A Concept of Self-Repair for Robotics

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

Abstract*— *If a human is injured, the biology of its body immediately begins repairing itself. Blood starts to clot in flesh wounds, antibodies are produced as a response to a virus, and neural structure changes to accommodate brain damage through neuroplasticity. Artificial neuroplasticity is the particular focus of this paper. In order to grant robotics the same liberties that humans enjoy in regards to self-repair, it will be shown how this can effectively be accomplished through the cooperation and reorganization of two neural networks. If a sensor sending input to an artificial neural network becomes damaged, the neural network of the damaged sensor will repurpose itself to be used for another task through self-reorganization and on-the-fly transfer learning. In other words, a robot can exhibit self-repair by borrowing from principles of biological neuroplasticity.

***Index Terms*—artificial intelligence, robotics, neuroplasticity, machine learning, transfer learning, neural network, generative adversarial network, spectrograph, motor control.**

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], but it is the knowledge of the author that no published work exists on attributing explicit effort toward the repair of damage. In the following framework, it is shown that 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.

II. Background

Just as humans have their five senses, robotic sensory modes can be broken down into different types of stimulus. Damage can be assumed to be a handicap in functionality to one or more of these senses. A concept of self-repair can be shown as an attempt to restore functionality to the damaged senses or to repurpose still-functioning resources previously used by the damaged sense. To demonstrate self-repair through neuroplastic behavior, two senses are leveraged for the experiment—vision and audio. This allows for

experimental control while also maintaining a highly tractable scenario in robotics. One can imagine how this concept can be extrapolated to somatosensory applications to emulate touch or even electrochemical applications to emulate smell. An experiment and architecture of an autonomous system is proposed to demonstrate a proof-of-concept approach. This autonomous system has the ability to perceive and respond to visual and auditory stimulus via microphones and cameras and coordinates self-navigation through a simple control loop that responds to feedback from both sound and sight. 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 may be shown in *Figure 1*.

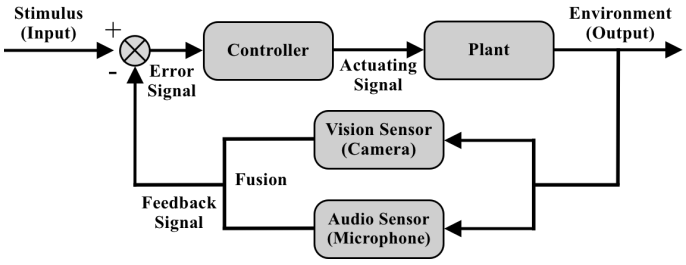

Figure 1 - The simple feedback loop that coordinates motor control in response to environmental input.

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]. The author attempts to synonymize this principle with two neural networks, a microphone array, and a camera.

This concept of plasticity is demonstrated in the autonomous navigation domain. Spatiotemporal awareness is crucial to perform any sort of localization, navigation, or mapping task whether robotic or human. In general, rapidly acquiring such 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. 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.

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.

III. DATA

Data for this experiment was leveraged from open resources. The computer vision algorithm used

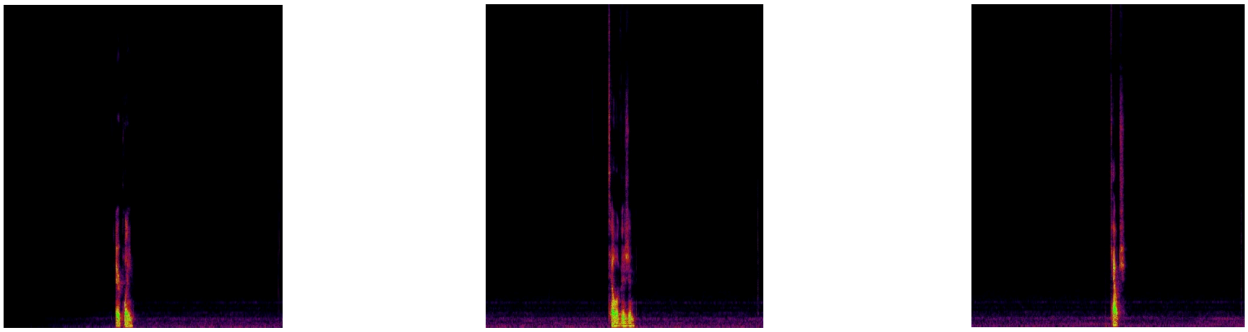

Figure 2 - From left to right, spectrographs for spoken words "airplane", "television", and "couch" are shown as images.

for object detection was the YOLOv3 detector [2] trained over the COCO dataset [3]. Audio data used to train the audio neural network was acquired through Google's AudioSet dataset [4].

