Designing Adversarially Resilient Classifiers using Resilient Feature Engineering

Kevin Eykholt, Atul Prakash

University of Michigan

Abstract

We provide a methodology, resilient feature engineering, for creating adversarially resilient classifiers. According to existing work, adversarial attacks identify weakly correlated or non-predictive features learned by the classifier during training and design the adversarial noise to utilize these features. Therefore, highly predictive features should be used first during classification in order to determine the set of possible output labels. Our methodology focuses the problem of designing resilient classifiers into a problem of designing resilient feature extractors for these highly predictive features. We provide two theorems, which support our methodology. The Serial Composition Resilience and Parallel Composition Resilience theorems show that the output of adversarially resilient feature extractors can be combined to create an equally resilient classifier. Based on our theoretical results, we outline the design of an adversarially resilient classifier.

1 Introduction

Deep neural networks and other machine learning algorithms have seen large success being used in industry for numerous tasks. However, the rise in interest towards these algorithms has also raised concerns in the community as to their trustworthiness. Namely, these algorithms are vulnerable to *adversarial inputs*. These are inputs that, to a human, appear no different from an existing input the algorithm can correctly label, but are slightly modified such that the algorithm assigns a different label to them. These types of inputs can harm the widespread adoption of machine learning algorithms, especially in safety critical tasks. For example, in autonomous driving, machine learning is used to perform object recognition, thus it is critical that the algorithm is robust to such adversarial inputs.

Previous research in adversarial machine learning limited attacks to the digital domain. Adversarial modifica-

tions would be directly applied to the input of the algorithm (*e.g.,* an image file). Recent work, however, has demonstrated that it is possible to craft adversarial inputs in the physical domain [7, 9]. Thus, it is critically important to design defensive measures that can reduce or remove the risk of adversarial examples. In the digital domain, many defensive solutions have been proposed, but most strategies are limited. As pointed out by previous work, most defensive solutions only work on a static adversary or under very specific conditions [1, 2, 4, 5].

Current literature identifies that adversarial examples are possible due to machine learning models relying on features which are weakly correlated with the correct output label [24] or assigning non-zero weights to irrelevant features [6]. Thus, classifiers should first use the highly predictive features to narrow down the set of possible output labels. These features are likely to be highly selective and in certain domains, may be easily identifiable. For example, in road sign classification, knowing the shape of the sign can be enough to uniquely identify the sign or sign class (*e.g.,* Octagon: Stop sign; Diamond: Warning sign).

In our work, we propose *resilient feature engineering*, which uses highly predictive features to create adversarially resilient machine learning classifiers. First, predictive features for the classification task are identified. For each identified feature, a feature extractor is constructed. The output of each feature extractor is provided to a classification algorithm, which outputs one or more possible class labels based on the task.

The key idea is to create these feature extractors so they are **resilient** to adversarial attacks. That is to say, if only a certain amount of input distortion is allowed, the feature extractors cannot be adversarially attacked. We provide two theorems, the Serial Composition Resilience and Parallel Composition Resilience theorems, that state if such conditions can be maintained, then the classifier is also adversarially resilient. We discuss a basic design for an adversarially resilient classifier based on our theo-

retical results.

2 Related Works

The existence of adversarial examples was first studied by Szegedy *et al.* [22]. In their work, they found that it is possible to arbitrarily manipulate the output of a neural network using imperceptible, non-random perturbations to the input, denoting these perturbed inputs as *adversarial examples*. Since then, numerous works have appeared proposing new algorithms for generating adversarial examples in both the digital domain [4, 11, 16, 17] and the physical domain [3, 7–9, 14].

In an effort to mitigate the effect of adversarial examples, there is a push to design defensive measures against adversarial examples. Research in this area can be divided into defensive methods [11, 18, 21, 23], which improve a model's resistance to adversarial examples, and detection solutions [10, 13, 15], which use statistical properties or other information to identify adversarial examples from natural examples.

The main issue with many of the proposed strategies is that many of them do not provide provable security guarantees assuming a certain threat model. Rather, they implement a technique, such as gradient masking, PCA analysis, or input normalization, and show that in testing the proposed strategy is resistant to many adversarial attacks. However, when exposed to an adaptive adversary that modifies the attack based on the defense, the defense fails [1, 2, 4, 5].

