

# Tracking Effects Layer by Layer from Adversarial or Naive Changes to a CNN Inputer

James P. LaRue

JADCO Signals  
James@JadcoSignals.com

**Abstract.** — In the AIPR 2017 Workshop we presented a paper that looked at “The Anatomy of a Neural Network”. The main purpose was to show how to take a simple LeCun-based neural network, with its multiple internal operations, and extract layer-to-layer input/output pairs to form a series of bidirectional association memory matrices (AMMs) from CNN Layers.

From these transformations we showed how to achieve a practical solution that met the expectations of the Universal Approximation Theorem; how we could to collapse a six-layer system into a single layer.

For the AIPR 2025 Workshop we will examine how a change in the image, be it adversarial or just a naïve-rotation of the image, can be physically tracked across each of the CNN/AMMs, when each is examined in vector form, i.e., we will visually exploit the error from the first layer on to the last layer.

There are many approaches to measuring layer vulnerabilities, such as auto-encoders, t-SNEs, histograms, and residuals. In contrast to these methods, we do not offer a measure, we are simply demonstrating that we can take a step back and look at how the error, be it adversarial or naïve, propagates within the neural network framework to make a more informed and more immediate decision on designing a measure.

**Keywords:** neural-network framework, adversarial intentions, layer by layer

## 1 Challenging the Panda Plus Noise Equals Gibbon Mirage

### 1.1 The Adversarial Icon Has More to Offer

The Panda Plus Noise example is classic in that it was one of the first adversarial examples to challenge the accuracy of the Neural Network. As a reminder we show the example in Figure 1. The motivation for submitting an abstract for consideration by AIPR-2025 was to bring up two points that may be worthy of consideration. The first relates to the decision-output in the context of exactly how many classes were trained in that NN and what were in the individual class scores with just the noisy image inputted by itself. The second question asks if there signs of ‘mis-classification starting early on, say, beginning with the first layer, i.e., why wait until the end to find out there is an error from something adversarial or from some unintentional naïve change to the input.

James P. LaRue

Since we do not have access to the original Panda Bear example, we have decided to use a simple six-layer NN trained on the very classic MNIST data set [1]. The examples will demonstrate the unquestionable need to look across the class scores in the decision layers and how to further analyze NN process early on by demonstrating that it is possible to flatten out the initial layers of the so-called black-box for early warnings.

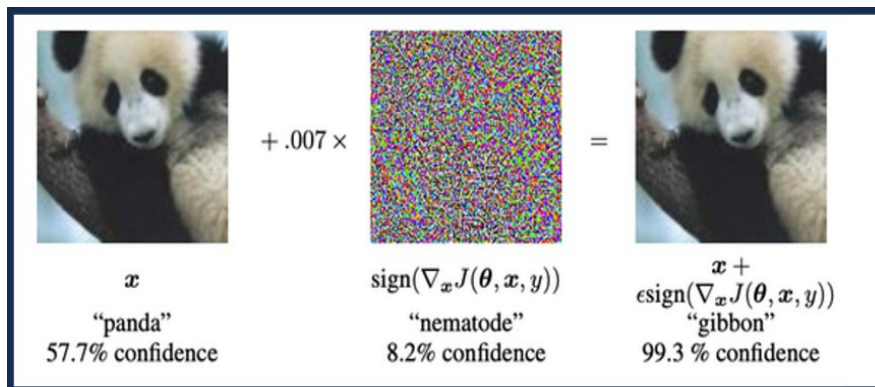
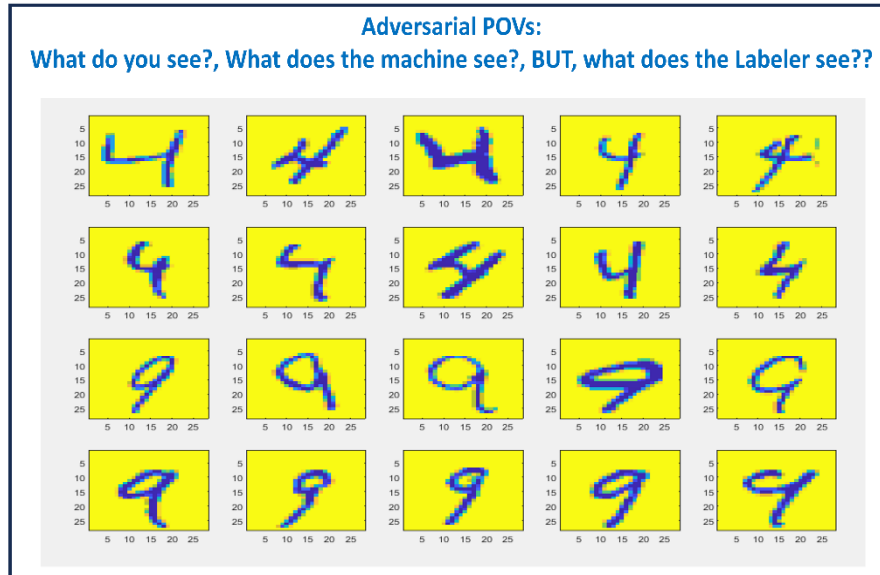


