Adversarial Example Decomposition

Horace He [1] Aaron Lou [*1] Qingxuan Jiang [*1] Isay Katsman [*1] Serge Belongie [1] Ser-Nam Lim [2]

Abstract

Research has shown that widely used deep neural networks are vulnerable to carefully crafted adversarial perturbations. Moreover, these adversarial perturbations often transfer across models. We hypothesize that adversarial weakness is composed of three sources of bias: architecture, dataset, and random initialization. We show that one can decompose adversarial examples into an architecture-dependent component, data-dependent component, and noise-dependent component and that these components behave intuitively. For example, noise-dependent components transfer poorly to all other models, while architecture-dependent components transfer better to retrained models with the same architecture. In addition, we demonstrate that these components can be recombined to improve transferability without sacrificing efficacy on the original model.

1. Introduction

Due to the recent successes of neural networks on a wide variety of tasks, they are now being widely applied in the real-world. However, despite their major successes, recent works have shown that in the presence of adversarially perturbed input, they fail catastrophically (Szegedy et al., 2013; Goodfellow et al., 2014). Moreover, Szegedy et al. (2013); Goodfellow et al. (2014) showed that inputs adversarially generated for one model often cause other models to misclassify images as well, a phenomenon commonly called *transferability*.

Our understanding of the causes of transferability is fairly limited. Tramèr et al. (2017) analyzes local similarity of decision boundaries to define a local decision boundary metric that determines how transferable adversarial examples between two models are likely to be. However, many questions are still open. The recent work Wu et al. (2018) hypothesized that adversarial perturbations could be decomposed into initialization-specific and data-dependent components. It is also hypothesized that the data-dependent component is primarily what contributes to transfer. However, Wu et al. (2018) provides neither theoretical nor empirical evidence to justify this hypothesis.

Our work aims to examine this hypothesis in greater detail. We first augment the previous hypothesis to provide decomposition into three parts: architecture-dependent, data-dependent, and noise-dependent components. Given this framework, our contributions are as follows:

- We propose a method for decomposing adversarial perturbations into noise-dependent and noise-reduced components.
- We also present a method to further decompose the noise-reduced component into architecture-dependent and data-dependent components.
- Extensive experiments are conducted on CIFAR-10 (Krizhevsky, 2009) using various architectures to show the above two decompositions have the desired properties. Results from an ablation study are given to show the significance of the nontrivial choices made in our methodology.

2. Motivation and Approach

Motivated by the reviewers' comments on Wu et al. (2018), we seek to provide further evidence that an adversarial example can be decomposed into model-dependent and data-dependent portions. First, we augment our hypothesis to claim that an adversarial perturbation can be decomposed into architecture-dependent, data-dependent, and noise-dependent components. We note that it is clear that these are the only things that could contribute in some way to the adversarial example. An intuition behind why noise-dependent components exist and would not transfer despite working on the original dataset is shown in Figure 2.

Not drawn explicitly in the figure is the architecture-dependent component. As neural networks induce biases in

[*]Equal contribution [1]Department of Computer Science, Cornell University, Ithaca, NY, USA [2]Facebook AI, New York, New York, USA. Correspondence to: Horace He <hh498@cornell.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

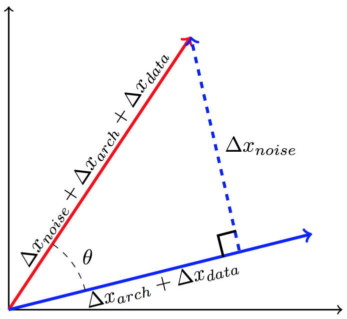
$\Delta x_{noise} + \Delta x_{arch} + \Delta x_{data}$
Δx_{noise}
θ
$\Delta x_{arch} + \Delta x_{data}$

Figure 1. **Noise Vector Decomposition**. Δx_{noise}, Δx_{data}, Δx_{arch} are as defined in Section 2.1. Note the orthogonality of Δx_{noise} and $\Delta x_{arch} + \Delta x_{data}$; though this is only an assumption, it is justified to be reasonable experimentally in the ablation study of Section 3.3.

