Supervised Learning as a Method of Robotic Repair

Kordel K. France
Whiting School of Engineering, Johns Hopkins University
Baltimore, MD, USA
Email: kfrance8@jh.edu

I. Introduction

The next generation of robotics will have an emphasis on collaborative problem solving. As humans, we each have our fair share of handicaps that limit us in some form and differentiate us from others, yet we work collectively in a team toward a common goal every day. Even upon injury, humans still carry out day-to-day duties. Artificial intelligence (AI) and robots should be no different. An additional layer of resilience is necessary, in order to expect a robotic agent to collaborate robustly in adverse environments, even upon damage. One way to solve this is by leveraging principles of biological neuroplasticity. Neuroplasticity refers to the brain's ability to adapt to change by modifying its structure and functionality. Many efforts have begun to define the landscape of neuroplasticity in artificial intelligence [6] [7], and others [22] have shown great success in achieving state-of-the-art performance with a single model across multiple data modalities, but it is to my knowledge that little published work exists on attributing explicit effort toward the real-time repair of damage. In the following framework, two key related concepts are demonstrated: (1) damaged hardware does not *entirely* compromise software and firmware associated with that hardware—the software can be repurposed for other tasks so long as it is dynamically programmed to do so; and (2) trained machine learning models unaffected by damage can provide supervised learning over models associated with damaged hardware in order to perform this repair. I demonstrate the framework for one full system and propose a design for two more systems in order to replicate results and show scalability, and I use the 6D-framework to do so.

II. Decomposition

A. Defining the Problem & Proof of Concept

Just as humans have their five senses, robotic sensory modalities can be broken down by different types of stimulus[1]. Damage can be assumed to be a handicap in functionality to one or more of these

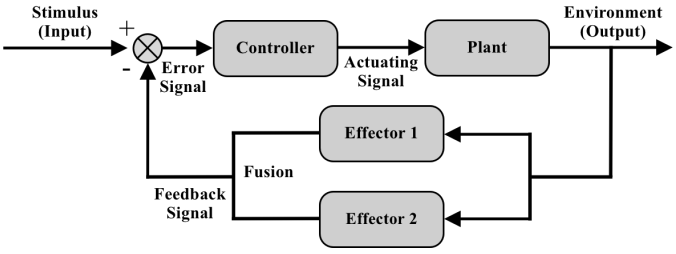

Figure 1 - The simple feedback loop that coordinates motor control in response to environmental input.

[1] Since this effort was inspired by human biology, it is prudent to have human anatomy as the vehicle to illustrate the significance of the results of the proposed experiment. Trauma to the human brain can sometimes result in permanent or severe cochlear damage. In some instances of this occurring, the auditory cortex (the part of the mammalian brain that processes audio) will slowly begin lending processing resources to other parts of the brain since they become dormant otherwise. Senses whose sections of the brain borrow from these resources gain new synaptic connections and can experience enhanced capabilities as a result. So a severe compromise in hearing can eventually manifest augmentations of other abilities, and those parts of the brain unscathed by the trauma eventually come in to colonize cells in the damaged region [1], [5]. I attempt to synonymize this principle between two neural networks and two different sensor modalities.

senses. A concept of self-repair can be shown as an attempt to restore functionality to the damaged subsystem or to repurpose still-functioning resources previously used by the damaged subsystem. To demonstrate self-repair through neuroplastic behavior, sensors with two different data modalities are leveraged for the experiment. This allows for experimental control while also maintaining a highly tractable scenario in robotics. The reasons for the selection of these sensors are detailed in the following sections. See *Figure 2* for a high-level introduction to the design.

A general framework for an autonomous system that incorporates this self-repair concept is defined within the following paragraphs. All of the rules, methods, and algorithms should generalize to data modality. In order to demonstrate the framework in action, a simple system is constructed using vision and audio as the data modalities. This autonomous system has the ability to perceive and respond to visual and auditory stimulus via cameras and microphones. Consequently, it coordinates self-navigation through a simple control loop that responds to feedback from both sight and sound. The system does not have motor hardware since it is not needed to demonstrate the concept of self-repair, but motor control is present and a feedback loop is constructed to facilitate control in response to stimulus. This simple feedback loop is shown in *Figure 1*.

B. Semantics and Terms

It is helpful to define a few terms moving forward to ensure consistency of understanding throughout the following paragraphs:

1. *agent*: defines the entire system as it exists in its environment, including hardware, software, both neural networks, and the control loop.
2. *damaged model/network, training model/network*: the computational model whose input sensors experience damage and is eventually retrained via supervised transfer learning by an undamaged model.
3. *retrained model/network, repurposed model/network*: the damaged model fully retrained through transfer learning, ready for deployment again.
4. *ground-truth model/network, supervising model/network, training model/network:* an undamaged computational model whose input sensors are not damaged and provides supervised learning of a damaged model.
5. *multi-modal data processor*: a part of the processing pipeline / control loop that fuses multiple data types together in sync with time.
6. *data augmentation engine*: a software module that augments data provided by the supervising model to be used for transfer learning.
7. *demonstration system*: a proof-of-concept agent that facilitates self-repair and is demonstrated with the proposed framework.
8. *Self-Repair Framework*: the unifying concept of all constituent models and principles defined in this paper.

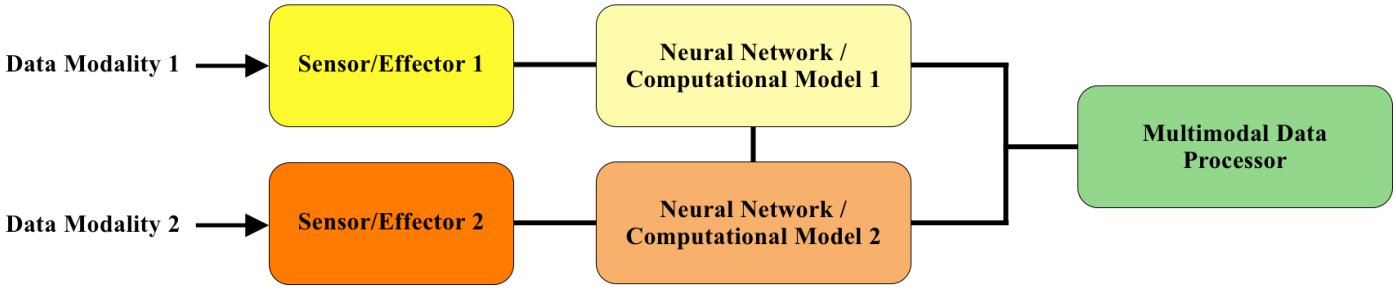

Figure 2 - A block diagram illustrating a high-level overview of the Self-Repair Framework architecture.

For each section, I establish guidelines for a general system and then describe the architecture of the demonstration system, explaining reasons for my engineering selections along the way.

C. Project Mission and Design Limits

Self-driving cars are typically equipped with at least radar sensors and cameras, and sometimes with lidar sensors as well. If a self-driving car receives hardware damage to, say, the radar sensors on the front bumper, and the car is able to "repair" itself by leveraging software resources to provide additional accuracy in the front-facing camera and lidar system, passengers will still likely not feel comfortable riding inside the car. The proposed framework fails in this scenario since full functionality is not likely to be restored with repair. This *Self-Repair Framework* really gains merit in high-stakes situation where receiving some functionality of an autonomous agent (even sub-par functionality) is better than receiving no functionality whatsoever. This is the primary motive for developing this concept of repair. Applications that exemplify this scenario well are military applications where damage to hardware is much more probable and there is an increased need for robustness. Other applications that could benefit from this framework are those that utilize swarm theory and those within the aviation, maritime, medical, and space industries.

III. Domain Expertise

This concept of self-repair is most valuable in domains where multiple data modalities are used to carry out a task. I provide general guidelines for design of a system that utilizes the framework, but I demonstrate the concept in the autonomous navigation domain. Spatiotemporal awareness is crucial to perform any sort of localization, navigation, or mapping task whether robotic or human. For the next generation of robotics, such systems should be engineered with an additional layer of resilience such that damage to one agent of a swarm does not compromise that agent (or the swarm) entirely, and that it may continue working toward its predefined goal nearly unabated. In general, rapidly acquiring data within this domain requires cooperation between multiple senses, or at least requires a single sense with multiple points of input (e.g. vision-only guidance with multiple cameras for depth perception).