Of interest is research that provides provable security guarantees against adversarial attacks. Hein and Andriushchenko provided the first formal guarantees on the adversarial robustness of a classifier [12]. More specifically, given a specific instance, they demonstrate a lower-bound on the l_2 norm of the input manipulation required to change the output of the classifier. Similarly, Raghunathan *et al.* also establish lower bounds on adversarial perturbations, but do so for the l_∞ norm [19].

Concurrently, Wong and Kolter also provide an adversarial robust training methodology for arbitrarily deep ReLU networks given a certain norm-bound on adversarial perturbations [25]. In their work, they establish a convex space around each input and ensure that the classification decision for correctly labelled inputs does not change within the space. Then, optimize the convex space on the training point with the highest training loss. Finally, Sinha *et al.* also provide methods to guarantee adversarial robustness, but their method uses defined distributional Wasserstein distances rather than the norm of the adversarial perturbations [20].

Compared to previous work, our work provides a more general methodology for generating adversarially resilient classifiers. Our approach transforms the problem of defending against adversarial attacks into designing resilient feature extractors. Furthermore, we can leverage previous results from other works to inform the design of resilient feature extractors, rather than attempting to apply the results of those works to an entire classifier.

3 Formal Background Definitions

In this section, we formally define classifiers and adversarial inputs as well as some terms related to them. These definitions are necessary before discussing resilient feature engineering.

3.1 Classifiers

Informally, a classifier is given a set of inputs and generates a set of outputs. Each output represents the class label for a given input where the class label belongs to a set of class labels. For example, a image classifier may be provided a picture of an animal and is expected to output the type of animal that is in the picture.

We define x, the input to a classifier, as a set of observations. An **observation** is a categorical, ordinal, integer valued, or real valued property. We define y, the output of a classifier, as a **label** or **category** belonging to a set of labels Y. Typically, y is a single value, but it may be a tuple.

With these two preliminary definitions, we say that a **classifier** F is a function that maps a set of observations x to a label y in the set of labels Y. That is to say:

$$F(x) = y$$

In order to determine the correctness of a classifier, there must be some source that generates the correct label for a given input. Given a classifier F, O_F denotes the **oracle** function for the classifier F. For an input x, $O_F(x) = l$ where $l \in Y \cup non$. *non* is a label that is not contained in Y. If $O_F(x) = non$, then we say that x is a **nonsense input**. If $O_F(x) \neq non$, then $l \in Y$ and we say that x is a **natural input** and is part of the **natural domain** of F, denoted by N_F.

A nonsense input is an input in which no label in Y would be appropriate. That is to say, the input is not suitable for the classification task. For example, if we treat a human labeller as the oracle, if given a picture of a dog and asked to label it as "stop sign" or "cup", the human labeller would be unable to make a correct decision. None of the labels in the label set can be reasonably associated with the input.

For natural inputs, the output of the oracle for a given x is referred to as the **true label** of x.

With the formal definition of an oracle, we can now verify the correctness of a classifier. Given an input $x \in N_F$, $F(x)$ is **correct** if and only if $F(x) = O_F(x)$. Otherwise, F is **incorrect**. If $\forall x \in N_F, F(x) = O_F(X)$, then we say that F is **perfectly accurate**.

We will denote the space of inputs correctly labelled by F as C_F and the space of inputs incorrectly labelled by F as I_F. Therefore, $N_F = C_F \cup I_F$.

3.2 Adversarial Inputs

Informally, an adversarial input to a classifier is an input that is maliciously designed to be mislabeled by the classifier, but would be obvious for a human to label correctly. Furthermore, adversarial inputs typically appear almost identical to a correctly labelled input.

For a classifier F, we say that an input $x \in I_F$ is λ-**adversarial** if $\exists \gamma : |\gamma| \leq \lambda$ and $F(x+\gamma) = O_F(x+\gamma)$ and $O_F(x+\gamma) = O_F(x)$. Given a λ, we denote the space of λ-adversarial inputs for F as A_F^λ.