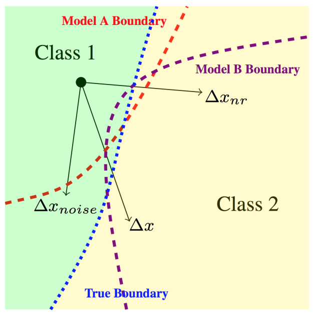

Figure 2. **Varying Decision Boundaries**. In the above figure, Δx is the adversarial perturbation, and Δx_{noise}, Δx_{nr} are as defined in Section 2.1.

the decision boundary, and specific network architectures induce specific biases, we would expect that an adversarial example could exploit these biases across all models with the same architecture.

2.1. Notation

We denote $\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k\}$ to be the set of model architectures. Let $\mathcal{M}^i = \{\mathcal{M}^i_\alpha\}$ to be a set of fully trained models of architecture $\mathcal{A}_i$ initialized with random noise. The superscript will be omitted when architecture is clear.

We define an attack $A(x, y, \mathcal{M}^i_j, \mathcal{L}, \Delta x_0) = \Delta x$, where x is an image, y its corresponding label, $\mathcal{M}^i_j$ is a neural network model as defined above, $\mathcal{L}$ is a loss function, Δx_0 the initial perturbation of x, and Δx a perturbation of x such that $\mathcal{L}(\mathcal{M}^i_j(x + \Delta x), y)$ is maximal.

For fixed architecture $\mathcal{A}_i$, model $\mathcal{M}^i_j$, and attack A, we denote $\Delta x_{noise}, \Delta x_{arch}, \Delta x_{data}$ to be the three components of Δx introduced in previous sections. Let $\Delta x_{noise\ reduced} = \Delta x_{arch} + \Delta x_{data}$; we will use the short hand Δx_{nr}.

Let $P_{x_1}(x_2)$ denotes the projection of vector x_2 onto vector x_1. Let $\widehat{x} = \frac{x}{||x||}$ be the unit vector with same direction as x.

2.2. Δx_{noise} and Δx_{nr} Decomposition

Description: We fix our architecture $\mathcal{A}_0$ and have $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ as our set of trained models. Set $\mathcal{L}$ to be the cross-entropy loss and let $\Delta x = A(x, y, \mathcal{M}_1, \mathcal{L}, \mathbf{0})$ be the generated adversarial perturbation for $\mathcal{M}_1$.

Proposition: Δx can be decomposed into $\Delta x_{noise} + \Delta x_{nr}$ such that the attack Δx_{noise} is effective on $\mathcal{M}_1$ but transfers poorly to $\mathcal{M}_2, \ldots, \mathcal{M}_n$, while Δx_{nr} transfers well on all models.

The equations for computing Δx_{nr} and Δx_{noise} are given in Equation 1 (see Appendix C for justification). The technique is illustrated in Figure 1.

$$\Delta x_{nr} \approx A(x, y, \frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j, \mathcal{L}, \mathbf{0}) \tag{1}$$
$$\Delta x_{noise} \approx \Delta x - P_{\Delta x_{nr}}(\Delta x)$$

2.3. Δx_{arch} and Δx_{data} Decomposition

Description: We reuse notation from the above section, except that we now consider a set of different architectures $\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1, \ldots \mathcal{A}_k\}$

Proposition: Δx_{nr} can be composed into $\Delta x_{arch} + \Delta x_{data}$ such that the attack Δx_{arch} is effective on $\mathcal{A}_0$ but transfers poorly to $\mathcal{A}_1, \cdots, \mathcal{A}_k$, while Δx_{data} transfers well on all models.