A large contributor to the success of artificial intelligence in the 21st century is its ability to respond to dynamic environments by fusing multiple sensor modalities. To this point, I demonstrate the proposed concept through the utilization of multiple sensor types because these multimodal sensor systems are very common within the field of AI; they are also subject to high attention in research. A good scientific claim provides reproducible results, so I evaluate this framework over multiple conceptual instances to demonstrate the duplication of efficacy. Multiple permutations of sensor modalities were evaluated. Refer to Table 1 for the complete list. A trade study allowed the down-selection of three combinations to isolate for this concept with the justifications provided below. In the end, the first combination was determined the most practical and easiest to demonstrate. The evaluation of the other two concepts are provided in the supplementary materials section.

1. Vision - Sound (defined and demonstrated): This was the simplest concept to prove because both vision sensors (cameras) and sound sensors (microphones) are inexpensive and widely used together within robotics tasks, so this combination would be most relatable to AI and robotics practitioners.

2. Vision - Radar (defined in supplementary material): This combination is commonly used within navigation of autonomous vehicles.

3. Vision - Electrochemical (defined in supplementary material): This combination is the

most unconventional, but one that is necessary in order to prove the robustness and versatility of the *Self-Repair Framework*. A camera is coupled with an electrochemical sensor that is designed to detect specific organic compounds [21]. This sensor combination allows an autonomous system to navigate through sight and "smell".

It is worth noting that the intention of this paper is not to elaborate on subsystem redundancy. Building redundancy into software systems is well understood and practiced and this is not an attempt to redefine it. The specific intent of this paper is to illustrate a concept of *repair* for a robotic system that has comprehensive or near comprehensive hardware damage to at least one of its key capabilities. Repair here is not defined as the enabling of an obvious backup subsystem when the primary system fails—it is the recruiting and repurposing of software resources that become idle as a result of critical hardware damage.

IV. Data

A. Data Definition

Data for this experiment shall be leveraged from open resources. Where possible, datasets should be standard benchmarking datasets (e.g. the COCO dataset for image recognition). For the demonstrated concept, vision data was leveraged from the COCO dataset[3]; audio data used to train the audio neural network was acquired through Google's AudioSet dataset [4]. As I shall show in the experimental design, one neural network (the "undamaged" or "supervising" network) shall be used to provide training data and provide ground truth class labels to the other neural net (the "damaged" network) as a form of supervised learning.

B. Data Normalization

The process of one neural network learning from another is known as transfer learning. In order for this to occur, both neural networks must effectively be interoperable, with the inputs and outputs being identical. For example, for a model allowing transfer learning between audio and vision data, sounds and images do not compose the same data medium so one must be transformed into the other. To accomplish this, sounds are transformed into spectrographs and processed as images. Audio processing then becomes a computer vision problem. This method has been proven in multiple other works and shown great success in sound recognition. [8], [9]. Both networks now obtain identically sized inputs and produce identically dimensioned outputs to allow for effective transfer learning. As an example, *Figure 2* shows a series of sounds represented as spectrographs. This same principle will be applied to every sensor modality, not just audio networks—radar, electrochemical, and lidar data can all be transformed into a high-definition spectrograph or similar to allow for this interoperability that will lend the system to self-repair. Put simply, images are a great vehicle for communicating multiple data modalities and all sensor data are converted to images for the purposes of this framework.

Table 1 - Evaluated Sensor Combinations

#	Modality 1	Modality 2	Knowledge Point
1	Vision	Vision	Depth perception
2	Vision	Sound	Navigation
3	Vision	Radar	Navigation
4	Lidar	Radar	Navigation
5	Lidar	Vision	Navigation
6	Sound	Radar	Spatial Awareness
7	Electrochemical	Vision	Navigation
8	Electrochemical	Electrochemical	Passive monitoring
9	Electrochemical	Radar	Navigation
10	Electrochemical	Lidar	Navigation
11	Vision	Ultrasonic	Spatial Awareness
12	Sonar	Vision	Navigation

Another advantage to using spectrographs in this scenario is to allow efficient synchronization of multimodal data processing. By maintaining identically sized inputs and outputs, it can be verified that results of data classification at one time step correspond to the results of other data classification at the same time step. One can see

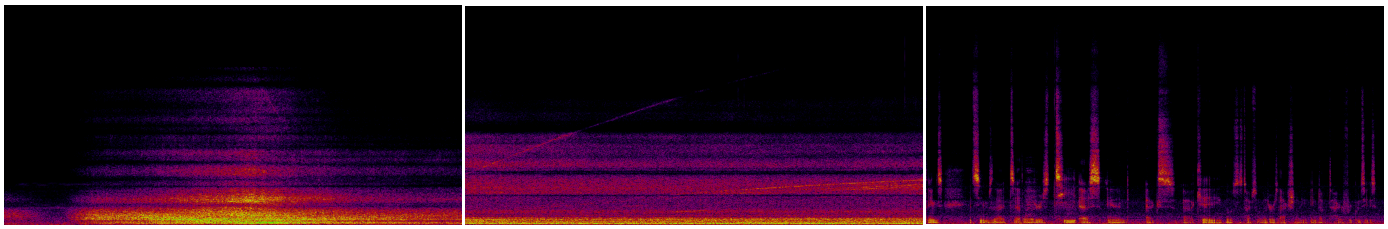

Figure 3 - From left to right, spectrographs for the sounds of a passing sedan, a jet engine starting up, and a human talking are shown.

how desynchronization of the processing pipeline can confuse the program in conducting adequate situational awareness by, for example, hearing an airplane, but seeing only a dog. For the demonstrated system, while the audio data acquired is from open-sourced resources, the spectrograph images are generated from a custom algorithm as part of the source code associated with this experiment.

C. Data Augmentation

A brief trade study was performed to evaluate whether a larger dataset should be acquired for model training or if a more sophisticated data augmentation pipeline should be constructed. It was ultimately determined to develop an elaborate data augmentation procedure since this more closely matches the "self-repair" process that occurs in the project later on. Specific augmentation techniques shall be used according to the modality of the data being augmented. For example, methods should be leveraged to vary the color (decibel level) of spectrographs, but other typical techniques should not be employed since flipping, mirroring, or rotating the spectrograph image produces a fundamentally different sound in the context of audio. This holds true for radar and lidar data as well, so similar decibel-varying techniques were used over these modalities.

D. Data Storage

The *Self-Repair Framework* really sees the most utility in offline edge scenarios. Therefore, the data storage system shall be stored on the system itself without the need for online access. Several advantages are gained from this, but speed is the primary benefactor. Transfer learning and data augmentation can occur at a much faster rate by not having to handle the latency of the cloud. I select to utilize a revolving storage approach in which older training data is deleted from storage once it has been "seen" in training.

A baseline image is acquired by the undamaged sensor and its data stored for later augmentation. Where applicable, it was selected to save raw time-series files instead of the actual spectrograph images to allow for dataset compression. Spectrographs are generated from the data as they are fed into the augmentation pipeline. The data for the baseline image is always saved so that it can be used for augmentation and training in the future, but that image's augmented counterparts are deleted after the training loss of the undamaged network converges. This relieves the edge storage mechanism of memory and positions it for the next self-repair event[2].

[2] I briefly investigated the impact of this storage and data augmentation procedure on the overall functionality and cost of the system design. It is prudent to incorporate extra storage (and perhaps even an auxiliary GPU for the data augmentation) into the system architecture from the beginning to accommodate for the need to store and generate all of this extra training data. This could obviously prove cumbersome for an edge computation system that is already resource-constrained. The alternative route is to facilitate transactions with cloud storage in order to generate and download training data. Google Cloud's Firebase or AWS's Lambda both work as great candidates for this. However, an edge system will likely have idle resources while it is "damaged" so this frees up some access to on-board storage and a GPU to facilitate repair. Once repair has ended, data augmentation and storage of training data is no longer needed, so these resources may be leveraged for their original design intent.