In current literature, γ is a measure of **distortion**, based on some measure of distance between two inputs. We use the addition operator (*e.g.,* $x+\gamma$), to represents changes made to an input x by some amount of distortion γ. λ is an upper bound on the distortion allowed when creating an adversarial input. It is typically small so adversarial inputs appear similar to correctly labelled inputs. For example, if the input is an image, the γ would be a vector of positive and negative pixel value modifications. A common choice for measuring $|\gamma|$ for image inputs is computing the L_p norm of γ.

Given a classifier F, we that F is λ-**resilient** if and only if $A_F^\lambda = \emptyset$. Furthermore, if for all $\lambda > 0, A_F^\lambda = \emptyset$, then we say that F is **perfectly resilient**. That is to say, there are no adversarial examples for F.

4 Resilient Feature Engineering

In this section, we present the two main theorems, which formally support resilient feature engineering. After which, we introduce resilient feature engineering and discuss the implications and use of the theorems.

4.1 Composition Resilience Theorems

We now present the Serial Composition Resilience and Parallel Composition Resilience theorems. The proof for both theorems can be found in the appendix. The Serial Composition Resilience theorem is as follows:

Theorem 1. *If a classifier function $F(\cdot)$ is λ-resilient for some λ, then classifier function defined as $G(F(\cdot))$ is also λ-resilient.*

The Serial Composition Resilience theorem demonstrates that one can use the output of a resilient classifier function as the input to another classifier function resulting in an equally resilient classifier function. Note that although we use the term "classifier", our definition of classifier function allows use to extend these results to other functional constructs. A feature extractor, for example, is a classifier function by definition. It takes in an input and produces an output belonging to a set of feature labels.

The Parallel Composition Resilience theorem is as follows:

Theorem 2. *If there are multiple classifiers $F_1(\cdot), F_2(\cdot)...F_n(\cdot)$ which are all λ-resilient for some λ, then the composition $G(< F_1(\cdot), F_2(\cdot)...F_n(\cdot) >)$ is also λ-resilient.*

The Parallel Composition Resilience theorem demonstrates that one can use the output of multiple resilient classifier functions as the input to another classifier function resulting in an equally resilient classifier function. This theorem extends the results of the Serial Composition Resilience theorem allowing us to join multiple resilient feature extractors as part of the classification decision. Often, using multiple feature can increase the selectivity for classification. For example, suppose F extracts the dominant sign color and G maps the color to set of road sign containing the color. Given the color "red", G could output {Stop sign, Do Not Enter sign}, which are both road signs that are predominantly red. However, adding a shape extractor would allow one to differentiate between a Stop sign and a Do Not Enter sign.

These two theorems allow us to simplify the task of creating an adversarially resilient classifier into building adversarially resilient feature extractors. The model designer would first identify predictive features for the classification task either using pre-existing domain knowledge or through some automatic discovery process. The difficulty remains in designing the feature extractors may be an easier task. If enough features are used, it may be possible to uniquely label every possible input.

4.2 Basic Design

Resilient feature engineering is composed of the following steps:

1. Identify a feature for the classification task

2. Create a λ-resilient feature extractor for the identified feature

3. Repeat steps 1 and 2 until no new features are identified or the selectivity given by the features does not increase
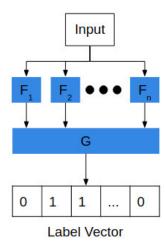
Figure 1: The basic implementation of a resilient classifier built using resilient feature engineering. The output is a vector of 0's or 1's with dimensions equal to the number of possible output classes. A 1 in index i means that class label i is a possible label for the input based on the extracted features.

4. Provide the output of the feature extractors as an input to a classifier, which maps the features to the label set for the classification task

In this paper, we assume feature identification is based on pre-existing domain knowledge. Each identified feature should result in an increase in selectivity for the final classifier. Otherwise, the feature is redundant. The basic model architecture implied by the two theorems is shown in Figure 1.

