermediate layers upto L of a neural network of total depth $D, D > L$, create adversarial examples that fool the entire neural network ? In our scenario, any labels are absent. For full fairness, we also cannot use the penultimate layer of logits, as a highly accurate model would simply be able to find the true label through an argmax. We show that this is indeed the case. In particular, we adapt the method advanced in ILAP to remove the need for label-based FGSM, and proceed instead with an angular loss based on the geometry of batch normalization. **Our concrete contributions are** :

- To provide a label-free attack that relies on only gradient access upto intermediate layers
- Show that two highly popular architectures - ResNets and EfficientNets - are vulnerable
- Attacks persist even in transfer learning setting when fine-tuning is done downstream

[†] {a3ghose,yaoliang.yu,ppoupart}@uwaterloo.ca, ‡ : apurvgupta1996@gmail.com

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Taken together, our results imply that simply releasing the first few layers of a model publicly, without any label access, may constitute successful ammunition for adversarial attacks, which has obvious security implications. Our attack exploits the radial geometry of batch normalization assuming that the norm of the latent representations concentrates around a constant, allowing us to exploit the geometry so formed. We show that the success of the attack has such a dependence - removing batch normalization and replacing it with alternatives such as Fixup nullifies the attack's success.

2 Previous attacks - FGSM, PGD and intermediate attacks

We describe the Projected Gradient Descent (PGD) attack [6]. We seek x' such that $\|x' - x\| \leq \epsilon$, usually in L_∞ norm. There is a step size α, typically far less than ϵ, used to iteratively update x_i^t as :

$$x_i^t = x_i^{t-1} + \alpha \times \text{sign}(\nabla L)_i^{t-1}$$

With a clamping step on x_i^t that enforces the constraint of $\|x' - x\| \leq \epsilon$, this step after initializing x_i^0 at x_i yields much stronger adversarial examples than FGSM (which is PGD with one step and $\alpha = \epsilon$) at higher t. L is the classification loss e.g. cross-entropy w.r.t. the model outputs and known label.

2.1 Intermediate Level Attack Projection (ILAP)

The above methods - FGSM and PGD - rely only on the input and the final layer's output loss with respect to the true label. However, we can consider a neural network N of depth D as follows :

$$N(x) = N_{i+1,D}(N_{1,i}(x))$$

With 1 based indexing, $N_{j,k}$ denotes the neural sub-network with layers from depth j to k. Let $z_i = N_{1,i}(x)$ be the latent representation at depth i. Consider x_{adv} obtained by any method, and let

$$z_i^{adv} = N_{1,i}(x_{adv}) \; ; \; z_i^{orig} = N_{1,i}(x) \; ; \; \Delta_i^{adv} = z_i^{adv} - z_i^{orig}$$

We then seek to find an alternate adversarial example, x_i^{ILAP} which works with the loss function :

$$L = \langle \Delta_i^{adv}, \Delta_i^{ILAP} \rangle \text{ where } \Delta_i^{ILAP} = z_i^{ILAP} - z_i^{orig} \text{ and } z_i^{ILAP} = N_{1,i}(x_i^{ILAP})$$

The above L can then be plugged into PGD. This loss encourages movement in the latent space in a direction similar to the movement under the label loss. L is in inner product agreement with Δ_i^{adv} and the later iterations use it instead of the true loss. This method is highly transferable - e.g., adversarial examples created using ResNet18 and ILAP succeed more against different models such as DenseNet than corresponding FGSM examples. Often, it improves [7] on FGSM on the source model itself. ILAP demonstrates that latent representations can fuel adversarial attacks, but suffers from needing an initial adversarial example x_{adv} created through the real loss gradient.

3 Angular attacks on hyperspherical representation spaces

Suppose that we had a representation space Z, given by a blackbox function $F, F(X_i) = Z_i$ where for each input pair X_i, X_j, the corresponding Z_i, Z_j satisfied $\|Z_i\| = \|Z_j\|$. Further, the "true" similarity between X_i, X_j could be evaluated based on $\langle Z_i, Z_j \rangle$. In this case, given x_i, we generate a x_i^0 satisfying $\|x_i^0 - x_i\| = \alpha$, using the **initial loss** $L_{init} = -||z_i||$,i.e.

$$x_i^0 = x_i + \alpha \times \text{sign}(\nabla L_{init})$$

We then iteratively update using an angular loss L to grow the initial perturbation :

$$x_i^t = x_i^{t-1} + \alpha \times \text{sign}(\nabla L)^{t-1} \;\; ; L = -\frac{\langle z_i, z_i^t \rangle}{\|z_i\| \|z_i^t\|} \text{ where } z_i = F(x_i) \text{ and } z_i^t = F(x_i^t)$$

Our attack can be seen as equivalent to the ILAP attack, where the initial direction Δ_i^{adv} is arrived at solely by means of an unsupervised radial loss without any label. Our loss then on is **solely** angular and maximizes the angular deviation from the original latent. The intuition behind our attack can be understood as follows. The most dissimilar point to Z_i under $\langle \cdot, \cdot \rangle$ is $-Z_i$. The surrogate loss $-||z_i||$ incentivizes movement along the chord joining these two antipodal points in the interior of the hypersphere, on whose surface the latent representations reside. Moving along this chord flips the label by moving to a different region of the latent space. Note that we can only manipulate x_i^t and cannot guarantee exact radial inward movement, only some decrease of $-||z_i||$, which upon projection, creates implied targeted deviations C. We show this in Figure 1.