V. Design

A. Guidelines for General Design

There are few components that are critical to consider for the design of a system capable of facilitating the *Self-Repair Framework*:

1. Define the supervising model. This can be a trained neural network acting as the labeler for training data or some other computational resource providing ground truth data.

2. Consider where computational resources for the generation of training data will come from. Preferably, there is a small auxiliary GPU that is designed in the system to handle this.

3. Ensure effectors / sensors have write capabilities so that they are able to save data for the generation of training data for the damaged network.

4. Evaluate whether the system can exist fully at the edge (offline) or if the constrain of resources is so great that online transfer learning is needed.

5. Define a neural network architecture that can be repurposed if needed. For example, for this project, I elected to use the YOLO architecture for object detection and a VGG-style architecture for classification of data across all object detectors and neural network classifiers. It is valuable to have a common architecture among all data modalities if possible, but not necessary. Obviously selecting a neural network architecture that maximizes the accuracy of the intended modality is priority; re-training this network on a differing modality may require experimental fine-tuning as a consequence in order to obtain desired loss and accuracy levels[3].

6. Define exact rules that allow the software to determine when a system is "damaged".

7. Determine an exact timeframe that is needed for the agent to assess dysfunctional sensors and confirm that damage has occurred. This is known as the *damage assessment period* and has value for two reasons: (1) it allows the system to confirm that actual damage has occurred as opposed to a lack of stimulus occurring or irregular noise patterns, and (2) it emulates a practical scenario in which an autonomous system would be afforded time to adjust a mission plan in light of this new hardware failure.

8. Define a data augmentation plan and the minimum amount of training data needed to retrain a neural network over each data modality of the system.

9. Define a threshold for either the minimum training accuracy or maximum cumulative training loss needed by the retrained neural network before it can re-enter deployment.

10. Define whether damaged models will work in parallel or in series with the undamaged models once retraining of the damaged models is complete.

11. Define a quality control or monitoring procedure that allows surveillance over the damaged model's performance after it is retrained to ensure that accuracy is maintained.

Since every data modality is converted into imagery of various sorts for the utilization of the *Self-Repair Framework*, it is helpful to define what that architecture looks like as inspiration for other approaches. For the demonstration system, I elect to use an object detector followed by a neural network

[3] A flavor of ResNet would be a good candidate for a common architecture among different data modalities due to its implementation of skip layers. These layers allow the more malleability of the network to be repurposed for a new task.

classifier as the pipeline for data. An object detector can be trained to identify very specific patterns within an image (e.g. locating vehicles within a parking lot) or spectrograph (e.g. locating noise patterns of an airplane engine within ambient environment). For some scenarios, this may be enough to gain the accuracy needed. For those in which an object detector is not enough, the classifier gains merit in identifying exactly what the object is within the image (e.g. what kind of vehicle was detected) and spectrograph (e.g. what kind of plane engine is detected). A design feature should be engineered that allows the classifier to be enabled or bypassed depending on the additional accuracy it provides and this can be a hyper parameter determined through experimentation. It is easy for one to see how a similar approach can be extrapolated to other data modality combinations as well.

B. Feature Selection & Feature Engineering

The features that the model will select for detection and classification are up to the application designer with good domain expertise. For the demonstration system, the features selected for the damaged model were identical to those established for the supervising model. This does not always have to be the case and it is important to identify this in cohesion with the model architecture in order to define whether or not some other computational model needs to be integrated in order to help the damaged model engineer the intended features from the training data.

C. Hardware and Programming Languages

The number of cycle times per second—or the refresh rate—will depend on the selection of hardware. This can pose challenges for resynchronization of hardware once damage occurs. For example, a camera may have a frame rate of 120 frames per second and an inference rate of 20 milliseconds when integrated with one neural network. Adding an additional neural network may cause inference time to increase due to the extra computational time needed. This may have drastic impacts over the entire system. Identically, having the undamaged network provide supervised learning over the damaged network will have similar effects. This is why it is important to specifically de-couple the data augmentation engine and the multimodal data processor into their own modules so that they can be desynchronized from the frame rate of the camera and adjusted to allow for enough inference time. Additionally, after the damaged network is retrained and working alongside the undamaged network, instead of running both neural networks in parallel, they can run in series such that each network infers every other camera frame. If retraining of the damaged network occurs correctly, there should be no desynchronization of the clock.

D. Programming Languages

While the *Self-Repair Framework* will see the most merit in hardware-based languages, it is agnostic to programming language overall. However, the guidelines established here are done so with sensor-level code in consideration such as C, C++, Python, Rust, and assembly languages. Java, Kotlin, and Swift were also considered for defining this framework for the iOS and Android operating systems.

E. Ethics and Bias

For any autonomous system, it is important to address any opportunities for ethical violations to occur within both the algorithm and data. Since the *Self-Repair Framework* is autonomous in nature, any ethical issues of the framework are directly dependent on any ethical issues of the system overall and are not necessarily sourced from the framework itself.

Although the framework does not exhibit many opportunities for ethical violations, several exist for bias. Bias is any rule within the system that, all other parameters held constant, handicaps it from performing all actions with equal probability. The biggest source of bias that exists comes from the

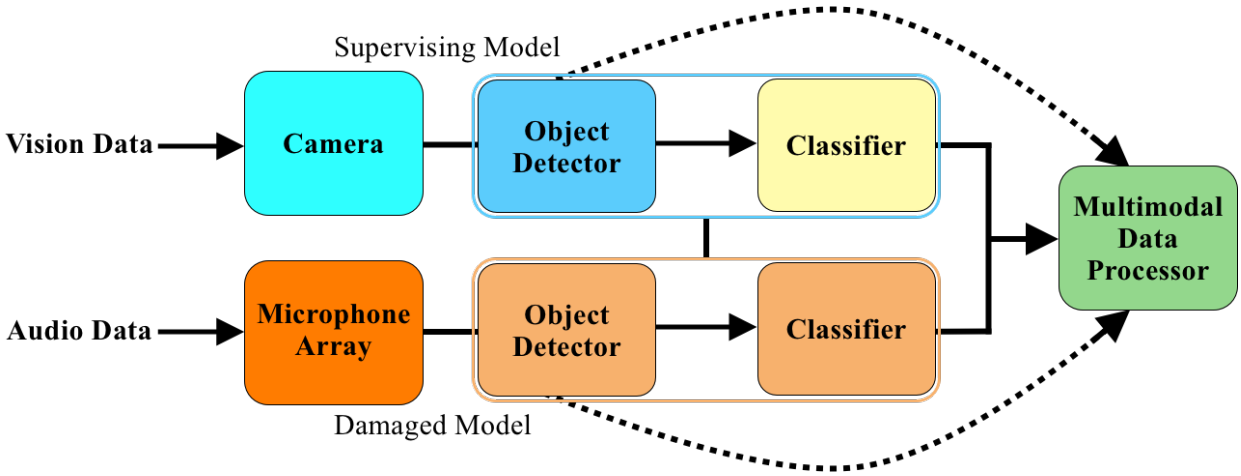

Figure 4 - A block diagram illustrating the interconnection between the audio and visual models and their associated hardware. The dashed line indicates that the data is allowed to bypass the classifier if it is deemed that the object detector provides enough accuracy without it. Notice the resemblance to Figure 2.

construction of training data by the supervising network. If the agent does not experience an environment in which all training labels are equally encountered, the supervising network will be biased from the start of training and likely never converge (e.g. a system exposed only to highway traffic can't possibly be trained on a well-rounded dataset if "aircraft" is one of its classification labels. It will train erratically and provide false indications of system performance. This could cause the system to misclassify data, fail to safely navigate itself, or provide invalid feedback to other users and agents. This is an important issue to address and there are a few immediate ways to do so:

1. Disable the damaged network from training until all training labels have been encountered. Store training data until the number of all training labels is equivalent (or near equivalent) and above a certain count.