The equations for computing Δx_{arch} and Δx_{data} are given in Equation 2, in which we set Δx_{nr} to be the noise reduced perturbation generated on $\mathcal{A}_0$ (see Appendix C for justification). We approximate the expectation for Δx_{data} by averaging across architectures.

$$\Delta x_{data} \approx$$
$$\mathbb{E}_{\mathcal{A}_i \in \mathcal{A}}\left[A(x, y, \frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}^i_j, \mathcal{L}, \Delta x_{nr})\right] \tag{2}$$
$$\Delta x_{arch} \approx \Delta x_{nr} - P_{\Delta x_{data}}(\Delta x_{nr})$$

3. Results

We empirically verify the approaches given in the motivation above and show that the isolated noise and architecture-dependent perturbations show the desired properties. Unless stated otherwise, all perturbations are generated on

Table 1. Δx_{noise} decomposition (ResNet18)

Δ	$\mathcal{M}_{orig}$	$\{\mathcal{M}_{avg}\}$	$\{\mathcal{M}_{test}\}$
Δx	68.3%	45.6%	46.7%
Δx_{nr}	63.7%	61.9%	59.5%
Δx_{noise}	60.2%	19.8 %	20.3%

Table 2. Δx_{noise} decomposition (DenseNet121)

Δ	$\mathcal{M}_{orig}$	$\{\mathcal{M}_{avg}\}$	$\{\mathcal{M}_{test}\}$
Δx	70.0 %	47.1%	49.8%
Δx_{nr}	64.3%	65.3%	66.6%
Δx_{noise}	64.9%	27.1%	29.6%

Table 2. All numbers reported are fooling ratios. Observe that Δx_{noise} exhibits exceptionally low transferability.

CIFAR-10 (Krizhevsky, 2009) (original images rescaled to $[-1, 1]$) using iFGSM (Kurakin et al., 2016) with 10 iterations, distance metric L_∞, and $\epsilon = 0.03$. All experiments are run on the first 2000 CIFAR-10 test images. In addition, all models are trained for only 10 epochs due to computational constraints. All percentages reported are fooling ratios (Moosavi-Dezfooli et al., 2017). For results with other settings, check Appendix A.

3.1. Δx_{noise} and Δx_{nr} Decomposition

We start off with a set of 10 retrained ResNet18 (He et al., 2016) models $\{\mathcal{M}_i\}$. We attack the first ResNet18 model $\mathcal{M}_1 (= \mathcal{M}_{orig})$ to get a perturbation Δx for a given x. We then follow the process outlined in Equation 1 to obtain Δx_{nr} from the other 9 retrained ResNet18 models $\{\mathcal{M}_{i>1}\} (= \{\mathcal{M}_{avg}\})$. We then test on an untouched set of 5 retrained ResNet18 models $\{\mathcal{M}_{test}\}$. We also do the same process for DenseNet121 (Huang et al., 2017) instead of ResNet18 and report their respective results in Tables 1 and 2.

We note that Δx_{noise} achieves a far lower transfer rate than either Δx_{nr} or Δx while still maintaining relatively high error rate on the original model, providing evidence for the success of this decomposition. To the best of our knowledge, this is the first methodology that is able to construct adversarial examples with especially low transferability. Although this is of low practical use, this is theoretically interesting. We note that although we attempt to generate Δx_{nr} by multi-fooling across 9 retrained models, reducing noise in high dimensions is difficult, so we are unable to achieve a perfect decomposition of Δx_{noise}. Ablation studies in Appendix B suggest that we may be able to achieve a better decomposition with a larger set of retrained models.

3.1.1. Recombining components

As the components $\Delta\widehat{x}_{noise}$ and $\Delta\widehat{x}_{nr}$ are linearly independent unit vectors, and by definition, Δx is in the span of these vectors, we can find unique scalars a and b such that $a \cdot \Delta\widehat{x}_{noise} + b \cdot \Delta\widehat{x}_{nr} = \Delta x$. Experimentally, we find that under our setting, $a \approx 1.319$ and $b \approx 0.386$. We note that for our original perturbation, this is perhaps an undue amount of focus paid to the noise-specific perturbation. We can now try setting a and b to different ratios, which correspond to how much we wish to emphasize attacking the original model vs. transferability. As we are now able to set an arbitrarily high a and b, allowing us to saturate the epsilon constraints, we sign maximize (ie: $sign(x) \cdot \epsilon$), as motivated in Goodfellow et al. (2014)) to level the playing field. The results in Table 3 show the results of performing these experiments on ResNet18. We find that we are able to generate perturbations that perform equivalently with Δx on $\mathcal{M}_{orig}$, while performing substantially better when transferring to $\{\mathcal{M}_{avg}\}$ and $\mathcal{M}_{test}$.