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. 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.

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 audio classification at one time step correspond to the results of vision classification at the same time step. One can see 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. 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. Data augmentation techniques were used to vary the color (decibel level) of the spectrograph, but typical data augmentation techniques were not employed since flipping, mirroring, or rotating the image produces a fundamentally different sound in the context of audio.

IV. Experimental Design

To accommodate the hardware necessary for a camera, microphone array, and processor that can handle the computation needed to perform this experiment, 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 neuroplasticity 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 shall be run on a smart tablet

since it contains a built-in microphone and camera. Specifically, the tablet shall be 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. The entire software build including all algorithms and the control loop shall be denoted as the "Agent" and the tablet running the app as the "device" throughout the rest of this paper.[1] The framework 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.

An object recognition algorithm is used to locate objects within the camera field of view. This object detector follows the YOLOv3 design with the Darknet architecture. 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 us to transfer learn a simpler neural network instead of an entire 53-layer object detector [15]. It also allows us 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. Similarly, once a specific sound is detected within an audio pattern, the waveform attributed to that sound is cropped within the spectrograph. The architecture for each of these neural networks is identical and may be viewed in *Figure 4*. 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*.

	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 3 - Darknet-53 architecture.

Before the experiment begins, the weights files for both convolutional neural networks are noted to ensure that transfer learning occurs successfully. Once hardware damage occurs, the affected neural network will need a way to acquire ground truth labels for its repurposed task. The other neural network will provide such ground truth labels and coordinate supervised learning over the damaged network. The damaged neural network

[1] Since the inspiration of this experiment is modeled after biological neuroplasticity, the author 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.

will then transfer learn from the unaffected one until testing accuracy in classification passes a threshold of 80%.

At this point, it is important to establish a few definitions in order to maintain comprehension throughout the experiment. Throughout this paper, the *audio model, retrained model, damaged model* or *damaged neural network* shall be defined as the neural network that was or will be affected by the damaged microphone array and was originally processing audio. The *vision model, ground-truth model, training model* or *training neural network* shall define the model strictly associated with visual processing and used to train the damaged model. The *Agent* shall define the entire program containing both models working together. It is worth emphasizing that the damaged and undamaged models are shown in the experiment as audio and visual processing neural networks only as an example, and that the concept shown by this experiment may be extrapolated to any AI model over any data medium.

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.

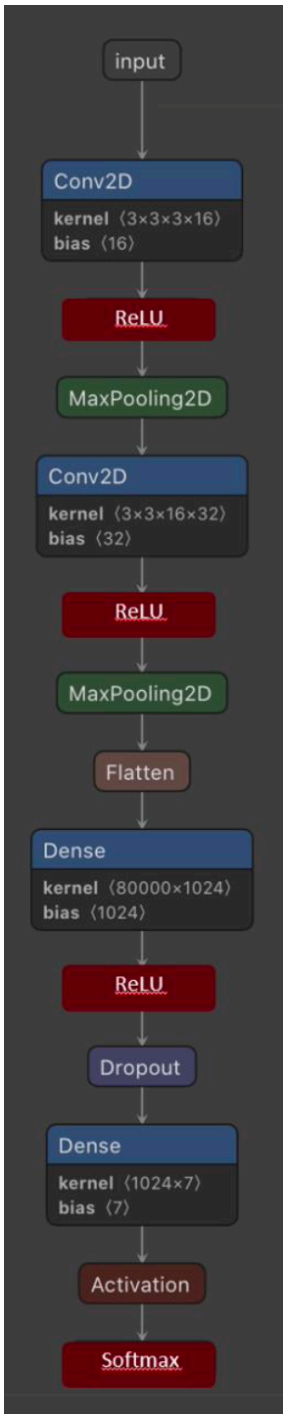

Figure 4 - Architecture for both the audio and visual neural networks.