2. Trim the damaged network outputs such that only the outputs (labels) that have been encountered are trained through transfer learning. In other words, disable all labels that do not receive training data from the supervising network. This can be done in a variety of ways, but [SOURCES] define some intuitive methods. This adds another layer of complexity in transfer learning since the architecture changes, but doing so still satisfies the main goal of this framework: *some functionality is better than no functionality in high-stakes situations*. Granted, it is important to ensure that this functionality is unbiased, correct, and useful.

3. Build in autonomous unit tests that can test a retrained model before deployment. These can be simple benchmarking tests to gauge the model's accuracy.

4. Elongate the transfer learning period. Simply allow transfer learning to occur over a very long period of time with a small learning rate to allow the model to slowly converge.

One major advantage to having the supervising neural network is that it will always be able to help gauge the disparity between actual and expected values from the retrained neural network, essentially performing a level of quality control throughout the remaining life of the retrained model. This alludes to the method of handling system failure that satisfies guideline 11 and will be discussed in the next section. Other standard practices for relieving bias from an autonomous

system still apply, such as a well-rounded review of training data for the pre-trained undamaged models by domain experts, engineers, and third-party reviewers alike.

F. Design of Demonstration System

The system used to demonstrate self-repair is based exactly on the framework described above. To accommodate the hardware necessary for a camera, microphone array, and processor that can handle the computation needed to illustrate the framework, the demonstration agent, in this case, will be a mobile application hosted on a smart tablet. The required hardware and computational power is readily available in modern tablets which allowed the focus to remain on replicating the principles of neuroplasticity in software code rather than handling drivers and nuances around wiring multiple sensors and processors together that communicate with different programming languages. The software principles behind robotic repair are the concentration of this paper and there is added value in showing the versatility of such a concept on off-the-shelf hardware. In light of this, the experiment was run on a smart tablet since it contains a built-in microphone and camera. Specifically, the tablet was a 2020 Apple iPad Pro. iPad's are easily programmable and Apple's iOS operating system provides an easy vehicle to take advantage of the onboard hardware through app development. An iOS mobile app shall be the platform to demonstrate the target concept. Guidelines 2 and 3 are satisfied since, respectively, the iPad has a GPU that can aid in data augmentation for transfer learning, and relevant sensors have write capabilities due to permissions granted by the iOS operating system. I elect to have the vision neural network act as the supervising model over the audio neural network, which will be the model receiving damage. The entire software build including all algorithms and the control loop shall be denoted as the "demonstration agent" and the tablet running the app as the "demonstration device" throughout the rest of this paper.[4] The general framework for repair presented here is agnostic to platform and programming language. However, the programming languages used for this experiment are largely C++ and Swift; the neural networks are programmed and initially trained in Python 3.7, and then exported to a format interpretable by the iPad ARM processor.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 5 - YOLO Darknet-53 architecture. One example of a common object detector that may be used as the baseline detector for the data pipeline.

[4] Since the inspiration for this experiment is derived from biological neuroplasticity, I investigated using a more analogous setup to a human (or at least a humanoid robot) with two separate cameras (eyes), two separate microphones (ears), separate processing chips for different processing regions in the brain (V1, V2, LGN, etc.), and even a speaker (mouth) to vocalize feedback. However, finding compatible hardware that could be coded in the same language, run off of the same clock, and return data in near real-time quickly became more difficult than was necessary to explain the fundamental scope of this paper. Consequently, the resultant hardware of the Agent was deduced to something readily available such as a smartphone or tablet. The principles behind the neuroplastic concepts remain unchanged by utilizing such hardware since the intended neuroplastic behavior is effectively a "software" effect anyway.

I elected to use one object recognition algorithm to locate objects within the camera field of view. This object detector follows the YOLOv3 design with the Darknet architecture, but others could be used in substitution. Ultimately, this is a design tradeoff that should be analyzed prior to development. Other architectures such as Fast-RCNN may be used, but YOLOv3 was selected for the demonstration agent due to its fast inference speed and generalization capabilities. The Darknet architecture is a 53-layer convolutional neural network as specified in *Figure 3* [2]. Once an object is detected in the camera field of view, the camera frame is cropped to the size of the bounding box proposed by the detector and another convolutional neural network classifies the bounding box. Structuring perception in this hierarchical manner allows one to transfer learn a simpler neural network instead of an entire 53-layer object detector [15], another design advantage. It also allows one to have a very generalized object detector that does not need extensive training to obtain acceptable accuracy results to start and that may be fine-tuned. Once again, I leverage the same approach in evaluating audio patterns— once a specific sound family is detected within the audio pattern, the waveform attributed to that sound is cropped within the spectrograph. This cropped waveform image is then sent to its respective classifier in order to detect exactly what type of sound was detected. The architecture for each of these neural networks is identical and may be viewed in *Figure 4*. These selections satisfy Guideline 5. Both networks are trained using a Nvidia T4 GPU and exported to a format interpretable by the tablet compiler. A block diagram detailing how each neural network relates to its corresponding hardware can be seen in *Figure 5.* The class labels for the object detector and neural network classifier are shown in *Figure 7.*

Training takes under two regimes—one for the object detector and one for the neural net classifier. Imagery coming in from the camera is fed directly to the data augmentation engine, with the supervising model providing labels and bounding box coordinates for the object detector training data. The supervising model also creates a copy of the same image but crops this copy around the detected object, providing this image with another (more detailed) label along to the data augmentation engine.

In order to effectively perform transfer learning of the audio neural network, it was determined that 100 images each of 10 distinct viewpoints would be needed for each of the 10 object classes. This constitutes to 10,000 images needed *prior* to data augmentation. I satisfy Guideline 8 by leveraging standard data augmentation techniques [16, 23] to multiply each instance of the dataset 12 times accommodating different reflections, distortions, contrasts, and angles of the original training image. At an average image size of 26 KB, the total space needed for transfer learning was 26 KB per image * 12 augmentations * 10,000 images = 3.12 GB. The tablet had more than enough RAM to handle this, so the entire system can function fully offline which satisfies Guideline 4. However, I took a different approach to transfer learning in that training

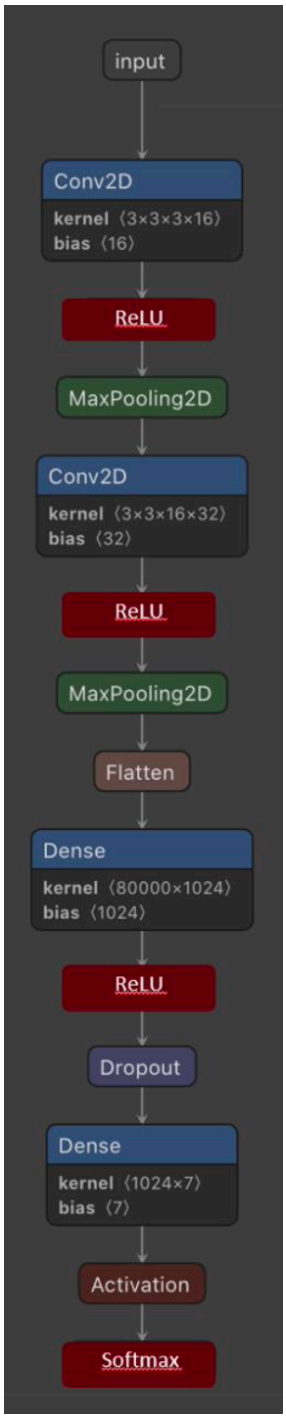

Figure 6 - Architecture for both the audio and visual neural networks.

occurred through batch processing—once the supervising network provided at least two images of every class, the images were augmented and the damaged model was trained on that batch. That batch of images was then discarded and relieved from RAM once the training accuracy for that series of epochs either achieved a threshold greater than 80% or showed an average of at least a 10% degradation in loss from epoch to epoch.