Table 3. Linear Combinations of $\Delta\widehat{x}_{noise}$ and $\Delta\widehat{x}_{nr}$

$b : a$	$\mathcal{M}_{orig}$	$\{\mathcal{M}_{avg}\}$	$\mathcal{M}_{test}$
Δx_{nr}	65.8%	63.6%	65.1%
2:1	68.5%	63.7%	65.2%
1.5:1	69.4%	61.2%	62.8%
1:1	69.8%	56.0%	56.4%
1:2	70.0%	53.1%	53.5%
Δx	69.8%	51.0%	51.0%

Table 3. All numbers reported are fooling ratios. Observe that as you increase the ratio $b : a$ we obtain better transferability with lowered effectiveness on $\mathcal{M}_{orig}$. Also note that we are able to construct perturbations that are strictly superior to either Δx or Δx_{nr}.

3.2. Δx_{arch} and Δx_{data} Decomposition

To evaluate decomposition into architecture and data-specific components, we consider the four architectures ResNet18 (He et al., 2016), GoogLeNet (Szegedy et al., 2015), DenseNet121 (Huang et al., 2017), and SENet18 (Hu et al., 2017). Results are given in Table 4. In each experiment we first fix a source architecture $\mathcal{A}_i$ and generate Δx_{nr} by attacking 4 retrained copies of $\mathcal{A}_i$, denoted as $\{\mathcal{M}_{source}\}$. We then generate Δx_{data} by attacking four copies of each $\mathcal{A}_{j\neq i}$ for twelve models total. We then test on another 4 retrained copies of $\mathcal{A}_i$ called $\{\mathcal{M}'_{source}\}$ as well as $\{\mathcal{M}'_{other}\}$, consisting of four copies of each of the other three architectures $\{\mathcal{A}_{j\neq i}\}$. We see that for all four models, Δx_{arch} obtains significantly higher er-

Table 4. Δx_{arch} decomposition

source	Δ	$\{\mathcal{M}_{source}\}$	$\{\mathcal{M}_{other}\}$	$\{\mathcal{M}'_{source}\}$	$\{\mathcal{M}'_{other}\}$
ResNet18	Δx_{nr}	60.9%	50.7%	59.4%	50.7%
	Δx_{data}	54.6%	61.4%	54.8%	60.8%
	Δx_{arch}	52.4%	26.7%	36.9%	30.3%
DenseNet121	Δx_{nr}	62.8%	46.2%	62.9%	47.1%
	Δx_{data}	58.4%	58.3%	57.2%	55.7%
	Δx_{arch}	54.1%	24.4%	43.1%	26.0%
GoogLeNet	Δx_{nr}	65.3%	41.9%	65.7%	41.9%
	Δx_{data}	59.5%	59.2%	59.5%	58.3%
	Δx_{arch}	57.9%	22.8%	44.8%	26.2%
SENet18	Δx_{nr}	53.8%	48.4%	53.2%	49.0%
	Δx_{data}	55.7%	64.5%	54.8%	63.8%
	Δx_{arch}	47.1%	28.1%	38.6%	29.8%

Table 4. All numbers reported are fooling ratios. Note that for all architectures, the adversarial decomposition holds. Namely, Δx_{arch} is more transferable to its specific architecture than to others, whereas Δx_{data} is equally transferable across architectures.

ror rate on $\{\mathcal{M}'_{source}\}$ than on $\{\mathcal{M}'_{other}\}$. In addition, the relative error between $\{\mathcal{M}'_{source}\}$ and $\{\mathcal{M}'_{other}\}$ for Δx_{arch} are close to the relative error between $\{\mathcal{M}_{source}\}$ and $\{\mathcal{M}_{other}\}$ for Δx_{nr} when averaged across models, supporting the success of our decomposition.

