**PRODUCT PERSONALIZATION**

*Let your product personalize itself to the needs of your customers.*

Smart products generate notifications to its users when they detect events of interest. Alexa messages users when she detects a smoke alarm. Ring cameras notify users when they detect intruders. Traffic monitoring systems inform police when they detect unusual traffic activity. Pipeline security systems inform gendarme when they detect oil theft. Border security systems inform guards when they detect border violations.

**False Alarms**

Possibility of “false alarms” (FA) is a major area of concern for product builders and users in all these examples. A false alarm is a notification that was generated inaccurately. If Alexa sends a smoke alarm notification to a user when there is no smoke, it at least annoys the user. If a pipeline security system generates FA, it may end up engaging gendarme to a remote location for no reason, wasting precious resources. If the FAR rate (FAR) is high, the user may stop using the product (or a feature of the product) altogether. Achieving an acceptable level of FAR is therefore a critical design issue.

FAs happen if the product does not fully capture the variation in where and how the product is used. Products may work well in the *fat head* of the distribution, which correspond to generic use scenarios and environments, but may fail more often in the *long tail* for less common and more specific cases. Ring cameras works well in most places, but may fail if the sun shines to a wall, which can be mistaken with a true motion. Eliminating such FAs in customer-specific cases is a challenge but can improve the value of the product significantly.

**Rule-based Personalization**

One way to reduce FAs is to personalize the product to the customer’s unique context and intent. If the Ring camera *knows* that sun shines to a particular place at a certain time of the day for one customer, then it can avoid generating FAs for that customer. Businesses have used rule-based technology to render personalization to date to tackle such situations. However, rule-based personalization has limits.

First of all, *a rule-based engine cannot improve over time*. Rule-based personalization relies on an impossible amount of human intervention and judgment. A developer would need to consider every data point and variable associated with each customer. They must determine what precisely causing a customer to behave in a certain way. A developer would then need to determine the impact of that correlation in relation to every combination of factors. No human is up to the task.

Moreover, *a rule-based engine is not able to update rules in real-time*. Rule-based personalization is determined and configured by a human. Someone literally needs to set the personalization rules. It is theoretically possible to do this on a daily basis but not on a one-to-one basis for more than a handful of customers.

*We aim to provide a commercial solution to this problem: The solution we are developing is an AGI-based Arbitrary Reasoning Engine (ARE) that can personalize your product by itself to the needs of your customers, specifically to achieve better FAR.*

**A General-Purpose Solution**

ARE is a general-purpose, domain-independent reasoning engine that does not require code changes to adapt to the individual context of different customers and applications. With only a few training samples, it can find unique patterns that result in FAs. This is critical for success, because the customers are expected to have only a few examples of FAs that they can use to personalize their products.

ARE can work with a single product (APP), or a collection of multiple products. APPs to be personalized will always have their own built-in AI logic. Ring cameras for example, have built-in capability to process video to detect motion, and Alexa to process audio to detect acoustic events. ARE will provide a complementary logic that can uncover patterns associated with false alarms, but are outside of the sight of individual APPs. Rather than processing video or audio signals directly (even though ARE can do this), it will process the outputs of Ring and Alexa at a meta-level, together with other available data, such as location and time, to discover patterns that result in FAs, as illustrated in the following figure.

Intruder

**ARE**

Motion Activity Time

**Internet**

**Alexa**

**Ring**

Video Sound Information

In its simplest mode, ARE will support applications in a supervised manner. The applications will provide training samples to ARE and indicate if it is an FA. This can be accomplished by simply requesting feedback from the user. The user clicks a button to indicate if this is a true alarm (TA) or FA. The APP then provides this annotation together with other relevant information to ARE, which in turn treats this information as a training sample, and updates its internal model incrementally to explain this sample together with all the previous training samples.

**Model**

**Base**

**ARE**

**APP**

Training Data

Inference

**A Sample Use Case**

Consider the example of monitoring a home using cameras. This is a simple but a real use case. I am monitoring my house in Massachusetts with multiple Ring cameras. When a camera detects a motion, it sends a notification so that I can watch the cameras alive to see what is happening. The system is working almost flawlessly, except one persistent pattern of false alarms. I can tell whether a notification is an FA by just looking at the time of the day (ToD) and which camera is generating the alarm. If it is around 3 pm EST and the notification is coming from Cam1, then I know that it is an FA. What happens is that the sun reflects (from the cars passing by the street) and shines into a wall visible to Cam1 and confuses the Ring software. Because of the angle of the sun, this happens only at around 3 pm.

Even though the FAs do not happen every single day and at exactly the same ToD, ARE should be able to figure this out by just correlating the FA with the ToD for individual devices. It will not only discover that this is happening around 3 pm but also figure out that it happens for Cam1 only. This info is sufficient to reduce the FAR of this system from about 2 per week to 0, from annoyance to a perfect system. An ARE executable can do this alone (without doing anything specific) after connecting the output of Ring to ARE, as follows.

Intruder?

**ARE**

User Feedback

Motion Time

**Internet**

**Ring**

Video Information

Now, it is critical to answer the following questions to see if ARE has an advantage over NNs in this specific example:

*Q. Can we achieve the same success with neural networks (NNs)?*

A. Since they cannot learn from a few examples, this is out of reach for NNs, even if we build a model specifically to catch such a pattern.

*Q. What if we support NNs with transfer learning or meta-learning?*

A. The intent here is to train an NN with data from a large population first so that it can learn customer-specific patterns from a few samples later. This is possible, only if the customer-specific pattern is visible in the larger population. There should be a meaningful underlying probability distribution. This is however unlikely, because FAs from sunshine will be arbitrarily distributed across all ToDs depending on device location. There is no such thing that FAs will peak around 3 pm in the population of all Ring devices. If my house were on the other side of the street, then Cam3 would be generating the FAs at around 10 am.