Figure 1. Question: Where is a picture of “the closet looking” Gibbon?

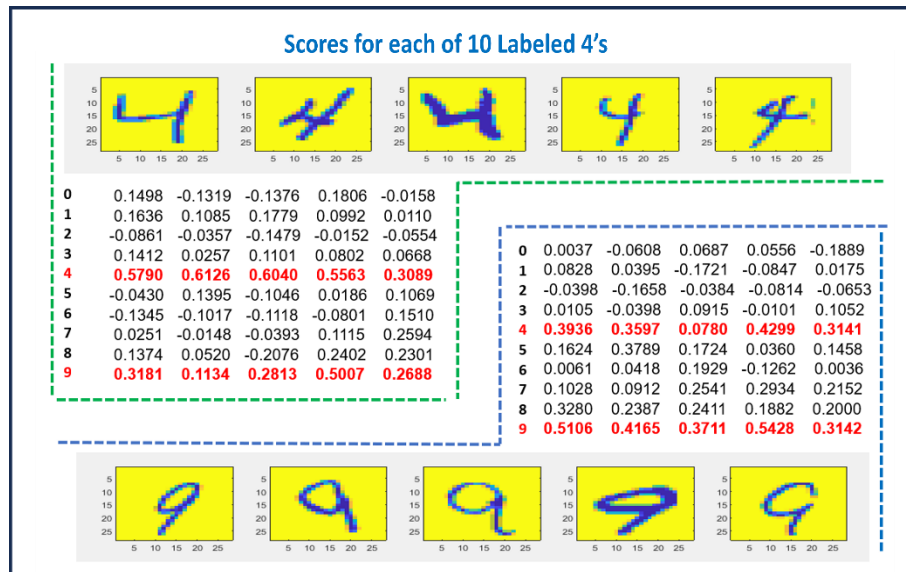
## 2 The Fours

### 2.1 The Fours and their scores

Figure 2 shows 20 examples of the number four drawn from the MNIST data set and you may agree that the top two rows appear as fours while the bottom two rows appear as nines. Figure 3 shows the scores from the top row and bottom row that our neural network assigned to each of the classes 0-9. Starting our analysis with the top row of fours we see that the second 4 scored highest in its class compared to the others, the fourth 4 narrowly scored higher than the 9's score, and the fifth 4 scores were uniform distributed across several classes although the 4's class did score higher. Now we turn our attention to the bottom row of so-labeled 4's and observe that this neural network scored all of them as 9's. It could be argued, and rightfully so, that had we continued training the NN over several epochs the scores may gravitate towards the correct 4 class although the counter argument for “overtraining”, making the NN brittle, also would have its place. But what we want to bring out, in the context of “adversarial and naïve changes”, may best be discussed using the 5<sup>th</sup> “4” on the right. The 4-9 scores are 0.0001 apart, with the 9's class winning the contest. What if an executable version of this NN was uploaded free to use on some website, then we, the one who trained it, would be aware of the pitfall of the couple missing pixels on the right side and what would happen if we, added to or removed from, a pixel in that opening, but adjusting that image and run it through the algorithm. As an interesting side-note, we gave an AI engine the bottom right “4” and asked it “What number do you see?”. Its answer: “The letter g”.