3.3. Ablation

Orthogonality We assume that Δx_{noise}, Δx_{arch}, and Δx_{data} terms are orthogonal. We note that if these vectors had no relation to each other, then due to the properties of high dimensional space, they are approximately orthogonal with very high probability.

We vary orthogonality by modifying the method in Section 2.2 to generate Δx_{noise} with $\Delta x - \alpha P_{\Delta x_{nr}}(\Delta x)$. When $\alpha = 1$, we recover the original algorithm, and when $\alpha = 0$, $\Delta x_{noise} = \Delta x$. Experimentally varying the orthogonality of Δx_{noise} and Δx_{nr} produces the results in Table 5; note that we achieve the greatest difference in efficacy between the original model and transferred models when they are near-orthogonal, suggesting that the assumption we made is reasonable.

However, it is not true that orthogonal components achieve the best isolation (given by the fact that the peak difference seems to be at $\alpha = 1.2$). This suggests that our current method of decomposition may simply be an approximation for the true components, and that a more nuanced method may be necessary for better isolation.

Number of Models We find that the higher the number of models we use to approximate Δx_{nr}, the more successfully we are able to isolate Δx_{noise}. Check Appendix B for full results.

Table 5. Varying α

α	$\mathcal{M}_{orig}$	$\{\mathcal{M}_{avg}\}$	Difference
Δx	68.3%	45.6%	22.7
Δx_{nr}	63.7%	61.9%	1.8
0.1	66.7%	42.2%	24.5
0.5	63.4%	29.1%	34.3
0.8	59.7%	21.4%	38.3
1.0	52.7%	16.6%	36.1
1.2	51.9%	11.3%	40.6
1.5	42.9%	9.4%	33.5
2.0	33.5%	7.2%	26.3

Table 5. All percentages reported are fooling ratios. Note that the $\alpha = 1.2$ setting is what produces maximal difference, which is slightly different from the assumed orthogonality ($\alpha = 1.0$).

4. Conclusion

We demonstrate that it is possible to decompose adversarial perturbations into noise-dependent and data-dependent components, a hypothesis reviewers thought was interesting but unsupported in (Wu et al., 2018). We go even further by decomposing an adversarial perturbation into model related, data related, and noise related perturbations. A major contribution here is a new method of analyzing adversarial examples; this creates many potential future directions for research. One interesting direction would be extending these decompositions to universal perturbations (Moosavi-Dezfooli et al., 2017; Poursaeed et al., 2017) and thus removing the dependence on individual data points. Another avenue to explore is analyzing various attacks and defenses and how they interplay with these various components.

References

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94, 2017.

Poursaeed, O., Katsman, I., Gao, B., and Belongie, S. J. Generative adversarial perturbations. *CoRR*, abs/1712.02328, 2017.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. D. Ensemble adversarial training: Attacks and defenses. *CoRR*, abs/1705.07204, 2017.

Wu, L., Zhu, Z., Tai, C., and E, W. Enhancing the transferability of adversarial examples with noise reduced gradient, 2018. URL `https://openreview.net/forum?id=ryvxcPeAb`.

Appendix

A. Different attack settings

To show that our decomposition is effective across a variety of attack settings, we perform the experiment of Section 3.1 with three different iFGSM settings corresponding to $\epsilon = 0.01, 0.03, 0.06$. Results are shown in Table 6.