*Q3. Can a single NN serve different applications each having different sets of sensors?*

A. A single ARE executable can serve difference customers, different applications and different sets of sensors. It can automatically reconfigure itself to support such variations. NN based architectures may require building and training different networks for these. We may have to investigate this further, but the answer is probably “no”.

*Q4. Is this correlation or causation?*

A. It is true that what ARE learned in this example is correlation not causation. ToD is not the cause of this FA; it is the angle of the sun. However, we use the term reasoning to refer to the process, not the outcome. As human beings, we take into account other information, such as the orientation of windows, the angle of sunshine, camera FoV, the street location, and so on.

**Plug and Play Reasoning Engine**

If ToD is the only piece of information to ARE, then it can falsely reject true Ring alarms. After learning the above pattern, if an intruder can enter the home from the front door (Cam1 FoV) around 3 pm, then ARE can falsely overwrite the motion signal coming from the Ring. If ARE has additional information however, it can prevent this. Let’s add an Alexa to detect activity sounds into the system, as follows.

Intruder?

**ARE**

User Feedback

Motion Activity Time

**Internet**

**Alexa**

**Ring**

Video Sound Information

Now, when somebody enters the house around 3 pm from the front door, ARE will figure out from the user feedback that such a pattern is not an FA if Alexa also detects an activity from the sound signals. What is more important, the same ARE executable will be able to do this without a single line of code change or reconfıguration. Through this plug and play capability, ARE will be able to scale to wide range of environments and customers.

**Engineering Challenges for Adaptation**

ARE has a plug and play architecture in terms of reasoning capabilities. We still need to communicate with products to send and receive information. In that sense, we can only integrate with APPs that provide the proper APIs. Ring and Alexa, in the above example, may not readily have such interfaces. One way to tackle this problem is to create an ecosystem of devices that are developed with ARE integration in mind. This however requires a backing of a strong player in the industry. Another approach can be to reduce the area of interest to devices that hast the required APIs. This requires further product investigation.

**Potential Use Cases**

ARE can contribute to reducing the FAR for individual sensors, or multi-sensors applications. The following are examples of how ARE will help product personalization to reduce FAs:

1. **Surveillance Cameras:** Lighting condition may change depending on the location of the camera and the time of the day. Ring camera generating FAs above is an example of this. ARE can help eliminating such FAs.
2. **Acoustic Sensors:** Acoustic environment may change depending on the location of the sensor and the time of the day. Alexa Guard+ generating FAs for human presence is an example of this. Alexa is providing a sensitivity slider (that can be set by the user) to address this issue. The user can experiment with different sensitivity levels and pick the one that best suits to his environment. But this can be achieved easier with ARE without requiring the user experimentation.
3. **Seismic Sensors:** The vibration characteristics of an area may change depending on the location of the sensor and the time of the day. This may be a source of FAs, which can be reduced with the help of ARE. If the sensor is placed close to a highway, and if there are a lot of vibrations during daytime due to traffic, then ARE can figure this out and reduce FAR.
4. **Smoke Alarm:** Smoke will be more common during cooking time. Frequent FAs can be avoided especially when somebody is in the kitchen. If we have another system that detects Human Presence in the kitchen, then ARE can help reducing such redundant alarms by leveraging the Human Presence information.
5. **Pipeline Security:** This is an example of a multi-sensor (acoustic, seismic) environment, and the acoustic and seismic characteristics of each location may differ at different times of the day. We may need to track certain sequence of events, such as, truck approaching, truck stops, footsteps, digging the ground, contact with the pipeline, and truck drives away. ARE can learn which sequence is associated with a theft activity depending on the location.
6. **Border Security:** This application is similar to pipeline security but may also involve EO/IR cameras, and possibly aerial imagery (UAVs). These types of multi-sensor systems with many smart subsystems can provide more information to ARE, which in turn can discover more complex patterns that may be hidden from the users of the system. ARE can help identifying such patterns and using them to reducing FAs in this environment through personalization.
7. **Internet of Things**: IoTs are another example of multi-sensor applications. We need to look at more concrete examples, but the sensing characteristics of each IoT may change depending on the location. City surveillance and traffic monitoring may fall into this category.
8. **Autonomous driving**: Self-driving cars can be treated similarly to IoTs not because a car has numerous sensors, but because the sensing characteristics of the sensors on a car will be different at different locations, on different weather conditions, and at different times.
9. **Robotics**: If we put a robot in a house, such as Roomba, each house will have different layout, and different acoustic environment that we may need to adapt. A moving robot also has similar characteristics to self-driving cars in that the location of the robot in the house will impact its sensors, and may be associated with different sources of FAs.
10. **Medical**: Artificial organs, personalized medicine.
11. **Education**: This is probably a totally different use case. Personalized education is an established market with billions of dollars flowing every year, and ARE can support AI-led or AI-assisted education products. Consider drafting a separate Positioning document for General AI. Listen to the start of Eric Schmidt’s podcast with Lex for justifying the need for a “general” approach.

**Hybrid Demo 1: Anomaly Detection in a Multi-Sensor Environment**

Consider a multi-sensor application, where each sensor monitors an environment. In the simplest case, we may have two sensors (visual and acoustic); whereas in more complex applications, we may have n sensors monitoring the environment in a different modality. In the normal operation, the sensors agree with each other. For example, if the camera detects a person walking then the acoustic sensor detects footsteps, vice versa. If the sensors do not agree with each other than this is an anomaly condition, and the user needs to me notified.

We are receiving the sensor data as a stream. We are asked to develop a general anomaly detector which can learn to detect anomalies from data regardless of the number of sensors and how the sensors are reporting and what.