My damage assessment plan for Guidelines 6 and 7 was simple. The weights files of both convolutional neural networks will be noted prior to initialization of the system to ensure that transfer learning occurs successfully. Once hardware damage occurs, the agent will wait 30 seconds to confirm that damage has indeed occurred and that there is not simply a lack of stimulus to the microphones. The damaged neural network will then need a way to acquire ground truth labels for its repurposed task—the undamaged neural network will provide such ground truth labels and coordinate supervised learning over the damaged network. The damaged neural network will then transfer learn from the unaffected network until testing accuracy in classification passes a threshold of 80%, satisfying Guideline 9. Once this threshold is achieved, the damaged model is now fully retrained and considered deployable. I will elect to have this newly trained model work in series with the supervising model such that each model sees every other camera frame, satisfying Guideline 10.

To begin, the Agent is powered "on" and remains idle. The Agent is linked to a laptop computer to allow diagnostics to be performed on its behavior. A camera on the back end of the device begins streaming visual data while the microphone array begins streaming audio data. The multimodal processor is fusing these two data mediums together and monitoring for significant events to add to its memory. While this is occurring, the Agent is also printing out a log of everything it sees and hears and any associations it determines in its environment. The intent of this is to add (1) an element of explainability and (2) a method for debugging such that the Agent describes its surroundings and what it perceives about its environment. After 300 seconds, the connection to the microphones is terminated, emulating "damage" to the auditory system. At this point, 60 seconds of time is allowed for the Agent to comprehend the damage and determine how to adjust its resources. This period of time is attributed as the *Damage Assessment Period*. This period is allowed for two reasons: (1) it allows the Agent to confirm that actual damage has occurred as opposed to experiencing a muted microphone or irregular noise patterns, and (2) it emulates a practical scenario in which an autonomous system would be afforded time to adjust a mission plan in light of this new hardware failure. Following this period, a multimodal data processor communicates to the vision model that resources from the audio model are no longer being utilized and its resources are being made readily available due to the auditory damage. The vision model slowly begins recruiting resources from and reorganizing the audio model, noting that the audio model is now dormant. From here, it is shown how visual capabilities begin to increase with additional resources from the audio model.

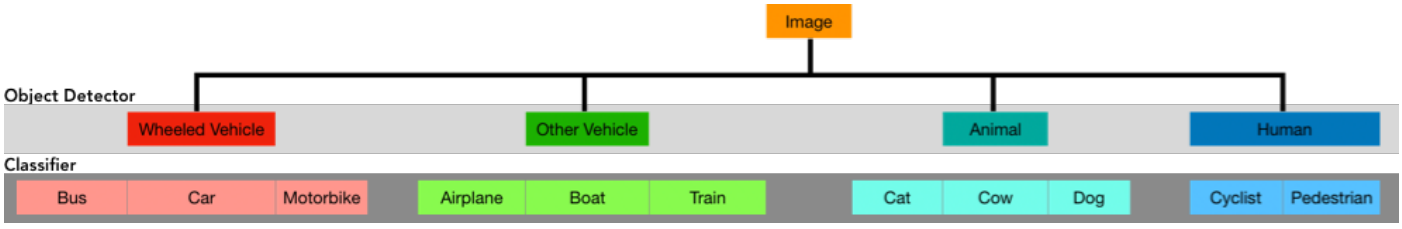

Figure 7 - Diagram of the hierarchical structure of the vision data labels.

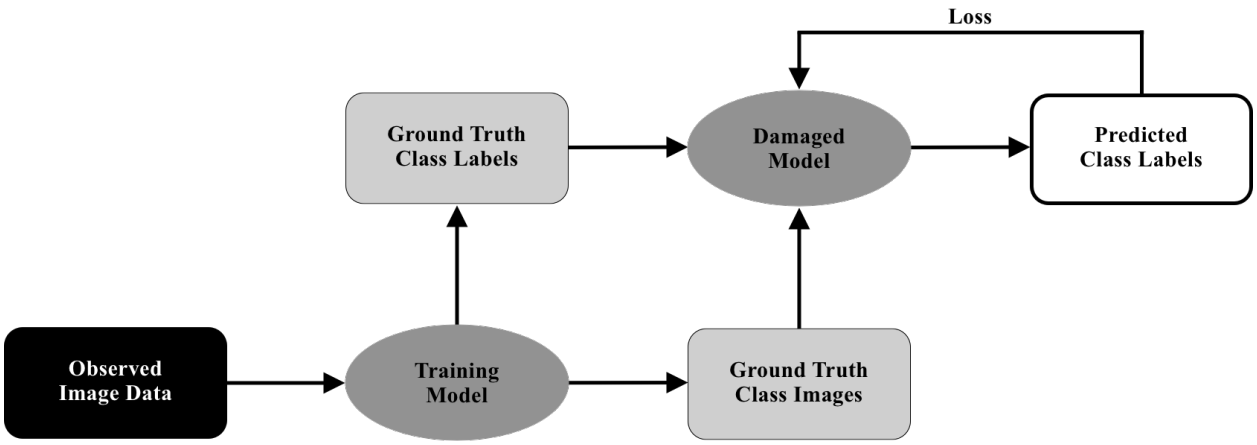

Figure 8 - One full update cycle of the Self-Repair Framework used to coordinate transfer learning for a damaged model. Notice the similarities between the discriminator and generator models within the GAN, but also the key differences in the weight update cycle. The Damaged Model is analogous to the Discriminator Model in the GAN, while the Training Model is analogous to the Generator Model.

In a more practical scenario, there is likely more time allotted for nominal performance before damage and the damage assessment period. I elected to accelerate the timeline of this process in the experiment for convenience, comprehension, and to show how the *Self-Repair Framework* can be leveraged in quick fashion.

VI. Diagnosis

A. Diagnosis of General Design

Finding the right hyperparameters for retraining the damaged network may take some experimentation. If the environment permits, one may even allow a slot of time to the agent to test certain permutations of hyperparameters and ascertain results. With the supervising network as oversight, it can easily be spotted if training of the damaged network begins to diverge. If the neural network of the damaged model is the same architecture as that of the supervising model, then a good start for training would be to start with identical hyperparameters as the supervising model. I select to adjust training epoch count according to the following loss function:

$$L_{tl}(y', y^*) = \underset{N}{\operatorname{argmax}} \left(\frac{1}{m} \sum_{m=1}^{m} y_i' - y_i^* \ , \ \delta \right) \tag{1}$$

Where L_{tl} denotes the loss of transfer learning between the two neural networks, m denotes the number of samples the network was trained on over each of the N epochs, y_i' denotes the classification result of the ground-truth model at the i^{th} sample, and y_i^* denotes the result of the re-trained model at the i^{th} sample. The goal of the re-trained damaged neural network is to minimize the loss L_{tl} between both networks and truncate transfer learning when the average change in loss between epochs no longer exceeds a certain threshold δ. Once this threshold is met, or N epochs have been completed in retraining, transfer learning should cease and the model tested on a different dataset. Successful convergence on this dataset over the same loss function results in the retrained model being deployed cooperatively with the training model and both should be considered "live".

During development, it was found that this loss function generalized well but may, in some instances, fail to produce classifications with high

confidence. In particular, some scenarios showed that, at the conclusion of transfer learning, both models produced identical classifications over the same imagery, but the retrained model produced them at significantly lower confidence values than the training model. This outcome warranted the need for another term within the transfer learning optimization that ensured the loss between the classification labels of both models was decreasing while the confidence score of those class labels was being optimized. In light of this, I include the following term within the optimization function:

$$V_{tl}(y', y^*) = \underset{N}{\operatorname{argmax}} \left(\frac{1}{m} \sum_{m=1}^{m} c_i' + c_i^* \right)$$

(2)

Where V_{tl} denotes the value of combined confidences between both neural networks, c_i' denotes the confidence of the classification result of the ground-truth model at the i^{th} sample, and c_i^* denotes the confidence of the result of the retrained model at the i^{th} sample.

Consolidating this all together, I see that the optimization of knowledge transfer between the training and damaged models is achieved by minimizing the loss between classification outputs of both models and maximizing the confidence between those classifications. I can define a final function that denotes the two corresponding terms of equations (1) and (2) together:

$$G_{tl} \longrightarrow G_{tl}(y_i', c_i', y_i^*, c_i^*) = \frac{V_{tl}(c_i', c_i^*)}{L_{tl}(y_i', y_i^*)}$$

(3)

Training ceases when the value of G_{tl} averages a global maximum over three epochs.