Table 6. Varying ϵ

ϵ	Δ	$\{\mathcal{M}_{orig}\}$	$\{\mathcal{M}_{avg}\}$	$\{\mathcal{M}_{test}\}$
.01	Δx	39.0%	16.4%	14.4%
	Δx_{nr}	25.1%	28.4%	22.2%
	Δx_{noise}	26.6%	06.2%	05.3%
.03	Δx	68.3%	45.6%	46.7%
	Δx_{nr}	63.7%	61.9%	59.5%
	Δx_{noise}	60.2%	19.8%	20.3%
.06	Δx	81.2%	69.7%	73.6%
	Δx_{nr}	81.1%	80.5%	85.8%
	Δx_{noise}	77.7%	39.4%	40.0%

B. Varying number of models/iterations

We investigate the effectiveness of the Section 3.1 decomposition as we vary hyper-parameters. Results for increasing iFGSM iterations in Table 7 and results for increasing the results for increasing the number of models are give in Table 8.

Table 7. Varying number of iterations used for iFGSM

# of iters	Δ	$\{\mathcal{M}_{orig}\}$	$\{\mathcal{M}_{avg}\}$
5	Δx	65.2%	43.4%
	Δx_{nr}	58.8%	58.8%
	Δx_{noise}	55.5%	20.6%
10	Δx	68.3%	46.7%
	Δx_{nr}	63.7%	61.9%
	Δx_{noise}	60.2%	19.8%
100	Δx	72.9%	48.6%
	Δx_{nr}	67.3%	65.2%
	Δx_{noise}	60.3%	18.7%

C. Justification of Equations

Justification of Equations in 3.1

Recall that the equations are given by

Table 8. Varying number of models used to approximate Δx_{nr}

# of models	Δ	$\{\mathcal{M}_{orig}\}$	$\{\mathcal{M}_{avg}\}$	$\{\mathcal{M}_{test}\}$
3	Δx	69.4%	46.6%	45.6%
	Δx_{nr}	57.6%	62.1%	51.9%
	Δx_{noise}	60.1%	24.9%	29.2%
5	Δx	68.4%	47.0%	44.8%
	Δx_{nr}	60.1%	62.0%	55.2%
	Δx_{noise}	57.5%	22.4%	24.6%
10	Δx	68.3%	45.6%	46.7%
	Δx_{nr}	63.7%	61.9%	59.5%
	Δx_{noise}	60.2%	19.8%	20.3%

$$\Delta x_{nr} \approx A(x, y, \frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j, \mathcal{L}, 0)$$
$$\Delta x_{noise} \approx \Delta x - P_{\Delta x_{nr}}(\Delta x)$$

We assume that the expected value of our noise term Δx_{noise} is 0 over all random noise. This is motivated because the random noise i at initialization is a Gaussian distribution centered at 0, and it is reasonable to assume that the model distribution and the noise distribution follows a similar pattern.

Letting $\Delta x^j = A(x, y, \mathcal{M}_j, \mathcal{L}, 0)$ over all random initialization i, we claim that $\mathbb{E}_j[\Delta x^j] = \Delta x_{arch} + \Delta x_{data}$. Since Δx_{arch} and Δx_{data} are noise independent, which means that

$$\Delta x^j = \Delta x^j_{noise} + \Delta x_{arch} + \Delta x_{data}$$

where Δx^j_{noise} is the noise component corresponding with the noise of model $\mathcal{M}_j$. Therefore, it follows that

$$\begin{aligned}\mathbb{E}_j[\Delta x^j] &= \Delta x_{arch} + \Delta x_{data} + \mathbb{E}_j[\Delta x^j_{noise}] \\ &= \Delta x_{arch} + \Delta x_{data} = \Delta x_{nr}\end{aligned}$$

By the law of large numbers, it follows that $\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^{n}\Delta x^j = \Delta x_{nr}$. Therefore, we note that, for sufficiently large n, it follows that

$$\frac{1}{n}\sum_{j=1}^{n}\Delta x^j \approx \Delta x_{nr}$$

We see that, since the cross entropy loss $\mathcal{L}$ is additive and the attack A that we examine are first order differentiation methods, we have

$$\mathcal{L}(\frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j(x),y) = \frac{1}{n-1}\sum_{j=2}^{n}\mathcal{L}(\mathcal{M}_j(x),y)$$
$$\implies A(x,y,\frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j,\mathcal{L},0)$$
$$= \frac{1}{n-1}\sum_{j=2}^{n}A(x,y,\mathcal{M}_j,\mathcal{L},0) \approx \Delta x_{nr}$$