**Figure 2.** Selected 4's from labeled MNIST data set. Were any mis-labeled?



**Figure 3.** 4's and their winning/losing scores.

Figure 4 shows how we track the winning scores across each class. We tested the NN with 500 images each of 0-9. The top plot shows the scores where the blue dots are

James P. LaRue

the winning scores the true 4's and the red dots are the winning scores when from another class. Knowing these scores is what facilitated our selected examples in Figure 2.

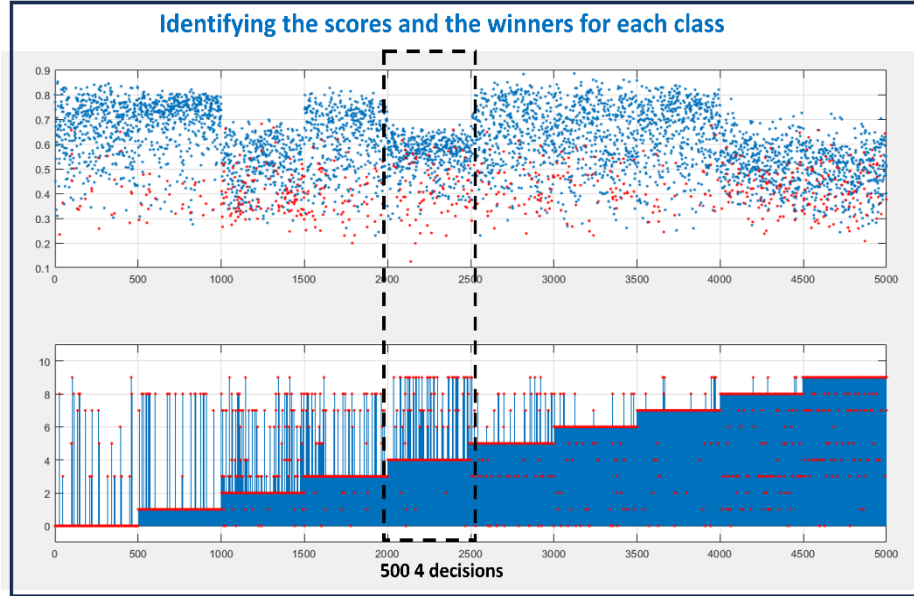


Figure 4. Scores are plotted for 500 images from each test class left to right.

### 3 Let's Open the Black Box and Flatten it

#### 3.1 Flattening all layers are review

In this section we will open up the so-called Black Box that sits between the input and the output layers. The concept was previously examined at AIPR 2017 with the purpose of describing our patented procedure for converting a CNN (convolutional NN) construct into an Association Memory Matrix (AMM) construct, where we flattened each layer in CNN in order to construct the layer-to-layer associations which allows us to by-pass the CNN internal layer operations [2][3]. We will now quickly review the process.

We start with Figure 5 which was taken from Krizhevsky et. al.'s 2012 paper where he pictured the middle layers as matrices of "convoluted" puppy dogs and the output layer as the flattened layer where max-output decided to which class the image belonged [4]. Figure 6 is constructed to show the flattening of the second layer in our NN in the context of MNIST. We show the 28x28 matrix image of a labeled number 2. To flatten out the 2 we simply reshape it as a 1x784 vector. We then passed the image 2 through 12 Gabor (edge/orientation) detection filters and imposed internal layer down-sampling and other operations, to end up with 12 sets of 14x14 matrix images. Now in the same way as we reshaped the single input image into a flattened vector, we do the same with each of the 12 14x14 first layer outputs to form a layer 1 flattened output

vector of dimension  $12 \times (196 \times 1) = 2352 \times 1$ . Although the Gabor-generated images are flattened and concatenated, there remains an easily observed structural change as we observe the flattened vector left to right across the groups of 196 points.