First, the input x is processed by each of the λ-resilient feature extractors. Then, the set of extracted features for that input is provided to a different classifier G, which outputs a vector containing one or more non-zero values. If the extracted features are sufficient to uniquely identify the label for x, then only one non-zero value is present. Otherwise, multiple non-zero values may be present indicate multiple possible output labels due to shared feature sets between labels. The weighting of the non-zero values in the final output vector is a design decision. In Figure 1, we assume equal weighting, but it could be based on some contextual information. Using unequal weighting may be desirable, especially if the output vector is intended to be used in a future computation.

We will use a basic traffic sign classification task to illustrate how the basic implementation works. Suppose, there are two resilient feature extractors, F_1 and F_2, that extract the dominant color and shape of a sign in the input respectively. G can be any classification algorithm, but for simplicity we will assume it is a decision tree with the possible output vector values pre-determined based on

existing domain knowledge. Furthermore, we will limit the possible output labels to the following list of U.S. stop signs[1]:

- Regulatory: Stop, Yield, Do Not Enter

- Warning: Left Turn Ahead, Right Turn Ahead

- Pedestrian: No Pedestrians

- Other: Speed Limit 25, Speed Limit 45, Hospital

Therefore, the output vector has 9 indices with the labels ordered based on the above list. We denote the combined classifiers F and G as R for convenience. Now, consider the following input signs: Stop, Left Turn Ahead, Hospital. The resilient classifiers F_1 and F_2 would have the following outputs:

- Stop: $F_1(x) = $ Red, $F_2(x) = $ Octagon

- Left Turn Ahead: $F_1(x) = $ Yellow, $F_2(x) = $ Diamond

- Hospital: $F_1(x) = $ Blue, $F_2(x) = $ Square

and the output of G based on the extracted features would be:

- Stop: $G(x) = <1,0,0,0,0,0,0,0,0>$

- Left Turn Ahead: $G(x) = <0,0,0,1,1,0,0,0,0>$

- Hospital: $G(x) = <0,0,0,0,0,0,0,0,1>$

For the Stop sign, knowing that the sign is red is insufficient to uniquely label the sign given the set of possible labels. Both Yield and Do Not Enter signs are also red. However, knowing the shape of the sign is sufficient to identify the sign as a Stop. We see a similar result for the Hospital sign. The shape of the sign isn't sufficient because a No Pedestrian sign is also a square, but the color, blue, uniquely identifies the sign based on the limited label set.

For the Left Turn Ahead sign, the output vector contains multiple non-zero indices because the extracted features belong to two possible signs. Left Turn Ahead and Right Turn Ahead are both "yellow" and "diamond". Therefore, the extracted features indicate the output is either of these two sign labels, but cannot specify further. Although this may seem like a limitation, consider the implementation shown in Figure 2. The label vector would only allow for adversarial attacks that between classes that share the same set of extracted features. In this example, it is not possible to cause the classifier to label a Stop sign as a Left turn Ahead sign and vice versa.
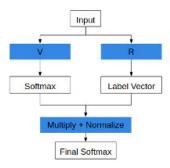
Figure 2: Augmenting an existing classifier *V* with the robust classifier. The softmax output of *V* is multiplied with the vector of possible class labels generated by the robust classifier *R*. The result is re-normalized to obtain a new softmax output.

5 Conclusion and Future Work

In this paper, we introduce resilient feature engineering as a methodology for designing adversarially resilient classifiers. Our methodology is supported by the Serial Composition and Parallel Composition theorems, which allow for the chaining of resilient functions to create an adversarially resilient classifier. We also discussed a basic implementation of an adversarially resilient classifier.

However, we leave several open problems for future work. The first problem is identifying predictive features to extract. In this paper, we discussed using pre-existing domain knowledge to identity features useful for classification, but such easily identifiable features are often not present. In other machine learning applications, such as medical imaging or speech recognition, it may be unclear what features are useful. However, work on improving the explainability of learned features as well as determining their relevance to the final prediction may be useful towards feature identification.

A second problem is the feasibility of designing adversarially resilient classifiers. For simpler features, such as color, it may be sufficient to compute the most common color in the input. However, more complex feature such as shape may require more powerful algorithms such as deep neural networks, due to timing constraints or feature complexity. For this, we turn to the existing work in provable adversarial security. Although the existing research focuses providing resilience guarantees for a general classifier, subdividing the classifier into a set of smaller feature extractors may result in more efficient or stronger resilience guarantees.