It is worth noting that the speed in which this plasticity occurs is significantly accelerated in our experiment in comparison to real-world scenarios in humans. This is intentional. If a human attains neurological damage to the brain resulting in compromised auditory processing, the plasticity

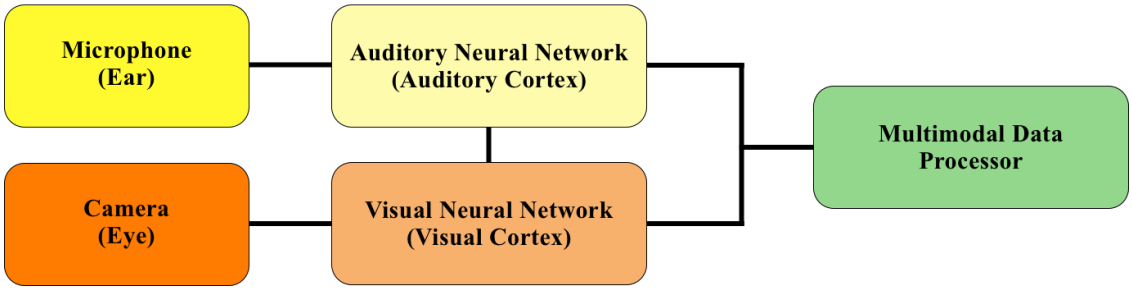

Figure 5 - A block diagram illustrating the interconnection between the audio and visual models and their associated hardware.

in which these parts of the brain are repurposed occurs over the course of months and years. The author elected to accelerate the timeline of this process in our experiment for convenience, comprehension, and to show how artificial intelligence can exhibit neuroplastic behavior with an expedited timeline.

V. Training

The parameters used for transfer learning were identical to the ones used in training the undamaged neural network. Similarly, each of the two neural networks were identical in architecture. Ten output classes were allowed for each neural network. The ten class labels were as follows: *airplane, boat, bus, car, cat, cow, dog, motorbike, person, train.* 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 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

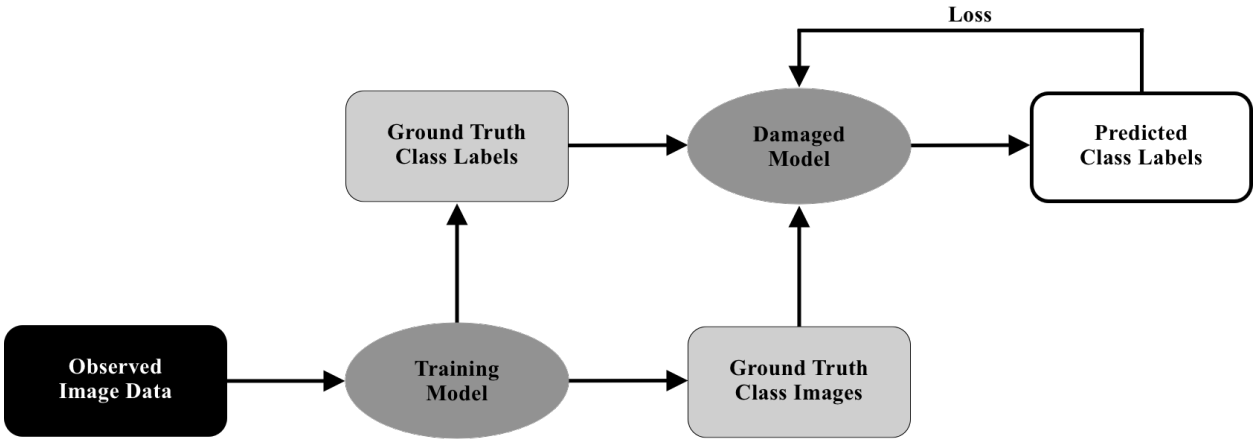

Figure 6 - One full update cycle of the repair concept used to coordinate transfer learning for a damaged model. Notice the similarities between the discriminator and generator models within the GAN, but also the key differences in the weight update cycle. The Damaged Model is analogous to the Discriminator Model in the GAN, while the Training Model is analogous to the Generator Model.

completed in retraining, transfer learning ceases and the model is tested on a different dataset. Successful convergence on this dataset over the same loss function results in the retrained model being deployed colinearly with the training model and both are 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, we 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, we 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. We 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 can be considered similar to the loss between the generator and discriminator