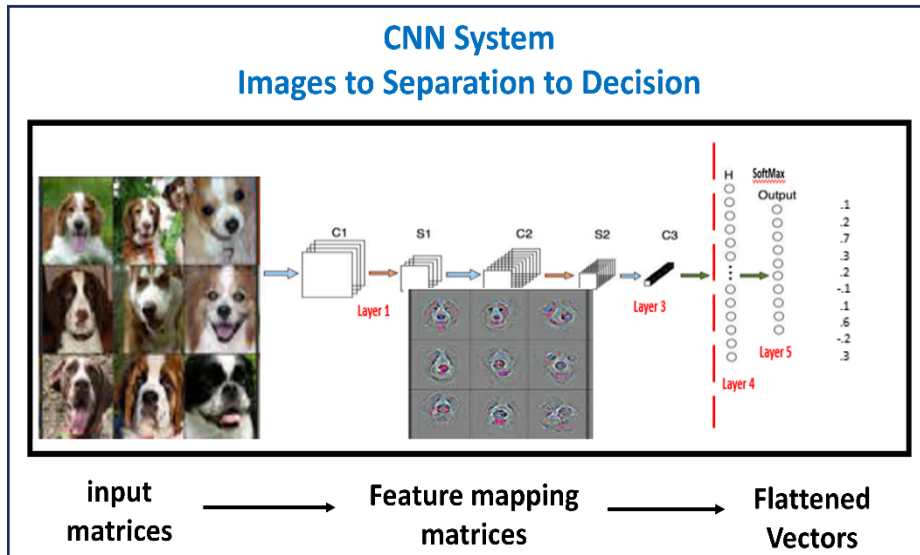


Figure 5. A CNN system with input, internal layer matrices, and a flattened out.

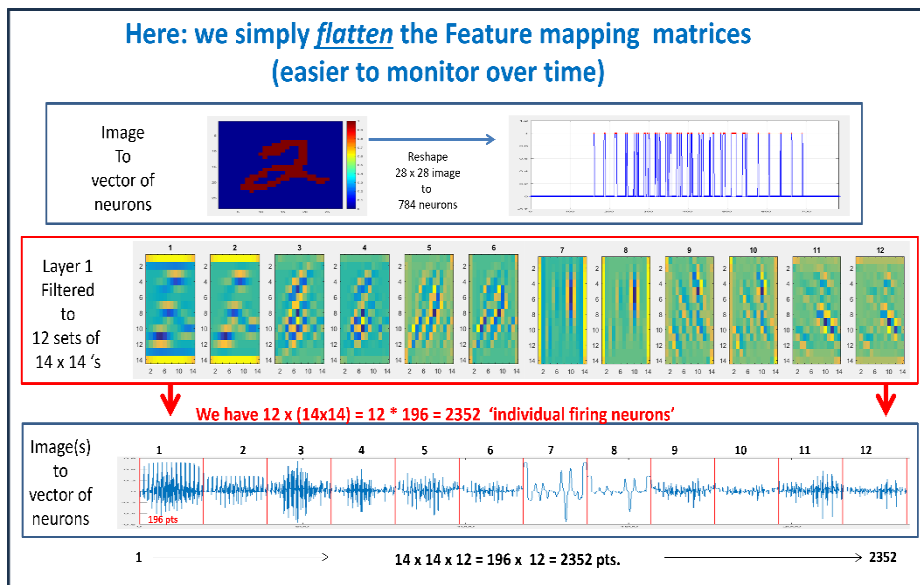
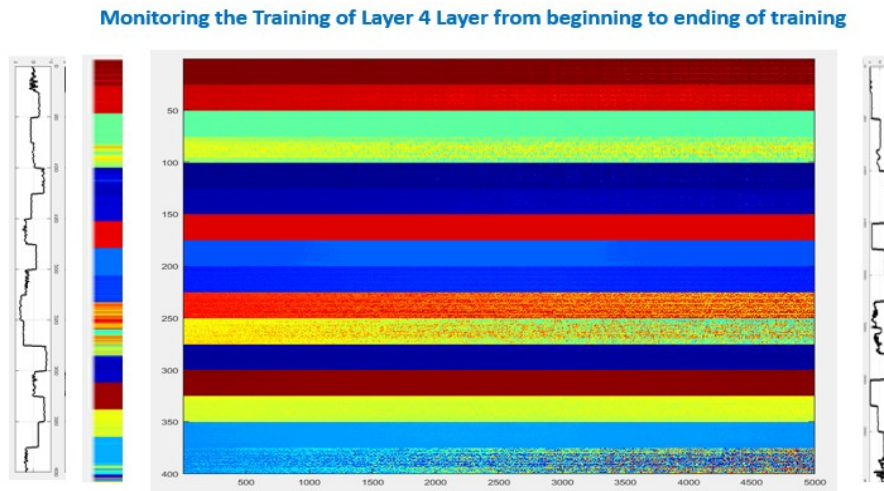


Figure 6. Flattening out the Gabor filter output from the second layer.