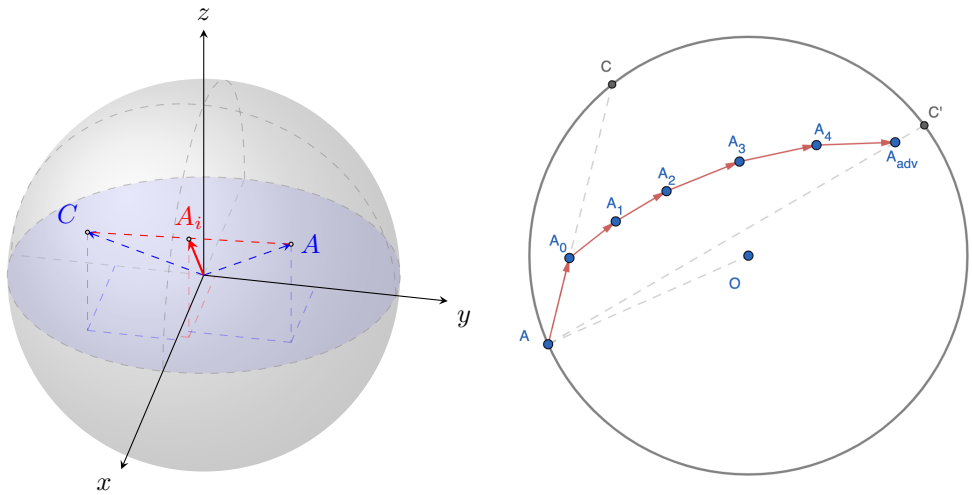


Figure 1: Left: A,C are latent representations of images, while A_i lies on the chord joining them. Right: A_0 results from the initial step towards O, with an implied movement towards C. $A_1, \ldots, A_{adv}$ show the evolution. The implied movement is towards C' - further, angularly, than C from A.

The intuition goes as follows: if we knew A, C (the latents) and the corresponding x_A, x_C we could move x_A towards x_C in latent space even if we did not know the labels to create an adversarial example. In a hyperspherical space, the most dissimilar latent to A, i.e., $-A$ is known "for free".

3.1 Where do hyperspherical latent spaces occur?

Given a minibatch of inputs x_i of dimensions n with mean μ_{ij} and standard deviation σ_{ij} at index j, consider batch normalization [9] or just **BN** as (ignoring any affine shift layers):

$$[\text{BN}(x_i)]_j = \frac{x_{ij} - \mu_{ij}}{\sigma_{ij}} \tag{1}$$

Suppose that the sample means and standard deviations in a pre-BN layer converge to actual sample statistics. After the BN layer, the representation vector Z satisfies indexwise: $\mathbb{E}(Z_i) = 0$ and $\mathbb{E}\left(Z_i^2\right) = 1$. Suppose the Z_i were independent, and the fourth moment bounded. In this case, it is a known result [10] that for a vector Z of high dimensionality d, the random variable $\|Z\|$ strongly concentrates around $\sqrt{d}$ - i.e., nearly all vectors after a BN step would lie near a spherical shell of radius $\sqrt{d}$. For deep linear models and batchnormed Z, we have the result that [11, 12]:

$$E[D_{KL}(Z \| \mathcal{N}(0, I_d))] = O((1-\alpha)^i + \frac{b}{\alpha\sqrt{d}})$$