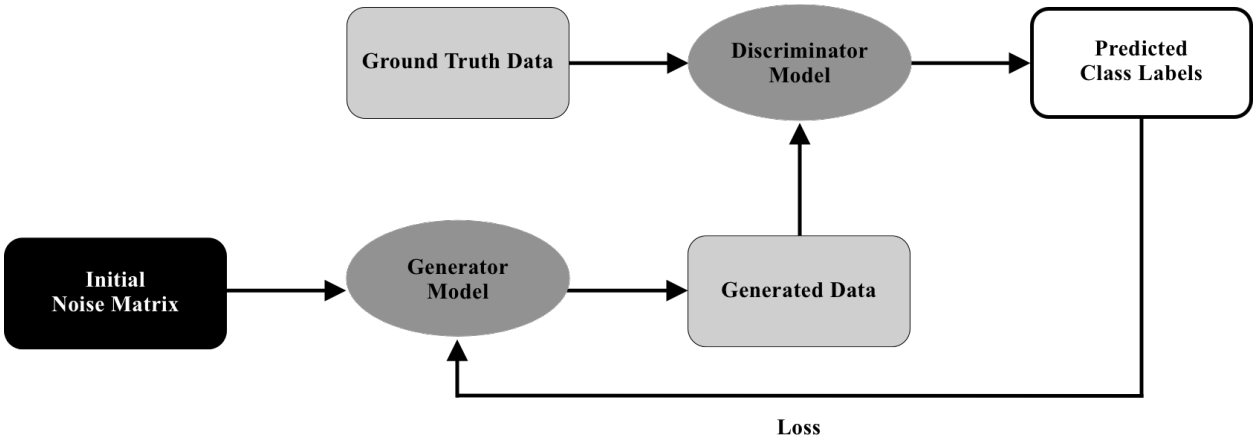

Figure 7 - One full weight update cycle of a Generative Adversarial Network used to generate authentic fake data.

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. Our principle 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 this neuroplastic method, 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 training and retrained models may be considered as globally optimal discriminators [17].

The allusion to GANs is worth focusing on for a moment. The weight update cycle for the repair architecture defined in this paper is starkly similar that of a GAN. *Figure 7* shows the weight update cycle of a GAN. *Figure 6* displays the weight update cycle for the repair network. Notice the similarity in position and functionality of both models within each of their respective cycle. The training and discriminator models both produce observational data for its counterpart model to learn, but the training model in the repair network is simply feeding the damaged model exact copies of its *own observations* as it perceives the world, while the generator in the GAN is *creating* fake data to feed to the discriminator. Note that the computed loss updates the generator in a GAN in contrast to the repair network where the computed loss over the predicted class labels updates the weights in the damaged model through Backpropagation [18].

VI. Experiment

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. Statements illustrating weight adjustment and additional visual recognition are programmed to log as the experiment occurs. 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. During the last five minutes of the experiment, the repurposed neural network shall undergo a final "validation" period in which its accuracy in generalizing over a new domain shall be evaluated.