James P. LaRue

Figure 7 shows a flattening of sixteen 5x5 matrices from layer 4 but now concatenated as vertical vectors inside a matrix; this tracks the learning process, left to right, over 5000 iterations of training. The horizontal bands give insight into how each of the sixteen (flattened) matrices learn. It is easily observed that the columns take on a rougher appearance as the 5000<sup>th</sup> training iteration is reach. On the far left is the first training output vector while on the right is the output of some input number at the end of the process. We note that we used this approach of tracking flattened output for each input class to create our patented approach to converting a CNN into an AMM.



**Figure 7.** A time domain approach using column vectors to track output learning.

## 4 Two tracking examples

### 4.1 Tracking an image translation layer-by-layer

Figure 8 shows our layer-by-layer flattening concept using two number 7's. The 7 on the left is our starting point. It is centered. The left-side figure follows in this manner: To the right the seven is flattened into a vector, the layer two output, LGN to V1, (using ventral pathway terminology), is shown flattened in the green box, the next layer V1 to V2 is shown in the light blue box, and followed sequentially by the purple, yellow, and red boxes where the red box shows the out with the number 7's winning score. The matrices that are in between the boxes are for monitoring the outputs. Moving to the right-side figure, we start with an off-centered 7 on the right and follow its flattened vector and matrix monitoring changes until we reach the output and see the number 5 is the winner. The point we are making is whether the number contains an adversarial change or a naive change, we can monitor the stability of controlled changes to the input in its effects throughout the system. It is possible to place safeguards in the lower layers that become visual markers.

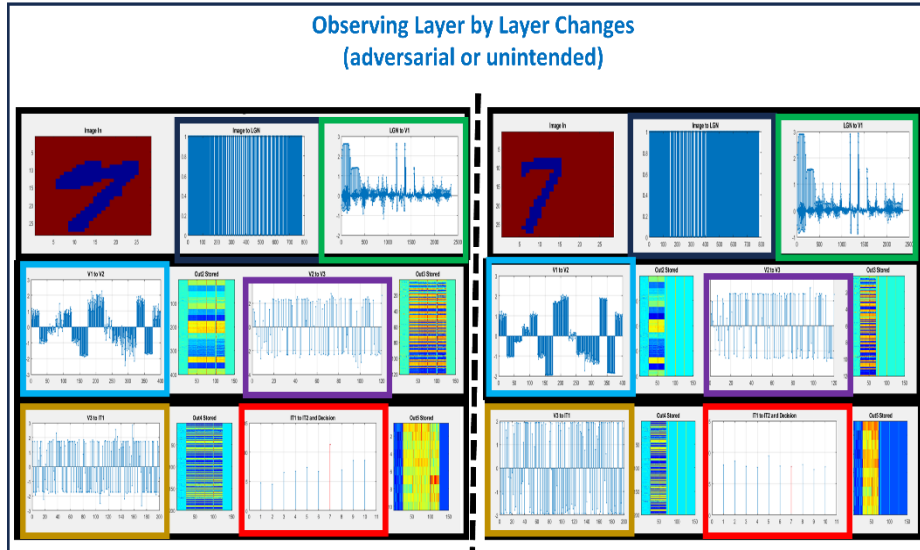


Figure 8. Sliding the number 7. Points 1-10 are ordered for classes 1-9 and 0.

#### 4.2 Tracking an image rotation at the output layer

Our last example, as shown in Figure 9 tracks the number 8 as it is rotated in one-degree increments. The top plot shows a column-by-column visualization of the output scores with red indicating the highest score. The second plot shows the winning score, the third plot indicates the winning decision class from 0-9, and the last plot shows the rotating 8's in increments of 30 degrees. What we find interesting is the stability of the rotating 8 decisions per winning class.