The loss L_{tl} between both networks during transfer learning should be considered similar to the loss between the generator and discriminator networks in generative adversarial networks (GANs) [17]. A GAN contains two neural networks that compete in a zero-sum game to manufacture fake (but very realistic) data, such as images. During GAN training, two loss functions are working together to minimize one another in order to generate more convincing data. The principle here is similar in that the training network is assumes the roll of the generator network by manufacturing training data for the damaged model which acts as the discriminator. The only difference here is that, in a GAN, the weights of both models are adjusted at the end of each epoch since both models are being trained—neither one is considered "ground truth"; with the *Self-Repair Framework*, only the damaged model is allowed to be adjusted at the end of each epoch as the training model weights are locked and considered ground-truth. At the conclusion of transfer learning, both the supervised and retrained models may be considered as globally optimal discriminators [17]. The computed loss over the predicted class labels updates the weights in the damaged model through Backpropagation [18].

B. Handling of System Failure

As I show in the *Results* section, it is very possible that the damaged network slowly loses accuracy after training or fails to converge during training at all. The biggest catalyst for this is the possibility that the agent may not be in an environment that allows the supervising network to provide enough of a well-rounded dataset to the damaged network in order to generalize well. In order to diagnose these types of issues, it is helpful to have some sort of monitoring system that can provide a level of quality control over the damaged model. Below are a few approaches to facilitate this and satisfy guideline 11:

1. Simple debugging statements saved to a log file or cloud dashboard.

2. Running a regular unit test procedure that tests known truth labels on the retrained network.

3. Performing a sampling routine where an image classified by the retrained network is pulled and the correct label confirmed.

4. Periodic re-enabling of transfer learning in order to expose the damaged network to a more universal dataset.

For the demonstration agent, I select methods 1 and 3 above. *Figure 9* gives a sample of the user interface of a separate logging application that automates this quality control process and gives real-time feedback.

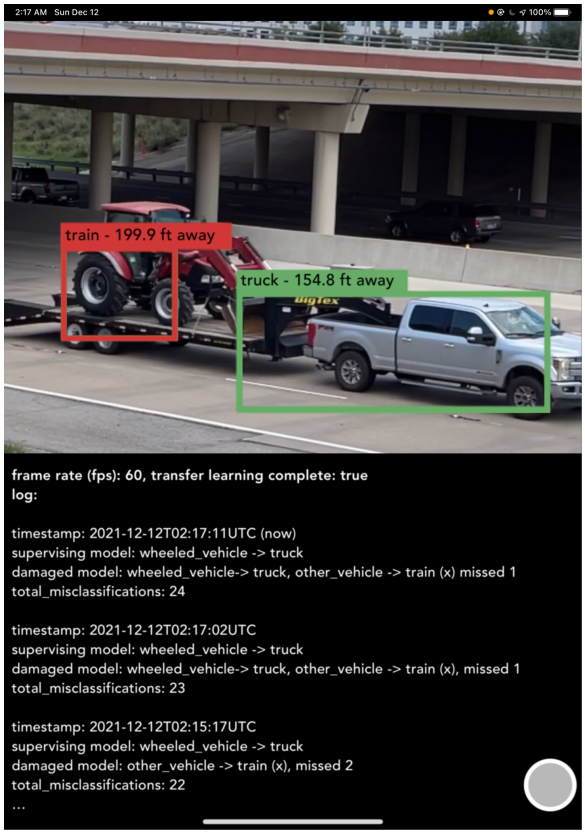

Figure 9 - simple logging application for quality control that checks the error between what the supervising network found and what the retrained network found. Red boxes indicate missed detections by the retrained network; green boxes indicate no error.

C. Diagnosis of Demonstration Agent

The parameters used for transfer learning were identical to the ones used in training the undamaged neural network. As previously stated, each of the two neural networks were identical in architecture. The audio model screened for audio patterns indicative that an object of one of the class labels was present within the camera's field of view; the vision model scanned each camera frame for key features indicating objects represented by the same class labels were present. A timestamp indicating the recognition of both a sound and visual pattern of the same class label provided confirmation that that particular object was indeed near the agent. All images were processed within the RGB channel.

Stochastic gradient descent was selected as the optimizer with a learning rate of 0.001, a batch size of 8, and no momentum. Categorical cross-entropy was selected as the loss function. The audio network was initially trained over 20 epochs while the vision network was trained using 50 epochs. One might notice that these hyper parameters are common neural network training values. This is by design in that it illustrates how neuroplastic effects can be achieved without specific or overly complex architectures and it leaves parameter optimization as a pursuit for a follow-on experiment. The damaged neural network maintained the exact same parameters during transfer learning as it did during training with the exception of epoch count. Epoch count was adjusted according to the loss function defined in Equation 3.

VII. Deployment

The core purpose of the *Self-Repair Framework* is to to provide a blueprint for the reorganization of resources so that *some* functionality can be

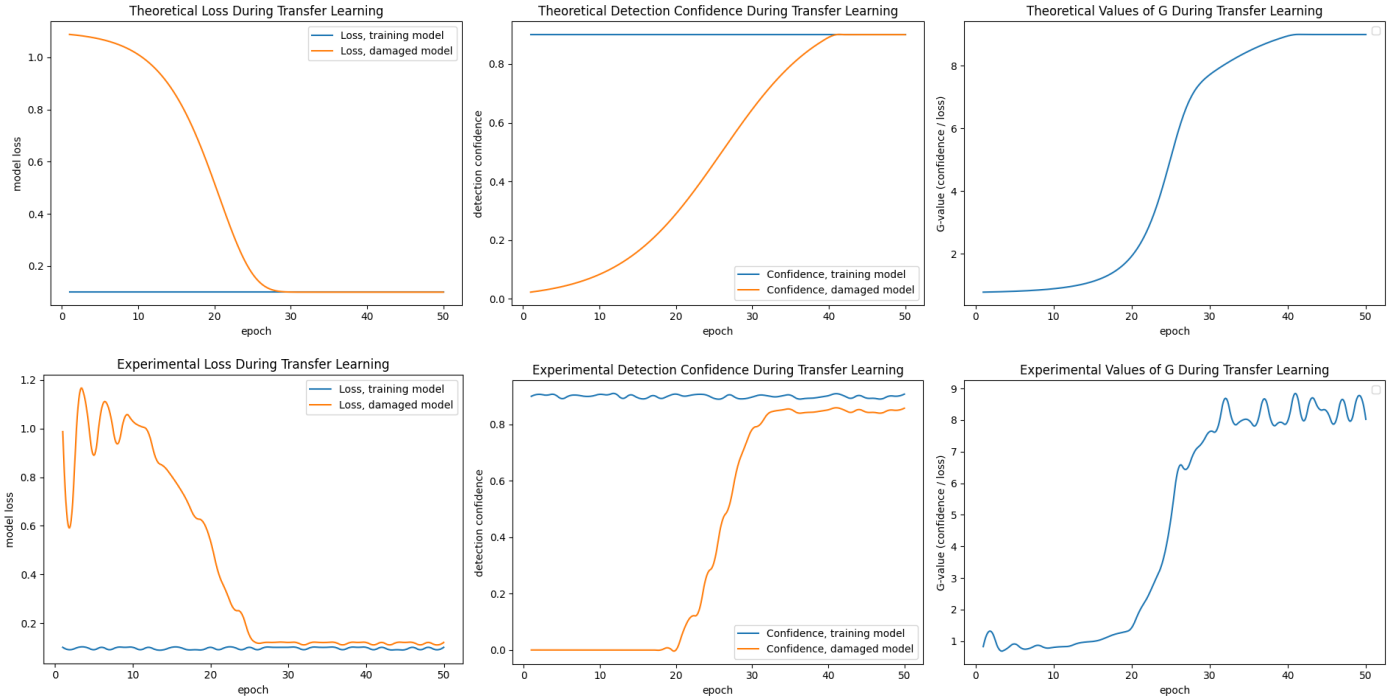

Figure 10 - ***Top:*** *metrics for what a globally optimal model should return if all knowledge from the training model is learned by the damaged model.* ***Bottom:*** *metrics for the actual model returned by preliminary experimentation.*

leveraged in a high-stakes scenario instead of no functionality whatsoever. Deployment of the framework should not be the first instance in which the framework is tested on an autonomous system—all sensor subsystems capable of participating in self-repair should be tested prior to deployment of the system to fine-tune hyperparameters, diagnose training data needs, and ensure the performed repair is actually useful.

