FLARE (Fault Logging and Assessment for Responsive EV repairs)

Ethan Deng ezdeng@ucsd.edu
Jason Gu jig036@ucsd.edu
Daniel Kong dskong@ucsd.edu
Irving Zhao ziz057@ucsd.edu

Phi Nguyen pnguyen@sdge.com
Ari Gaffen AGaffen@sdge.com
James McClosekey JMcClosekey@sdge.com

Anderson Bolles ABolles@sdge.com
Kelly Park KPark@sdge.com

Abstract

As the world slowly adopts electric vehicles (EVs) to promote cleaner transportation, there is one key issue that hinders the widespread adoption of EV, and that is the public EV charging infrastructure. Public EV chargers are simply unreliable and remains a significant barrier for individuals considering an EV as their next vehicle. Unlike gas stations, where operational reliability is nearly guaranteed, EV chargers frequently suffer from damaged cords, broken screens, or software malfunctions. Current reporting systems rely on phone calls or written feedback on an app like Plugshare, which lacks standardization and clarity on the specific damages.

The goal of this project is to automate EV charger fault detection by integrating computer vision with structured user input. We use semantic segmentation and binary classification to classify EV charger faults from images, supplemented by user-reported classification from a menu.

The proposed solution is implemented as a mobile app prototype where users can submit photos and select issue categories via a structured reporting interface. Additionally, the app could integrate with PlugShare's database to improve the visibility of broken chargers. Our approach improves the efficiency of fault logging, reduces reporting ambiguity, and provides utility companies with structured insights for more responsive EV charger repairs.

Code: https://github.com/dskong07/FLARE_ev_infra/tree/main

Website: https://github.com/JingChengGu/FLARE-website

1 Introduction

The shift to electric vehicles (EVs) is essential for reducing carbon emissions and achieving sustainable urban mobility, as EVs have up to 67% fewer emissions over their lifetime compared to ICE vehicles (Wood Mackenzi). However, one of the major challenges preventing widespread adoption is the inconsistent reliability of public EV chargers. Unlike gas stations, where infrastructure is well-established, EV chargers often suffer from hardware failures, software glitches, and physical damages, leading to poor user experiences. When a charger malfunctions, users typically report issues through phone calls, emails, or app-based feedback, but these manual submissions are often vague, unstructured, and lack key details. Utility companies must then dispatch technicians for on-site verification, introducing delays in repairs and inefficiencies in resource allocation.

To address these challenges, the team developed FLARE (Fault Logging and Assessment for Responsive EV repairs), an automated fault detection system that combines fault detection and structured user reports to streamline the identification and resolution of charger issues. We use Nvidia's state-of-the-art semantic segmentation model (MIT-B3) to segment components of EV chargers from user-submitted images. These images are then passed through a Vision Transformer model (ViT) that classifies charger components as either 'healthy' or 'broken.' These models are supplemented with user input from predefined issue categories such as "damaged cord," "broken screen," "damaged plug," or "out of order." This structured input eliminates ambiguity and ensures standardized data collection. Ideally, the system could interface with PlugShare's API, allowing real-time updates on charger status and improving visibility into infrastructure reliability.

Previous research primarily focused on charger availability tracking rather than fault detection, relying on subjective, crowdsourced reports without automated validation. While some platforms allow users to report issues, they lack integration with classification models and fail to provide structured data for maintenance teams. This project advances prior work by implementing semantic segmentation for real-time fault detection and proactive maintenance insights. By automating fault reporting and improving data quality, FLARE enhances the efficiency of EV charger maintenance, reducing downtime and increasing user confidence in charging infrastructure. This paper details the methodology, implementation, and evaluation of FLARE, providing a proof-of-concept for an improved EV charger fault logging system.

2 Methods

2.1 Data Collection

To train the fault classification model, images of EV chargers were collected from both manual and online sources. The dataset includes images taken under varying conditions to improve model generalization.

Manual data collection involved capturing images of charging stations in different operational states, including healthy, broken screen, damaged cord, damaged plug, and out of order. For more data variety, pictures of chargers were taken at multiple angles, but the majority are from a front-facing perspective. Photos were captured during both daytime and nighttime to account for lighting variations that may affect classification performance. Charger locations were identified using PlugShare, with priority given to stations that were either reported as faulty or undergoing maintenance.

To expand the dataset, additional images were sourced online. These images were gathered from Google Images, Reddit, PlugShare, and blog posts discussing charger malfunctions. Many of these sources contain real-world user-reported failures, providing valuable diversity in the dataset. Images collected from online sources were manually reviewed to ensure relevance and consistency with the manually collected dataset. Here are some example images we collected 1