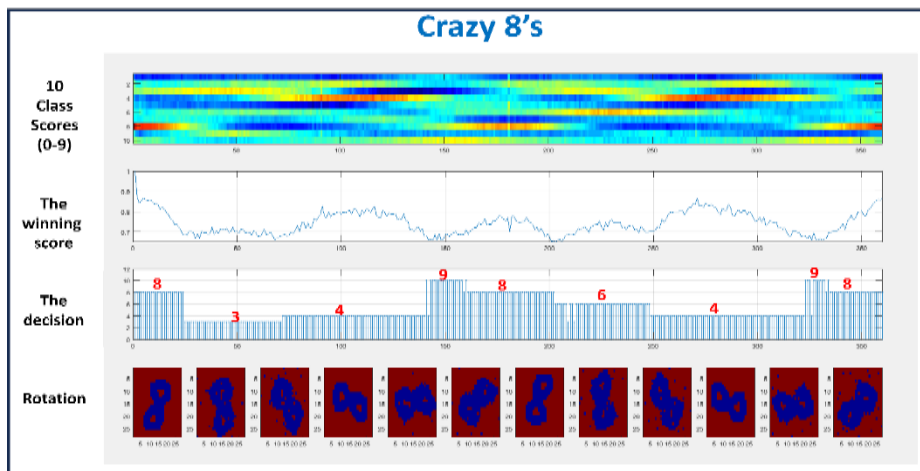


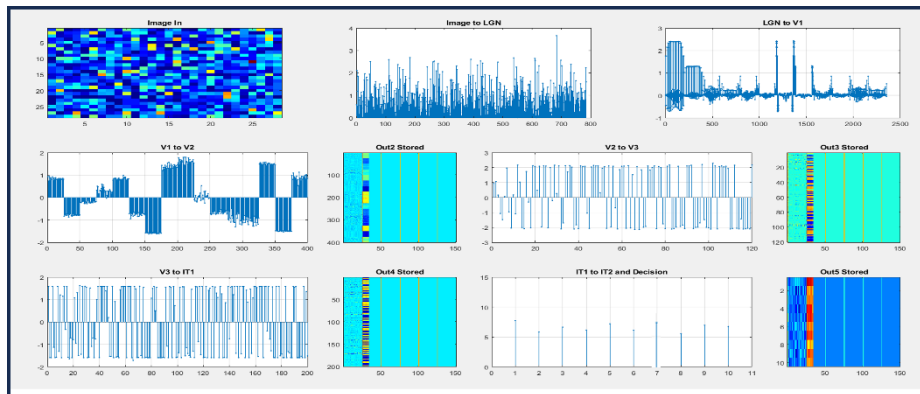
Figure 9. Tracking outputs of Rotating 8's.

## 5 Wrap up

### 5.1 It's just noise

We close this paper with Figure 10 where we show the progression of our “just noise” example throughout the NN. Of note is the comparisons between the output vectors of the 7 and the noise. For example, the LGN to V1 outputs show very similar structure, which shows there are constraints in the system stay in place. It's not until the output layer, where the noise shows a uniform distribution of scores among the ten classes. There is a winning score of course, the NN system is deterministic with its programmed number crunching and it will produce a maximum output, the number 1 in this case. Perhaps a measure of entropy could be applied to each layer inside, the so-called black box, which hopefully, we have made a little more transparent.

We wish to thank the AIPR workshop for the invitation to present and George Washington University for hosting the event.



**Figure 10.** Noise as it traverses through the neural network. Points 1-10 are ordered for classes 1-9 and 0.

**Disclosure of Interests.** The author has no competing interests to declare that are relevant to the content of this article.

## References

1. Y. LeCun, Courant Institute, Corinna Cortes, Google Labs, New York, “The MNIST database of handwritten digits,” <http://yann.lecun.com/exdb/mnist>
2. James LaRue<sup>1</sup>, Richard Tutwiler<sup>2</sup>, Dennison LaRue<sup>3</sup>, ‘The Anatomy of a Neural Network’. IEEE AIPR Workshop, The Cosmos Club, Washington DC (2017), <sup>1</sup>Jadco

Signals, <sup>2</sup> Professor Emeritus Applied Research Lab Penn State University/CEO LiveMotion3D LLC, <sup>3</sup>University of South Carolina Law School.

3. James LaRue and Denise LaRue, Patent (*Granted March 2019*): Joint Proximity Association Template for Neural Networks. A tool to convert Convolutional Neural Networks into a matrix-based system allowing for order of magnitude increase in intra-layer execution and allows for inter-layer stitching between neural networks. US 10,242,313, B2.
4. Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey, "ImageNet Classification with Deep Convolutional Neural Networks, Advances in neural information processing systems, 2012, pp. 1097-1105.