To confirm that "self-repair" actually occurred, proof may be visible in three ways: (1) output logged to the console of the computer linked to the device, (2) the change of weights of the retrained model before and after damage, and (3) change in classification scores of the retrained model before and after damage. At the conclusion of the experiment, the weights of the two convolutional neural networks need to be checked in order to confirm that transfer learning did indeed take place.

For my demonstration system, statements indicating performance for the damaged model were programmed to log as the experiment occurred. The diagnostic app indicated in *Figure 9* was also used. During the last five minutes of the experiment, the retrained neural network underwent a final "validation" period in which its accuracy in generalizing over a new domain was evaluated.
The following schedule took place where time T denotes the number of minutes since startup and τ denotes the duration of training time needed to achieve a global maximum G_{tl} during transfer learning:

Schedule 1:

1. ***T* + *0*:** Startup, Agent initialization
2. ***T* + *1*:** Vision and audio models confirm nominal performance
3. ***T* + *5*:** Damage to microphones occur, Damage Assessment Period begins

4. ***T* + *5.5*:** Damage Assessment Period terminates, sensor failure confirmed, control loop self-adjusts, vision model notified and begins generating training data
5. ***T* + *6*:** Damaged audio model begins transfer learning from vision model
6. ***T* + *τ*:** Damaged model ceases transfer learning, vision model ceases training data generation, training data relieved from Agent RAM
7. ***T* + *τ* + *0.1*:** Damaged model retrained as vision classifier and now online, Agent now has two vision models that work cooperatively

Based on the training time, the dataset size, and the number of epochs used to initially train the original auditory neural network, the time to fully perform transfer learning for the damaged model and achieve 80% validation accuracy is expected to take between 60-120 minutes assuming the the training model encounters enough observations in the environment to manufacture training data at least once every five seconds. It is entirely possible that performance of the damaged model completely diverges during transfer learning. This will be identified before the experiment ends due to the logging methods built that are built into the program, and there are precautions built into the program to help mitigate this, such as restarting the transfer learning session and reverting to the pre-transfer model, or by simply adjusting hyper parameters.

VIII. Results

In theory, the loss of the damaged model should exactly converge to that of the training model. One would expect the damaged model to learn all that the training model knows and slowly converge toward a replication of the training model, provided that the architectures are identical. The same can be argued for detection confidence. This cannot actually occur in practice unless the damaged model is trained on the *exact* dataset the training model was initially trained on and under identical hyper parameters. Due to this, the loss and confidence values of the training model may be viewed as an asymptotic upper bound of the damaged model loss and confidence. In other words, G for the damaged model is bound by G for the training model. *Figure 10* shows the evolution of losses, confidence scores, and final values of G under two conditions: (1) as they should resemble if perfect knowledge transfer occurred, and (2) as they were actually experimentally measured through training. While not perfect, these first-order experimental results do show trends that tend toward the same distributions as those of the theoretical which gives good confidence in the general direction of the experimental design. It's worth noting that the same control loop shown in *Figure 1* is employed throughout the entire experimental validation process with the only modification being that the audio sensor is bypassed and no longer contributes to the feedback signal due to damage.

The very first trial runs showed divergent loss around halfway through training, in general. These results are actually what drove the requirement for the optimization of a two-term loss function (3) by adding the optimization function for the confidence values since a single-term loss function was not enough dimensionality to adequately capture model performance during training. After restructuring the code and processing pipeline to accommodate the optimization of this new loss function, training results resembled performance typically exhibited by a converging model. The loss of the model usually converged a few epochs earlier than the model confidence as as viewed in the bottom row of *Figure 10*. Intriguingly, a majority of the optimization for both loss and confidence occurred over the sequence of only 10-20 epochs, both performance measures showing much sharper sigmoid curves than theoretically anticipated. Reasons to this were investigated but not determined, so there exists some room for

optimization of the experimental design to rectify this.

An original training attempt occurred over the course of 71 minutes by positioning the iPad on a camera tripod and having it observe street traffic near a strip mall. This provided a real-world element to coordinate training and all ten output classes were observed, but the classes were not observed in a balanced manner so the resulting model was not optimal. To reconcile this, a subsequent training effort was constructed such that a series of YouTube videos were observed by the Agent. These videos were leveraged from the same Google dataset [4] [19] that was used to train the original audio model by extracting sounds from the videos; however, in this case, the full video was observed, not just the audio. This effort appropriately rectified the issue of class imbalance to achieve the performance shown in *Figure 8* and the time for transfer learning was completed in 42 minutes. Additionally, the loss and confidence of the damaged model never achieved the exact loss and confidence values of the training model, further reinforcing the asymptotic bound the training model establishes on the damaged model during transfer learning. Upon reception of these results, it was confirmed that the devised experiment was minimally viable in assessing the concept of robotic self-repair.

IX. Opportunities for Improvement

The above framework admits several methods for the acknowledgement and coordination of self-repair by a robotic agent. However, there are several techniques that can be used to improve the structure, data pipeline, and training time. Currently, the time for transfer learning may take a matter of hours in order to fully retrain the repurposed neural network to a degree that it is useful. I was able to partially solve this by allowing transfer learning to occur in batches, but batches are constructed at the rate of which the Agent's environment makes such data available. Additionally, batch training in this manner is very prone to bias, based on my experimental evidence, because the Agent can't possibly encounter a dataset that is adequately representative of all data labels in a short amount of time. I believe malleable neural networks with skip connections such as ResNets show great promise with the framework and can help soften the issues with bias.

Admittedly, there are more effective ways to communicate this framework than the demonstration system detailed by this project. However, the resources needed to create a better demonstration began to quickly approach an amount of work that was not feasible for the length of time allotted for the project. Even so, one can imagine how this concept can be expanded to a more complex situation where a robotic agent has critical damage to more than one type of sensor. In this situation, how does this self-repair framework work? A subroutine built into the agent beforehand could triage damage in such a manner as to allocate resources hierarchically. I began tinkering with this idea with multiple cameras and CNNs, but have no qualifying experimental data to show. In some scenarios where there is not a convenient function to repurpose for, idle neural networks may rewire as a backup neural network in order to construct simple redundancy within a subsystem.

Certain design considerations are needed at the initial definition of the software architecture in order to implement this *Self-Repair Framework*. Hardware selections in mechanical and electrical design even warrant a second consideration as well. As previously stated, smoothly interchanging neural networks through transfer learning requires identical data distributions. This requires the system architecture to be specified in such a manner. In some instances, this may not be feasible, and retrofitting appropriate hardware and software may prove difficult. These are limits that are recognized by this model and give opportunity to optimize this concept further.

X. Conclusion

Artificial Intelligence that can demonstrate a higher level of resiliency when damaged has the potential to greatly impact robotics used in adverse environments. The concept I define here illustrates a very feasible framework for robotic self-repair with the help of supervised learning. By building upon the advantages this concept has proven while reconciling delinquencies found in experimental validation, there exists an achievable route to increased robustness in robotics and artificial intelligence. I explore reasons why self-repair behavior is needed in robotics and the potential that successful implementation of such behavior can manifest in real problems. I define an experiment to test this theory in line with the scientific method and attempt to quantify ways to reconcile training losses mathematically. Limitations of the framework are defined and opportunities to optimize the experimental results are discussed. Above all, the *Self-Repair Framework* illustrates how hardware damage to an autonomous system does not necessarily compromise the software or firmware associated with that damaged subsystem. After all, in a high-stakes situation, *some functionality is better than none*.

References

1. Than, K. (2021, May 3). *Why deaf have enhanced vision*. Science. Retrieved 18 October 2021, from https://www.nationalgeographic.com/science/article/101011-deaf-enhanced-vision-brain-health-science.