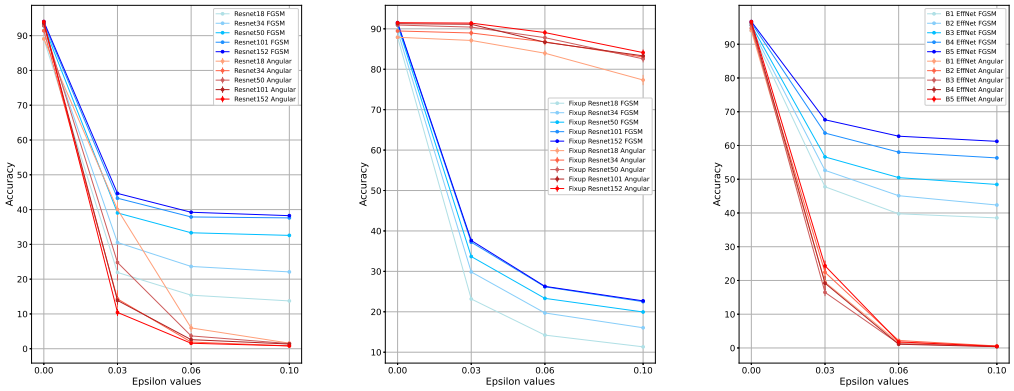
Where Z is the distribution after $N_{1,i}$ i.e., i layers, and α is a constant, and b is the batch size used for training. The LHS indicates the KL divergence between the distribution of the latents to the isotropic Gaussian falls and implies the latent norms being clustered around $\sqrt{d}$. The "gaussianization" of latents goes beyond just deep linear networks [13, 14, 15, 16]. Another work implying that latents lie on hyperspheres is neural collapse [17]. Often, e.g. in neural collapse, the claim is that latents lie on a hypersphere of unknown center and radius. Through BN, we know the center to "reflect" the latent over for free. The above equation indicates deeper layers i.e., increasing i will align with our assumptions - this exponentially drops the KL divergence to a Gaussian, with secondary benefits from the width increase. These theoretical predictions line up with results from ILAP's empirical ablations, which demonstrated that the optimal i to create the latents for the attack occurred in the range 0.6 to 0.8 (with the entire network depth normalized to 0 to 1). We can expect our optimal layers for the latent to lie closer to the end of the network. If we assume our latents lie on a hypersphere, adapting the inner product metric (which ILAP shows to work) to the sphere results in an angular cosine metric. Finally, we can take $N_{1,i}$ (the neural network upto i, ending in a BN layer) as F for our angular attack. We can also combine loss signs from $N_{1,i}, N_{1,i'}$ i.e., work with $\sum_j \text{sign}(\nabla L_j)^{t-1}$ where L_j is from layer j, and use L_{init} for more than 1 initial iterations. These details are in appendix B.

4 Results

Our results show the efficacy of the angular attack in figure 2. Ablations, visualizations, statistical details and further results are in the appendix. We show transfer results on CIFAR-100 [18] in table 1 (results on CIFAR-10 are in the appendix). We run experiments on ResNet [19]-$\{18, 34, 50, 101, 152\}$ and EfficientNet models [20] B-1 to B-5. As a batch norm free architecture, we choose Fixup [21]. We train Fixup resnets on imagenet as alternatives to the batchnormed models for Resnet only because fixup does not generalize to efficientnets. As expected, the attack does **not** work on Fixup Resnets. These models repeatedly stack similar blocks which each possess at least one batch norm layer, allowing extraction of the appropriate latent (in our case, from roughly $\frac{3}{4}$-th the depth of the net - see appendix). All models were ran on Imagenet [22], with standard normalization pre-processing steps. We obtained models from publicly available repositories for PyTorch [23], namely torchvision and EfficientNets-PyTorch. As ϵ for the adversarial attack, we chose $\epsilon = 0.03, 0.06, 0.1$ and $\alpha = 0.01$ over 40 iterations. For the CIFAR-100 case, the model is fine-tuned in last few blocks, and the adversarial example on CIFAR-100 is produced accessing the latent at the end of the frozen layers only, without seeing the changes in last layers. (Details in appendix)

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	0.0	0.0	0.0	0.0	0.0	0.0
Random	60.88	86.19	49.94	78.68	42.41	71.08
FGSM	13.28	36.48	15.88	37.77	13.84	32.8
Angular	2.17	8.57	1.57	7.04	1.34	6.48

Table 1: Accuracies (percentage) on CIFAR-100 transfer, Resnet-18



Original Resnet Fixup Resnet EfficientNet

Figure 2: Accuracies (%) of networks under FGSM (blue) and our Angular attack (red). (Color intensity proportional to network size). Accuracies under angular attack are lower without Fixup.

5 Conclusion

We have shown a powerful label-free attack which only needs **one** data instance and a portion of all the layers available for a network to construct adversarial examples fooling the entire network. It succeeds without knowing the label, other instances, training surrogate models, or having gradient access to the full model, and these adversarial examples - without ever seeing succeeding layers - generalize to the novel dataset transfer case where the model was fine-tuned afterwards in its later layers, allowing attacks on publicly released, privately fine-tuned models. These results are relevant to adversarial robustness and also the study of batch normalization [24, 25, 26] and its drawbacks.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[4] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[5] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[6] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[7] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4733–4742, 2019.

[8] Qizhang Li, Yiwen Guo, and Hao Chen. Yet another intermediate-level attack. In *European Conference on Computer Vision*, pages 241–257. Springer, 2020.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Roman Vershynin. High-dimensional probability, 2019.

[11] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Advances in Neural Information Processing Systems*, 33:18387–18398, 2020.