To prove the other claim, we have already shown through empirical results and an intuition that Δx_{noise} and Δx_{nr} are linearly independent that Δx_{noise} and Δx_{nr} are very close to orthogonal and compose Δx. Therefore, it follows that we can take the use the projection of Δx_{nr} implies that

$$\Delta x_{noise} + P_{\Delta x_{nr}}(\Delta x) \approx \Delta x$$
$$\implies \Delta x_{noise} \approx \Delta x - P_{\Delta x_{nr}}(\Delta x)$$

up to a scaling constant.

Justification of Equations in 3.2

Recall that the equations are, given Δ_{nr} generated on $\mathcal{A}_0$,

$$\Delta x_{data} \approx \mathbb{E}_{\mathcal{A}_i\in\mathcal{A}}\left[A(x,y,\frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j^i,\mathcal{L},\Delta x_{nr})\right]$$
$$\Delta x_{arch} \approx \Delta x_{nr} - P_{\Delta x_{data}}(\Delta x_{nr})$$

We make two core assumptions:

- The value of $\Delta\mathbb{E}_{\mathcal{A}}[x_{arch}] = 0$. This is a reasonable assumption since our generated architectures $\mathcal{A}$ should produce roughly symmetric error vectors x_{arch}.
- $A(x,y,\frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M},\mathcal{L},\Delta x')$ is equivalent $A(x,y,\mathcal{M},\mathcal{L},0)$ in the sense that the former produces a noised reduce gradient closer to $\Delta x'$. This is reasonable because the space of there are many adversarial perturbations (different directions) and changing our start location won't cripple our search space. Furthermore, we use this to generate a Δ_{nr} close to $\Delta x'$.

We claim that $\mathbb{E}_{\mathcal{A}}[\Delta x_{nr}] = \Delta x_{data}$ where we take Δx_{nr} over architecture $\mathcal{A}$. To see this, we note that

$$\mathbb{E}_{\mathcal{A}}[\Delta x_{nr}] = \mathbb{E}_{\mathcal{A}}[\Delta x_{arch}] + \mathbb{E}_{\mathcal{A}}[\Delta x_{data}] = \mathbb{E}_{\mathcal{A}}[\Delta x_{data}]$$

and so again we can approximate it with $\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\Delta x_{nr}^i = \Delta x_{data}$ where Δx_{nr}^i is the Δx_{nr} component generated for model $\mathcal{A}_i$. For sufficiently large n, it follows that

$$\frac{1}{n}\sum_{i=1}^{n}\Delta x_{nr}^i = \Delta x_{data}$$

Therefore we have

$$\Delta x_{data} = \mathbb{E}_{\mathcal{A}_i\in\mathcal{A}}[\Delta x_{nr}^i] = \mathbb{E}_{\mathcal{A}_i\in\mathcal{A}}\mathbb{E}_j[(x,y,\mathcal{M}_j^i,\mathcal{L},0)]$$
$$\approx \mathbb{E}_{\mathcal{A}_i\in\mathcal{A}}\left[A(x,y,\frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j^i,\mathcal{L},0)\right]$$

and by our assumption this is roughly equivalent to

$$\mathbb{E}_{\mathcal{A}_i\in\mathcal{A}}\left[A(x,y,\frac{1}{n-1}\sum_{j=2}^{n}\mathcal{M}_j^i,\mathcal{L},\Delta x_{nr})\right]$$

as desired. To prove the other claim, we use an analogous argument to the one above as we have shown that Δx_{arch} and Δx_{data} are orthogonal and applying the same projection technique yields

$$\Delta x_{data} + P_{\Delta x_{data}}(\Delta x_{nr}) \approx \Delta x_{nr}$$
$$\implies \Delta x_{arch} \approx \Delta x - P_{\Delta x_{data}}(\Delta x_{nr})$$

up to a scaling constant.