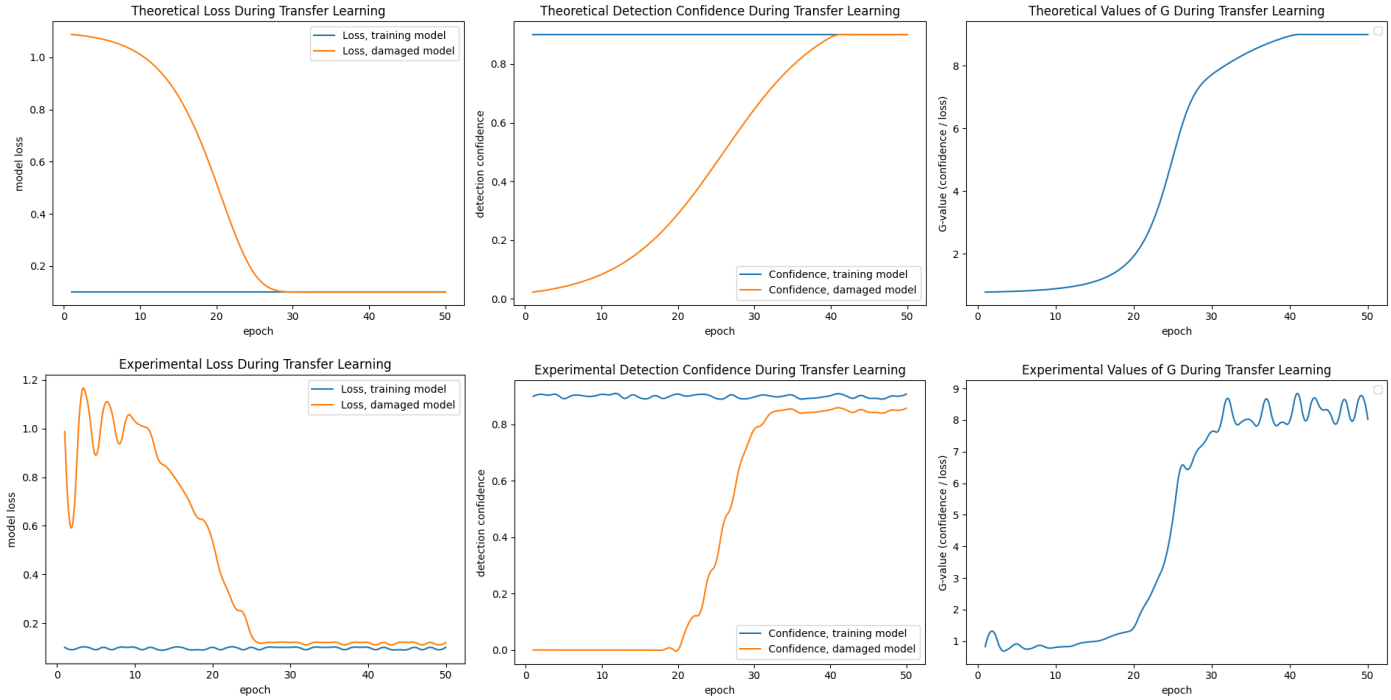

*Figure 8 - **Top:** metrics for what a globally optimal model should return if all knowledge from the training model is learned by the damaged model. **Bottom:** metrics for the actual model returned by preliminary experimentation.*

The following schedule shall proceed as follows 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:

1. $\boldsymbol{T + 0}$**:** Startup, Agent initialization
2. $\boldsymbol{T + 1}$**:** Vision and audio models confirm nominal performance
3. $\boldsymbol{T + 5}$**:** Damage to microphones occur, Damage Assessment Period begins
4. $\boldsymbol{T + 5.5}$**:** Damage Assessment Period terminates, sensor failure confirmed, control loop self-adjusts, vision model notified and begins generating training data
5. $\boldsymbol{T + 6}$**:** Damaged audio model begins transfer learning from vision model
6. $\boldsymbol{T + \tau}$**:** Damaged model ceases transfer learning, vision model ceases training data generation, training data relieved from Agent RAM
7. $\boldsymbol{T + \tau + 0.1}$**:** 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.

VII. 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 replicate exactly. 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 8* 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 8.* 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.

VIII. Advantages and Disadvantages

The above framework admits several methods in which neuroplastic behavior may be implemented 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. In this method, transfer learning occurs in batches, but batches are constructed one image at a time and at the rate of which the Agent's environment makes such data available. One technique to accelerate this timeline includes

programming the Agent with the ability to expand the training dataset through standard data augmentation techniques such as by flipping the observed image and rotating it several degrees, making several copies of one image [16]. The training network currently coordinates a supervised learning effort for the damaged neural network, but a subroutine that takes ground truth data perceived by the teacher network and augments the data 10-20x could expedite transfer learning significantly, especially in a real scenario. For example, consider a self-driving vehicle that uses both radar and cameras to navigate. With this model, if the vehicle obtains damage to its radar sensors, the computational stack assigned to processing Lidar data may be repurposed to provide additional compute power to the undamaged cameras in vision-only navigation.

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 neuroplastic behavior work? A subroutine built into the agent beforehand could triage damage in such a manner as to allocate resources hierarchically. 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 concept of self-repair. 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.

IX. Conclusion

This framework for robotic self-repair is defined by leveraging principles of biological neuorplasticity. Reasons why neuroplastic behavior is needed in robotics is explored and the potential that successful implementation of such behavior can manifest is shown. An experiment is established to test this theory in line with the scientific method and a review of the software design is performed. Limitations of the Agent model are defined and results achieved by the Agent throughout the experiment are discussed. Finally, methods in which the results of the experiment can be improved are expounded upon in order to provide better traction in a real-world scenario.

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 proposed in this paper leverages principles of neuroplasticity in order to illustrate how hardware damage to a robot or autonomous system does not necessarily compromise the software or firmware associated with that damaged subsystem. In future work, the author hopes to extrapolate this concept further to facilitate faster transfer learning and show how this neuroplastic behavior can work in a more complex scenario with critical damage of more than one sensor. The concept defined here illustrates a very feasible framework for robotic self-repair, and by building upon the advantages this concept has proven while reconciling delinquencies encountered in experimental validation, there exists an achievable route to increased robustness in robotics and artificial intelligence.

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. Garey, M. R., & Johnson, D. S. (2003). *Computers and Intractability: A Guide to the Theory of NP - Completeness.* W.H. Freeman and Co.

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.