[12] Hadi Daneshmand, Amir Joudaki, and Francis Bach. Batch normalization orthogonalizes representations in deep random networks. *Advances in Neural Information Processing Systems*, 34:4896–4906, 2021.

[13] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

[14] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[15] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[16] Amur Ghose, Abdullah Rashwan, and Pascal Poupart. Batch norm with entropic regularization turns deterministic autoencoders into generative models. In *Conference on Uncertainty in Artificial Intelligence*, pages 1079–1088. PMLR, 2020.

[17] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[21] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[24] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[25] Philipp Benz, Chaoning Zhang, and In So Kweon. Batch normalization increases adversarial vulnerability: Disentangling usefulness and robustness of model features. 2020.

[26] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*, 2019.

A Appendix

The appendix consists of the following :

- Full descriptions and specifications of models examined and their baseline performance under FGSM, PGD, and random perturbation, as well as no noise.
- Results obtained on these models, under angular attacks, with confidence intervals added.
- Ablations on resnet-18, resnet-34 and resnet-50, showing that depth controls the accuracy drop, and some stronger baselines.
- Transfer accuracy results on CIFAR-10 and CIFAR-100.
- Visualizations of some sample adversarial images produced by our method to contrast against the baseline.

B Loss details

We take the last two layers upto which we have access as i, i' to sum the signs of the angular loss. We cut the α by 2 to adjust for the sum of the signs. Also, before switching over to the angular loss, we can keep using L_{init} for more than the initial iterations. We utilize 20 such iterations before switching, at $\alpha' = \alpha/40$. Hence, in total, our **angular PGD attack** consists of 20 iterations of finding an unsupervised radial direction of moving to the antipodal point in latent space, and 20 iterations of maximizing the angular deviation given the initial movement.

Projection back to the valid PyTorch Tensor space for the adversarial instance (adjusting for normalization layers) proceeds as per normal PGD methods. The ϵ values here are calculated in the normalized Tensor space i.e. after normalizing the $[0, 1]$ tensor Imagenet image with parameters of mean as $[0.485, 0.456, 0.406]$ and standard deviation as $[0.229, 0.224, 0.225]$ channelwise. The projection is such that it respects both the original tensor's range and the ϵ in the normalized space.

Normalization settings for Imagenet for Resnet were kept as-is from the pytorch examples [3] and also as per the Fixup repository[4] as well as for Efficientnet[5].

[3] https://github.com/pytorch/examples/blob/main/imagenet/main.py

[4] https://github.com/hongyi-zhang/Fixup

[5] https://github.com/lukemelas/EfficientNet-PyTorch/tree/master/examples/imagenet

C Original ResNets

We recall that in this class fall 5 models, namely Resnet-18,34,50,101, and 152. In terms of block count, they respectively possess $8, 16, 16, 33, 50$ layers. All radial attacks are performed with the last 2 layers upto which we have access. Access is granted upto blocks $6, 12, 12, 25, 38$ respectively. So for example, on Resnet-18 the radial attack uses the loss signal from blocks $5, 6$ alone to craft the example. **Every value provided in every table is a percentage accuracy metric.**

C.1 Baseline Performance

ResNet type	18	34	50	101	152
Acc@1	69.75	73.31	76.13	77.37	78.31
Acc@5	89.07	91.42	92.86	93.54	94.04

Table 2: Clean accuracies for ResNets

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	FGSM	Random	FGSM	Random	FGSM	Random
Resnet-18	1.95	69.40	1.16	68.58	1.22	67.11
Resnet-34	4.36	73.06	2.95	72.43	3.13	71.23
Resnet-50	8.18	75.79	6.62	74.89	7.10	73.57
Resnet-101	9.93	77.24	8.32	76.79	9.10	75.57
Resnet-152	10.23	77.21	8.76	76.76	9.82	76.28

Table 3: Comparison of FGSM and Random noise, Top-1 accuracy.

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	FGSM	Random	FGSM	Random	FGSM	Random
Resnet-18	21.93	88.94	15.39	88.51	13.74	87.46
Resnet-34	30.54	91.31	23.65	90.96	22.08	90.26
Resnet-50	39.04	92.07	33.33	92.32	32.58	91.58
Resnet-101	43.27	93.52	37.88	93.25	37.61	92.75
Resnet-152	44.65	93.78	39.23	93.48	38.25	92.98

Table 4: Comparison of FGSM and Random noise, Top-5 accuracy.

Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
18	0.006 ± 0.001	3.374 ± 0.04	0.0 ± 0.0	0.484 ± 0.04	0.0 ± 0.0	0.17 ± 0.02
34	0.006 ± 0.001	4.386 ± 0.03	0.006 ± 0.001	0.666 ± 0.04	0.004 ± 0.001	0.288 ± 0.03
50	0.028 ± 0.007	8.076 ± 1.23	0.006 ± 0.002	2.914 ± 0.4	0.0 ± 0.0	1.956 ± 0.3
101	0.032 ± 0.07	9.648 ± 1.76	0.008 ± 0.002	3.902 ± 0.62	0.006 ± 0.002	2.702 ± 0.54
152	0.042 ± 0.08	9.864 ± 1.83	0.014 ± 0.003	4.16 ± 0.73	0.015 ± 0.004	3.253 ± 0.64

Table 5: Comparison of PGD attacks, top-1 and top-5 accuracy, with confidence intervals on different Resnet types.

Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
18	23.416 ± 1.22	40.034 ± 2.12	2.422 ± 0.52	5.980 ± 1.12	0.454 ± 0.11	1.602 ± 0.39
34	7.164 ± 1.28	14.320 ± 2.48	0.592 ± 0.64	1.916 ± 0.32	0.216 ± 0.04	0.814 ± 0.27
50	13.968 ± 2.82	24.796 ± 5.22	1.418 ± 0.22	3.680 ± 0.83	0.432 ± 0.12	1.478 ± 0.26
101	7.012 ± 0.62	13.886 ± 1.23	0.874 ± 0.126	2.606 ± 0.78	0.390 ± 0.122	1.346 ± 0.28
152	5.030 ± 0.4	10.438 ± 1.02	0.534 ± 0.12	1.572 ± 0.265	0.248 ± 0.05	0.808 ± 0.21

Table 6: Comparison of Angular attacks, top-1 and top-5 accuracy, with confidence intervals on different Resnet types.

At higher values of ϵ, angular attack outperforms the FGSM one. The confidence intervals (constructed by bootstrapping, denoted by $\pm$) mark the 5 and 95 percentile confidence intervals and can be used to gauge statistical significance.

D Fixup ResNets

Just as with original Resnets, here we have 5 models, namely Resnet-18,34,50,101, and 152. In terms of block count, they respectively possess $8, 16, 16, 33, 50$ layers. All radial attacks are performed with the last 2 layers upto which we have access. Access is granted upto blocks $6, 12, 12, 25, 38$ respectively. So for example, on FixUpResnet-18 the radial attack uses the loss signal from blocks $5, 6$ alone to craft the example. **Every value provided in every table is a percentage accuracy metric.** In short, everything is performed just as with the original Resnets.

D.1 Baseline Performance

ResNet type	18	34	50	101	152
Acc@1	68.212	70.466	72.938	73.596	73.866
Acc@5	87.910	89.470	90.932	91.273	91.528

Table 7: Clean accuracies for ResNets

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	FGSM	Random	FGSM	Random	FGSM	Random
Resnet-18	2.09	68.18	1.25	67.61	1.19	66.28
Resnet-34	3.73	70.79	2.18	70.39	2.08	69.36
Resnet-50	5.34	73.29	3.63	72.80	3.51	71.61
Resnet-101	6.37	74.41	4.17	73.93	4.17	72.76
Resnet-152	6.59	74.68	4.71	74.21	4.60	73.06

Table 8: Comparison of FGSM and Random noise, Top-1 accuracy.

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	FGSM	Random	FGSM	Random	FGSM	Random
Resnet-18	23.17	87.91	14.23	87.56	11.33	86.75
Resnet-34	29.86	89.52	19.71	89.28	16.05	88.58
Resnet-50	33.67	91.12	23.32	90.83	19.95	90.10
Resnet-101	37.22	91.83	26.18	91.63	22.50	91.00
Resnet-152	37.64	92.09	26.29	91.80	22.70	91.24

Table 9: Comparison of FGSM and Random noise, Top-5 accuracy.

Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
18	0.034 ± 0.003	4.490 ± 0.32	0.014 ± 0.003	0.322 ± 0.05	0.008 ± 0.002	0.058 ± 0.013
34	0.106 ± 0.02	6.524 ± 0.8	0.046 ± 0.008	0.608 ± 0.108	0.030 ± 0.007	0.142 ± 0.03
50	0.090 ± 0.015	5.592 ± 1.26	0.016 ± 0.003	0.266 ± 0.05	0.006 ± 0.002	0.022 ± 0.005
101	0.140 ± 0.03	6.988 ± 1.13	0.018 ± 0.005	0.317 ± 0.04	0.006 ± 0.001	0.048 ± 0.004
152	0.124 ± 0.08	7.474 ± 1.18	0.020 ± 0.003	0.340 ± 0.06	0.004 ± 0.001	0.030 ± 0.005

Table 10: Comparison of PGD attacks, top-1 and top-5 accuracy, with confidence intervals on different Resnet types.

Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
18	66.972 ± 1.78	87.128 ± 0.68	62.856 ± 0.78	83.970 ± 0.68	54.666 ± 1.24	77.336 ± 1.27
34	69.968 ± 0.84	88.972 ± 0.67	67.012 ± 1.27	86.780 ± 0.84	61.868 ± 0.94	83.034 ± 0.53
50	72.914 ± 0.42	90.434 ± 0.62	68.368 ± 1.2	87.818 ± 0.67	61.664 ± 1.2	82.570 ± 0.87
101	73.306 ± 1.07	91.128 ± 0.48	69.424 ± 0.47	88.686 ± 0.34	62.550 ± 0.81	83.318 ± 0.56
152	73.626 ± 0.72	91.412 ± 0.37	70.224 ± 0.84	89.078 ± 0.64	63.700 ± 0.84	84.116 ± 0.74

Table 11: Comparison of Angular attacks, top-1 and top-5 accuracy, with confidence intervals on different Resnet types.

Unlike the original resnets, the angular values do not even come close to the FGSM counterparts. This strongly suggests that removal of batch norm completely fixes this mode of vulnerability.

E EfficientNets

Here we have 5 models, namely EfficientNets B1 to B5. In terms of block count, they respectively possess $23, 23, 26, 32, 39$ layers. All radial attacks are performed with the last 2 layers upto which we have access. Access is granted upto blocks $17, 17, 19, 24, 31$ respectively. So for example, on B1 the radial attack uses the loss signal from blocks $16, 17$ alone to craft the example. **Every value provided in every table is a percentage accuracy metric.** In short, everything is performed just as with the original Resnets.

E.1 Baseline Performance

Efficientnet type	B1	B2	B3	B4	B5
Clean accuracy (top-1)	78.382	79.808	81.532	83.026	83.778
Clean accuracy (top-5)	94.036	94.732	95.646	96.342	96.710

Table 12: Clean accuracies for EfficientNets

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	FGSM	Random	FGSM	Random	FGSM	Random
B1	21.314	77.863	16.860	77.950	14.220	77.296
B2	26.522	79.604	21.926	79.302	20.218	78.564
B3	30.700	81.430	26.526	81.064	25.164	80.678
B4	39.380	82.842	34.820	82.712	33.274	82.344
B5	42.012	83.744	38.002	83.628	36.764	83.424

Table 13: Comparison of FGSM and Random noise, Top-1 accuracy.

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	FGSM	Random	FGSM	Random	FGSM	Random
B1	47.767	93.663	39.758	93.681	38.54	93.582
B2	52.634	94.696	45.114	94.532	42.356	94.222
B3	56.612	95.650	50.478	95.484	48.448	95.282
B4	63.678	96.314	58.024	96.226	56.314	96.116
B5	67.620	96.740	62.750	96.708	61.226	96.626

Table 14: Comparison of FGSM and Random noise, Top-5 accuracy.

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
B1	0.430 ± 0.082	3.318 ± 0.43	0.078 ± 0.012	0.386 ± 0.05	0.017 ± 0.004	0.197 ± 0.022
B2	0.618 ± 0.102	3.092 ± 0.52	0.064 ± 0.006	0.340 ± 0.07	0.012 ± 0.002	0.218 ± 0.04
B3	0.698 ± 0.108	2.482 ± 0.56	0.136 ± 0.02	0.484 ± 0.07	0.044 ± 0.006	0.318 ± 0.07
B4	0.590 ± 0.08	2.112 ± 0.4	0.054 ± 0.009	0.470 ± 0.08	0.018 ± 0.004	0.380 ± 0.05
B5	0.812 ± 0.12	1.728 ± 0.35	0.135 ± 0.02	0.260 ± 0.04	0.073 ± 0.008	0.125 ± 0.03

Table 15: Comparison of PGD attacks, top-1 and top-5 accuracy, with confidence intervals.

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
B1	11.479 ± 1.25	19.501 ± 1.86	0.592 ± 0.063	1.616 ± 0.22	0.156 ± 0.024	0.533 ± 0.083
B2	13.902 ± 1.74	22.421 ± 1.54	0.878 ± 0.122	2.208 ± 0.37	0.153 ± 0.024	0.589 ± 0.028
B3	9.596 ± 0.64	16.476 ± 1.12	0.442 ± 0.07	1.267 ± 0.112	0.130 ± 0.002	0.474 ± 0.028
B4	12.045 ± 1.54	19.192 ± 2.05	0.377 ± 0.042	1.104 ± 0.17	0.091 ± 0.008	0.377 ± 0.042
B5	16.164 ± 2.41	24.350 ± 1.54	0.728 ± 0.062	1.793 ± 0.289	0.104 ± 0.014	0.468 ± 0.047

Table 16: Comparison of Angular attacks, top-1 and top-5 accuracy, with confidence intervals.

With the re-introduction of Batchnorm, the radial attack is again competitive and beats FGSM. We re-perform the same procedure as for Resnets to determine statistical significance against FGSM.

F Ablations on resnets-18,34,50

We first check if the independence structure of latents grows more independent with depth by examining the cross-diagonal average absolute correlation across block groups in Resnet-18. This is seen to decay with depth.