Acknowledgements

References

[1] ATHALYE, A., AND CARLINI, N. On the robustness of the CVPR 2018 white-box adversarial example defenses. *CoRR abs/1804.03286* (2018).

[2] ATHALYE, A., CARLINI, N., AND WAGNER, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICLR* (2018).

[3] ATHALYE, A., ENGSTROM, L., ILYAS, A., AND KWOK, K. Synthesizing robust adversarial examples. *CoRR abs/1707.07397* (2017).

[4] CARLINI, N., AND WAGNER, D. Towards evaluating the robustness of neural networks. In *S&P* (2017), IEEE, pp. 39–57.

[5] CARLINI, N., AND WAGNER, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR abs/1705.07263* (2017).

[6] CHALASANI, P., JHA, S., SADAGOPAN, A., AND WU, X. Adversarial learning and explainability in structured datasets. *CoRR abs/1810.06583* (2018).

[7] CHEN, S., CORNELIUS, C., MARTIN, J., AND CHAU, D. H. Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector. *CoRR abs/1804.05810* (2018).

[8] EYKHOLT, K., EVTIMOV, I., FERNANDES, E., BO LI, A. R., TRAMER, F., PRAKASH, A., KOHNO, T., AND SONG, D. Physical adversarial examples for object detectors. In *WOOT* (2018).

[9] EYKHOLT, K., EVTIMOV, I., FERNANDES, E., LI, B., RAHMATI, A., XIAO, C., PRAKASH, A., KOHNO, T., AND SONG, D. Robust physical-world attacks on machine learning models. In *CVPR* (2018).

[10] GONG, Z., WANG, W., AND KU, W. Adversarial and clean data are not twins. *CoRR abs/1704.04960* (2017).

[11] GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. *CoRR abs/1412.6572* (2014).

[12] HEIN, M., AND ANDRIUSHCHENKO, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. *NIPS* (2017).

[13] HENDRYCKS, D., AND GIMPEL, K. Visible progress on adversarial images and a new saliency map. In *ICLR* (2017).

[14] KURAKIN, A., GOODFELLOW, I. J., AND BENGIO, S. Adversarial examples in the physical world. *CoRR abs/1607.02533* (2016).

[15] LI, X., AND LI, F. Adversarial examples detection in deep networks with convolutional filter statistics. In *ICCV* (2017).

[16] MOOSAVI-DEZFOOLI, S.-M., FAWZI, A., AND FROSSARD, P. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR* (2016).

[17] PAPERNOT, N., MCDANIEL, P., JHA, S., FREDRIKSON, M., CELIK, Z. B., AND SWAMI, A. The limitations of deep learning in adversarial settings. In *EuroS&P* (2016), IEEE, pp. 372–387.

[18] PAPERNOT, N., MCDANIEL, P. D., WU, X., JHA, S., AND SWAMI, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *S&P* (2016).

[19] RAGHUNATHAN, A., STEINHARDT, J., AND LIANG, P. Certified defenses against adversarial examples. In *ICLR* (2018).

[20] SINHA, A., NAMKOONG, H., AND DUCHI, J. Certifying some distributional robustness with principled adversarial training. In *ICLR* (2018).

[21] SONG, Y., KIM, T., NOWOZIN, S., ERMON, S., AND KUSHMAN, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *ICLR* (2018).

[22] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. In *ICLR* (2014).

[23] TRAMÈR, F., KURAKIN, A., PAPERNOT, N., GOODFELLOW, I., BONEH, D., AND MCDANIEL, P. Ensemble adversarial training: Attacks and defenses. In *ICLR* (2018).

[24] TSIPRAS, D., SANTURKAR, S., ENGSTROM, L., TURNER, A., AND MADRY, A. Robustness may be at odds with accuracy. *CoRR abs/1805.12152* (2018).

[25] WONG, E., AND KOLTER, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. *ICLR* (2018).