2. Redmon, Joseph; Farhadi, Ali (2018). YOLOv3: An Incremental Improvement. *ArXiv.* arXiv:804.02767.

3. Lin, Tsung-Yi; et al. (2015). Microsoft COCO: Common Objects in Context. *ArXiv.* arXiv:1405:0312v3.

4. Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., & Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. https://doi.org/10.1109/icassp.2017.7952261.

5. Wiesel, Torsten N.; Hubel, David H. (1964). Extent of Recovery from the Effects of Visual Deprivation in Kittens. *Journal of Neurophysiology.* https://10.220.32.246.

6. Liang, Yuchen; Ryali, Chaitanya; Hoover, Benjamin (2021). Can a Fruit Fly Learn Word Embeddings? *International Conference on Learning Representations 2021.* arXiv:2101.06887v1.

7. Li, Yang; Ji, Shihao (2019). Neural Plasticity Networks. *ArXiv.* arXiv:1908.08118v2.

8. Shuvaev, Sergey; Giaffar, Hamza; Koulakov, Alexei (2017). Representations of Sound in Deep Learning of Audio Features from Music. *ArXiv.* arXiv:1712.02898.

9. Franzoni, V., Biondi, G., & Milani, A. (2020). Emotional sounds of crowds: Spectrograph-based analysis using Deep Learning. *Multimedia Tools and Applications*, *79*(47-48), 36063–36075. https://doi.org/10.1007/s11042-020-09428-x.

10. Cormen, T. H., & Leiserson, C. E. (2009). *Introduction to Algorithms, 3rd edition.*

11. Alpaydin, Etham. *Introduction to Machine Learning*. Fourth Edition. 2020, Massachusetts Institute of Technology.

12. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron. *Deep Learning.* 2016, Massachusetts Institute of Technology. p 147 -149, 525 – 527.

13. Apple, Inc. (2020). Visualizing Sound as an Audio Spectrogram [web log]. Retrieved August 7, 2021, from https://developer.apple.com/documentation/accelerate/visualizing_sound_as_an_audio_spectrogram.

14. Kleinberg, J., & Tardos, É. (2014). *Algorithm Design.* Pearson India Education Services Pvt Ltd.

15. France, Kordel & Newman, Zachary. (2020). Cluster Neural Networks for Edge Intelligence in Medical Imaging. *ResearchGate.* https://www.researchgate.net/publication/345761193_Cluster_Neural_Networks_for_Edge_Intelligence_in_ *Medical_Imaging*

16. Musk, J. A., Sahai, S. K., & Elluswamy, A. K. (2021, March 23). *Estimating Object Properties Using Visual Image Data.*

17. Goodfellow, Ian J; et al. (2014). General Adversarial Nets. *ArXiv.* arXiv:1406.2661v1.

18. Russel, Stuart J.; Norvig, Peter. *Artificial Intelligence: A Modern Approach.* Third Edition. 2015, Pearson India Education Services Pvt. Ltd. p 961-962.

19. Google. (n.d.). *AudioSet*. Google. Retrieved November 6, 2021, from https://research.google.com/audioset/index.html.

20. Furton, K. G., Caraballo, N. I., Cerreta, M. M., & Holness, H. K. (2015). Advances in the use of odour as forensic evidence through optimizing and standardizing instruments and canines. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *370*(1674), 20140262. https://doi.org/10.1098/rstb.2014.0262

21. Likhosherstov, Valerii; et al. (2021). Poly-ViT: Co-training Vision Transformers on Images, Videos and Audio. *ArXiv.* arXiv:2111.12993v1.

22. B, N. (2019, January 18). *Image data pre-processing for Neural Networks*. Medium. Retrieved December 12, 2021, from https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258.

23. Sony. (2019). Sony [Product Information] IMX677-AAPH5, Rev 1.1. Sony Semiconductor Solutions Corporation. URL: *https://www.sony-semicon.co.jp/products/common/pdf/IMX677-AAPH5_AAPJ_Flyer.pdf*

24. Ainstein (2021). US-D1 UAV Radar. Ainstein AI. URL: *https://ainstein.ai/wp-content/uploads/2021/11/US-D1-Data-Sheetv3.pdf*

25. PalmSens BV. (2019). EmStat pico Data Sheet, Rev 8-2019-004. PalmSens BV. URL: *https://cdn.palmsens.com/wp-content/uploads/2021/04/PSDAT-ESP-EmStat-Pico-Datasheet.pdf*

SUPPLEMENTARY MATERIAL

To show duplication of results with the *Self-Repair Framework*, I evaluate two other combinations of sensors. In each evaluation, I follow the guidelines specified for the framework and show how one evaluation fails while the other succeeds.

A. Vision-Radar Repair:

This combination is commonly used within navigation of autonomous vehicles. My intent was to perform something similar to what was proposed in the first experiment using a Sony camera array [23] and Ainstein radar sensor [24]. I leveraged the same YOLO object detector and convolutional neural network from the original experiment. A separate CNN was built to detect patterns received from the radar. Data between both sensors fused successfully, but the radar did not provide enough resolution to allow for accurate object detection on its own initially. Allowing the radar to scan back and forth in a pendulum motion helped gather the needed resolution. The neural net over the radar was "damaged", retrained, and provided results that were very consistent with that of the camera. This effort was a successful duplication of results.

Figure 1S - Images of the camera (left) and radar (right) used for this concept of the framework.

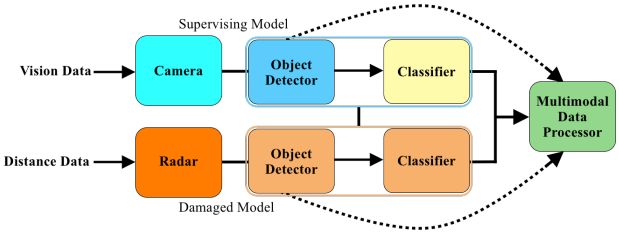

Figure 2S - The block & interconnect diagram for the vision-radar Self-Repair Framework.

B. Vision-Electrochemical Repair:

This combination is the most unconventional, but one that is necessary in order to prove the robustness and versatility of the *Self-Repair Framework*. A camera is coupled with an electrochemical sensor that is designed to detect the trace of specific organic compounds at a distance [20]. In this case, the sensor has been trained to detect 2-ethyl-1-hexanol/2,4-dinitrotoluene which is a primary component of the explosive known as C4. This sensor combination allows an autonomous system to navigate through vision and traces of electrochemical "smell". The same object detector and CNN from the previous two experiments were leveraged. Both models were successfully trained over the data, but the firmware on the electrochemical sensor is locked by the manufacturer and does not allow for detection frequencies that will synchronize reliably with the camera frame rate. The data was difficult to coordinate due to this. Also, aerodynamic effects of air flow caused ambiguity on when chemical traces were actually contacting the sensor. This attempt has not yet yielded reproducible results for the *Self-Repair Framework*.

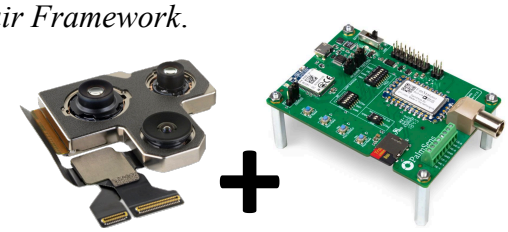

Figure 3S - Images of the camera (left) and electrochemical processor (right) used for this concept of the framework.

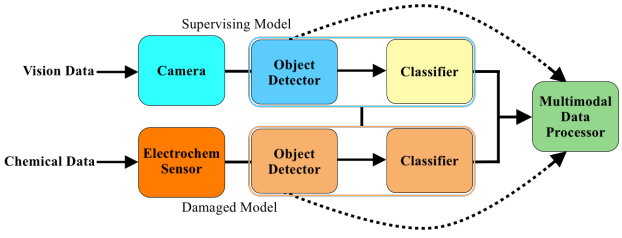

Figure 4S - The block & interconnect diagram for the vision-electrochemical Self-Repair Framework.