Block count	1	2	3
Average off-diagonal coefficient (absolute)	0.41	0.32	0.19

Table 17: Comparing independence structures on Resnet-18 by comparing absolute value of cross-diagonal correlations, across block groups.

Now, we present results that exhibit the variation in accuracies as the layers being attacked grow further in the network. As a reminder, resnet-18 possesses 8 blocks divided into 4 groups as $[2, 2, 2, 2]$ while resnets 34 and 50 both possess 16 blocks divided as $[3, 4, 6, 3]$. We will use 1-based indexing to denote the layer upto which we have access, and the last 2 layers upto which we have access will provide the entire signal.

Table 18: Ablation on Resnet-18. Last 2 layers of the net granted access to are used to craft the attack.

Access till	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
5	47.55 $\pm$2.2	71.09 $\pm$3.2	12.6 $\pm$1.2	27.8 $\pm$2.1	3.38 $\pm$0.4	8.28 $\pm$0.6
6	22.57 $\pm$1.4	39.82 $\pm$1.2	2.12 $\pm$0.3	6.55 $\pm$0.4	0.60 $\pm$0.1	1.81 $\pm$0.3
7	7.84 $\pm$0.83	16.58 $\pm$0.76	0.18 $\pm$0.03	1.44 $\pm$0.16	0.13 $\pm$0.02	0.54 $\pm$0.12

Table 19: Ablation on Resnet-34. Last 2 layers of the net granted access to are used to craft the attack.

Access till	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
8	54.46 $\pm$1.78	77.35 $\pm$0.68	18.44 $\pm$0.45	32.68 $\pm$0.68	4.95 $\pm$0.86	11.68 $\pm$1.26
9	35.84 $\pm$0.78	59.32 $\pm$0.82	6.32 $\pm$0.56	13.84 $\pm$0.56	1.39 $\pm$0.37	13.84 $\pm$1.78
10	26.22 $\pm$1.56	41.5 $\pm$1.45	1.58 $\pm$0.38	3.88 $\pm$1.02	0.60 $\pm$0.12	1.52 $\pm$0.37
11	13.20 $\pm$0.26	22.68 $\pm$0.86	0.94 $\pm$0.31	2.34 $\pm$0.22	0.34 $\pm$0.06	0.92 $\pm$0.17
12	6.40 $\pm$0.72	12.05 $\pm$1.89	0.45 $\pm$0.12	2.27 $\pm$0.56	0.20 $\pm$0.08	0.65 $\pm$0.22

Table 20: Ablation on Resnet-50. Last 2 layers of the net granted access to are used to craft the attack.

Access till	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
9	53.06 $\pm$1.89	72.74 $\pm$2.76	14.48 $\pm$1.74	25.52 $\pm$1.29	2.16 $\pm$0.48	5.84 $\pm$0.62
10	38.10 $\pm$1.44	54.3 $\pm$0.57	6.44 $\pm$0.42	12.24 $\pm$0.82	1.08 $\pm$0.24	2.24 $\pm$0.45
11	24.55 $\pm$1.22	42.78 $\pm$1.04	2.84 $\pm$0.32	8.44 $\pm$0.18	0.14 $\pm$0.06	2.04 $\pm$0.28
12	8.56 $\pm$1.08	14.08 $\pm$1.78	1.76 $\pm$0.035	3.22 $\pm$0.83	0.55 $\pm$0.12	1.55 $\pm$0.37

G Baselines of targeted attacks and stronger random attacks

Here, we check if providing a latent, generated from an instance of a different label, as the initial direction of ILAP is useful as opposed to using $-||z_i||$. To be clear, in this case, $||z_i - z_j||$ is minimized, where z_j arises from an instance of a different label. Then, the initial deviation is increased via ILAP.

We also construct an alternative random baseline, termed "naive ILAP". In this, an initial random deviation of a random, not necessarily batchnormed layer (we select a random ReLU layer per instance), is increased via ILAP, in the L_2 norm.

We run these baselines for $\epsilon = 0.03, 0.06$ as at 0.1 the values rapidly approach zero for any method. It is apparent that the targeted method is roughly on par with ours and the naive method performs worse. Hence, our method is basically as good as having these points to perturb towards, "for free" just from the manifold structure.

Table 21: Targeted benchmark on all original Resnets

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$	
	Top 1	Top 5	Top 1	Top 5
Resnet-18	25.16	42.58	1.46	5.85
Resnet-34	12.2	21.8	1.9	2.86
Resnet-50	13.55	30.97	0.65	5.16
Resnet-101	5.85	13.66	2.86	6.67
Resnet-152	9.68	16.77	1.29	3.87

Table 22: Naive ILAP on all original Resnets

Net Type	$\epsilon = 0.03$		$\epsilon = 0.06$	
	Top 1	Top 5	Top 1	Top 5
Resnet-18	31.11	52.11	9.92	20.66
Resnet-34	19.57	37.59	6.34	15.15
Resnet-50	25.37	36.59	5.81	14.84
Resnet-101	25.37	37.56	5.71	14.29
Resnet-152	22.58	36.13	3.23	8.39

H Transfer results on CIFAR-10 and CIFAR-100

We show accuracies under the radial attack when Resnet-18,34,and 50 are subjected to it after transfer learning on CIFAR-10 and CIFAR-100. We also show the corresponding baseline drops under FGSM, PGD, random noise and no noise. For Resnets, the final block group (i.e. for example, Resnet-50 has $[3, 4, 6, 3]$ as its block groups, so the last 3) is tuned along with a linear classifier.