A Formal Proofs

A.1 Proof of Theorem 1

Suppose we have two label sets denoted as Y and Z, such that $Y \cap Z = \emptyset$. That is to say, they do not share any labels. Despite this, let us assume it is possible to map some or all of the labels in Z to labels in Y. We will define this mapping function as the oracle O_L of the classifier L where $L(z) = y$. We define two classifiers F and G as such:

- $F(x) = y \in Y$

- $G(y) = z \in Z$

where F is λ-resilient. Using F and G, we also use a classifier R to denote the composition $G(F(x))$:

- $R(x) = G(F(x)) = z \in Z$

- $O_R(x) = O_G(O_F(x)) = l \in Z$

Note that the mapping function O_L is the inverse function for O_G since both are oracle functions and thus always produce the correct result.

Using these definitions, we restate the Serial Composition Resilience theorem:

Theorem. *For arbitrary $z_1, z_2 \in Z$, if $O_L(z_1) = y_1$, $O_L(z_2) = y_2$, $y_1 \neq y_2$, and $R(x) = z_1$, then there is no $\alpha < \lambda$ and $|\gamma| < \alpha$ such that $R(x + \gamma) = z_2$ and $O_R(x + \gamma) = O_R(x) = z_2$*

We will verify this theorem using a proof by contradiction and show that assuming the opposite of the implication will result in a contradiction with the assumption that F is perfectly resilient.

$\exists \alpha < \lambda, 0 < |\gamma| < \alpha :$	Assume
$R(x + \gamma) = z_2$ and	
$O_R(x) = O_R(x + \gamma) = z_2$	
$G(F(x + \gamma)) = z_2$	Def of R
$R(x) = z_1$	Given
$G(F(x)) = z_1$	Def of R
$z_1 \neq z_2$	O_L is a function
$G(F(x)) \neq G(F(x + \gamma))$	Substitution
$F(x) \neq F(x + \gamma)$	G is a function

With this, we have show that the initial assumption has resulted in $F(x) \neq F(x + \gamma)$. To complete the proof, we need to demonstrate that $F(x + \gamma) = O_F(x + \gamma)$. In doing so, it would mean that x is an adversarial example

in A_F^α and since $\alpha < \lambda$, F would not be λ-resilient by definition.

$\exists \alpha < \lambda, 0 < |\gamma| < \alpha :$	Assume
$R(x + \gamma) = z_2$ and	
$O_R(x) = O_R(x + \gamma) = z_2$	
$O_L(z_2) = y_2$	Given
$O_G(O_F(x)) = z_2$	Def of O_R
$O_L(O_G(O_F(x))) = O_L(z_2)$	
$= O_F(x) = y_2$	O_L is the inverse of O_G
$O_G(O_F(x + \gamma)) = z_2$	Def of O_R
$O_L(O_G(O_F(x + \gamma))) = O_L(z_2)$	
$= O_F(x + \gamma) = y_2$	O_L is the inverse of O_G

At this point, we've show that the true label for x and $x + \gamma$ based on the label set Y is y_2. What remains for the proof is to establish that F outputs a label for x that is incorrect.

$R(x + \gamma) = z_2$	Given
$G(F(x + \gamma)) = z_2$	Def of R
$G(F(x + \gamma)) = O_G(O_F(x + \gamma))$	Substitution
$F(x + \gamma) = O_F(x + \gamma)$	Apply O_L
$F(x) \neq F(x + \gamma)$	Established previously
$F(x) \neq O_F(x)$	Substitution
$x \in I_F$	Def of I_F
$x \in A_F^\lambda$	Def of A_F^α
F is not λ-resilient	Def of λ-resilient

However, it was given that F is perfectly resilient. Therefore, Theorem 1 is true through proof by contradiction.

A.2 Proof of Theorem 2

Suppose we have $n + 1$ label sets denoted $Y_1, Y_2 ..., Y_n, Z$ such that $Y_1 \cap Y_2 ... \cap Y_n \cap Z = \emptyset$. We define the mapping function as the oracle O_L of the classifier L where $L(z) = < y_1, y_2 ..., y_n >$. y_i denotes a label in the label set Y_i.

We define have $n + 1$ classifiers such that:

- $F_1(x) = y_1 \in Y_1 ..., F_n(x) = y_n \in Y_n$

- $G(< y_1, y_2 ..., y_n >) = z \in Z$

where each F_i for $i = 1...n$ is λ-resilient. Note that O_L is the inverse function of O_G.