Network type	Resnet-18	Resnet-34	Resnet-50
Clean accuracy (top-1)	89.380	89.500	91.210
Clean accuracy (top-5)	99.640	99.790	99.740

Table 23: Clean accuracies on CIFAR-10

Table 24: Results on CIFAR-10, Resnet-50

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	4.230	16.680	4.250	15.360	4.250	15.320
Random	83.930	99.480	71.730	98.520	59.180	83.220
FGSM	52.950	94.770	48.250	92.00	38.300	83.220
Angular	15.020	58.700	14.160	55.310	14.080	54.200

Table 25: Results on CIFAR-10, Resnet-34

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	2.88	13.74	2.81	8.67	2.84	7.97
Random	81.59	99.19	66.7	96.95	52.25	93.30
FGSM	47.00	92.08	42.77	89.58	30.79	83.18
Angular	10.29	54.79	9.4	52.17	8.940	52.100

Table 26: Results on CIFAR-10, Resnet-18

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	4.3	6.3	4.3	5.6	4.28	5.58
Random	84.10	99.25	73.69	98.05	68.22	97.08
FGSM	36.82	88.74	36.52	87.33	30.65	82.2
Angular	10.83	52.38	10.82	49.22	10.51	49.14

Network type	Resnet-18	Resnet-34	Resnet-50
Clean accuracy (top-1)	67.410	69.500	71.170
Clean accuracy (top-5)	90.270	91.140	92.510

Table 27: Clean accuracies on CIFAR-100

Table 28: Results on CIFAR-100, Resnet-50

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	0.16	0.20	0.14	0.14	0.14	0.14
Random	64.42	88.18	53.1	79.76	38.63	66.23
FGSM	29.28	59.02	25.86	51.95	15.01	35.14
Angular	2.32	10.79	2.16	9.72	1.88	9.49

Table 29: Results on CIFAR-100, Resnet-34

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	1.16	1.38	1.13	1.2	1.14	1.22
Random	60.37	85.01	46.68	73.09	34.37	60.34
FGSM	25.97	52.32	25.27	48.92	20.08	40.94
Angular	1.63	7.55	1.440	6.22	1.39	6.16

Table 30: Results on CIFAR-100, Resnet-18

Method	$\epsilon = 0.03$		$\epsilon = 0.06$		$\epsilon = 0.1$	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
PGD	0.0	0.0	0.0	0.0	0.0	0.0
Random	60.88	86.19	49.94	78.68	42.41	71.08
FGSM	13.28	36.48	15.88	37.77	13.84	32.8
Angular	2.17	8.57	1.57	7.04	1.34	6.48

I Visualizations

We show some visualizations of adversarial images created by the radial attack on Imagenet to contrast to FGSM and PGD in table 31. As might be expected, the images sometimes look very similar to the baseline images even at $\epsilon = 0.1$, and sometimes have clear noise over them.

Now, has our method just rediscovered what FGSM and PGD do, or is it independent ? To answer this, we checked the absolute L_1 distance between various images at $\epsilon = 0.1$. The results are as follows :

- Average L_1 distance of 15887.2 between FGSM and angular attacks
- Average L_1 distance of 10282.5 between PGD and angular attacks
- Average L_1 distance of 12866.4 between FGSM and PGD attacks
- Average L_1 distance of 8968.8 between the clean image and the angular attack.

As can be seen, our method is not too similar to either FGSM or PGD and instead is closer to the base image, indicating that PGD and angular attacks diverge in opposite directions from the original image and do not merely copy each other. This is good as we are not just re-creating some other method. Indeed the closeness of our method is more to PGD than to FGSM, which is a good sign, because PGD is the superior attack.

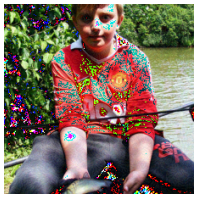
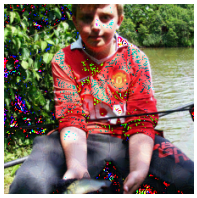
Baseline	FGSM	Angular	PGD
			

Table 31: Table of figures of adversarial images produced under FGSM, PGD, and our Angular method on Imagenet with $\epsilon = 0.1$, with Resnet18 being attacked.