Using these classifiers, we use R to denote the composition of G and each F_i:

- $R(x) = G(F_1(x), F_2(x)..., F_n(x)) = z \in Z$

- $O_R(x) = O_G(O_{F_1}(x), O_{F_2}(x)..., O_{F_n}(x)) = l \in Z$

Using these definitions, we restate the Parallel Composition resilience theorem:

Theorem. *For arbitrary $z_1, z_2 \in Z$, if $O_L(z_1) = C_1$, $O_L(z_2) = C_2$, $C_1 \neq C_2$, and $R(x) = z_1$, then there is no $\alpha < \lambda$ and $|\gamma| < \alpha$ such that $R(x + \gamma) = z_2$ and $O_R(x + \gamma) = O_R(x) = z_2$*

For clarification, C_1, C_2 are n-tuples and elements of the set $Y_1 x Y_2...x Y_n$. We will verify this theorem using a proof by contradiction and show that assuming the opposite of the implication will result in a contradiction with the assumption that every classifier F is perfectly resilient.

$\exists \alpha < \lambda, 0 < |\gamma| < \alpha:$	Assume
$R(x + \gamma) = z_2$ and	
$O_R(x) = O_R(x + \gamma) = z_2$	
Let $S_2 = < F_1(x + \gamma)..., F_n(x + \gamma) >$	Let
$G(S_2) = z_2$	Def of R
$R(x) = z_1$	Given
Let $S_1 = < F_1(x)..., F_n(x) >$	Let
$G(S_1) = z_1$	Def of R
$z_1 \neq z_2$	Given
$G(S_1) \neq G(S_2)$	Substitution
$S_1 \neq S_2$	G is a function
$\exists i$ for $1 \leq i \leq n:$	Equality of n-tuples
$F_i(x) \neq F_i(x + \gamma)$	

With this, we have show that the initial assumption has resulted in $F_i(x) \neq F_i(x + \gamma)$ for some i between 1 and n. We now move to show that $x \in I_{F_i}$ and $x + \gamma \in C_{F_i}$ for that i.

$\exists \alpha < \lambda, 0 < |\gamma| < \alpha:$	Assume
$R(x + \gamma) = z_2$ and	
$O_R(x) = O_R(x + \gamma) = z_2$	
$O_L(z_2) = C_2$	Given
Let $T_1 = < O_{F_1}(x)..., O_{F_n}(x) >$	Let
$O_G(T_1) = z_2$	Def of O_R
$O_L(O_G(T_1)) = O_L(z_2)$	
$= T_1 = C_2$	O_L is the inverse of O_G
Let $T_2 = < O_{F_1}(x + \gamma)..., O_{F_n}(x + \gamma) >$	Let
$O_G(T_2)) = z_2$	Def of O_R
$O_L(O_G(T_2)) = O_L(z_2)$	
$= T_2 = C_2$	O_L is the inverse of O_G
Let c_z^i denote the ith element of C_z	Let
$O_{F_i}(x) = O_{F_i}(x + \gamma) = c_2^i$	Equality of n-tuples

Remember that i is the index for the classifier function that is not λ-resilient. At this point, we've show that the true label for x and $x + \gamma$ based on the label set Y_i is c_2^i. What remains for the proof is to establish that F_i outputs a label for x that is incorrect.

$R(x + \gamma) = z_2$	Given
$G(S_2) = z_2$	Def of R and S_2
$G(S_2) = O_G(T_2)$	Substitution
$S_2 = T_2$	Apply O_L
$F_i(x + \gamma) = O_{F_i}(x + \gamma)$	Apply O_L
$F_i(x) \neq F_i(x + \gamma)$	Established previously
$F_i(x) \neq O_{F_i}(x)$	Substitution
$x \in I_{F_i}$	Def of I_{F_i}
$x \in A_{F_i}^\alpha$	Def of $A_{F_i}^\alpha$
F_i is not λ-resilient	Def of λ-resilient

However, it was given that for $i = 1...n, F_i$ is λ-resilient. Therefore, Theorem 2 is true through proof by contradiction.