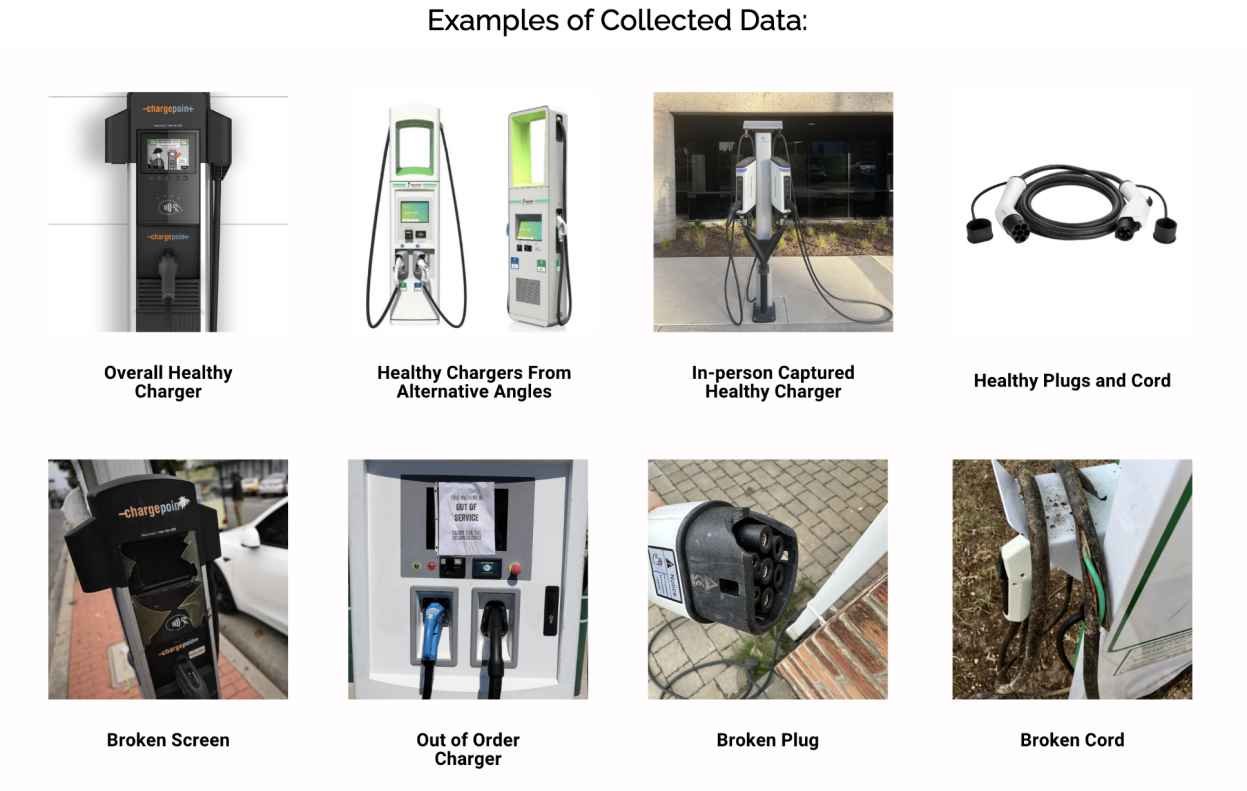


Figure 1: Example images of the chargers collected during data collection

All images were labeled into five predefined categories to facilitate supervised learning. The labels assigned were: healthy, broken screen, damaged cord, damaged plug, and out of order. The labeling process was performed manually to ensure high-quality annotations. To maintain consistency, the images were in jpg format, resized, and pre-processed before being used for training. In total, we ultimately collected 306 healthy charger images and 101 broken charger images.

2.2 Data Labeling

Once images of EV chargers were gathered, these images were manually labeled to prepare for training. We used Segments.ai, a machine learning-assisted labeling tool to annotate our collected images. By segmenting the components of EV chargers, such as screens, cords, plugs, and bodies, we trained our classification model to recognize the features of an EV charger. All of these images needed to have a .jpg extension to ensure streamlined model training down the line. A total of 265 images were labeled, such as full-body images of chargers, up-close pictures of screens, day- and night-time photos, or just plugs and cables. Below are examples of data on the labeling results 2 and what the labeling process looked like on Segments.ai 3.

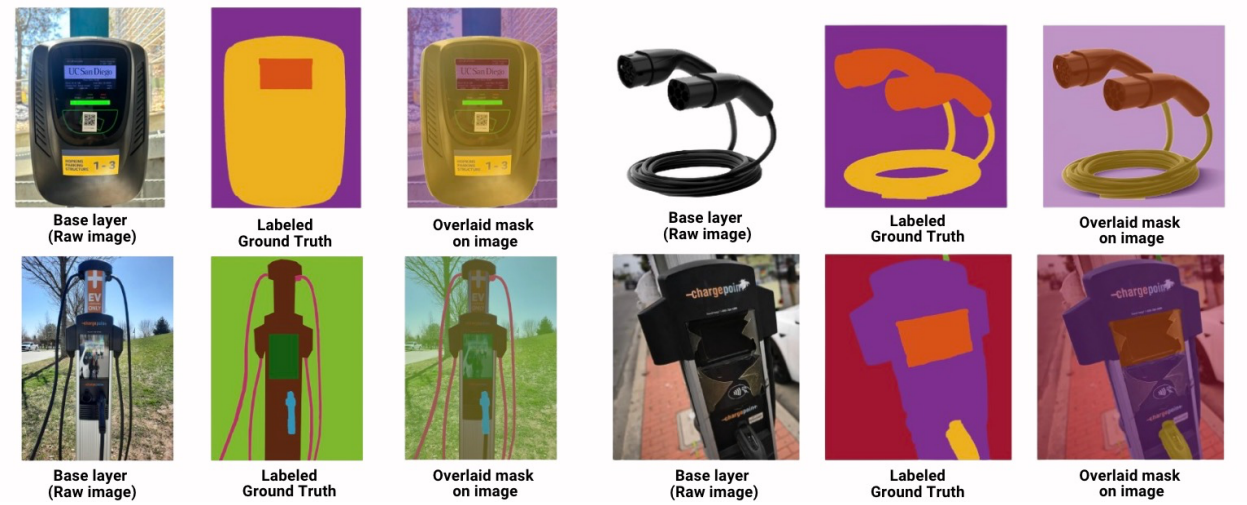


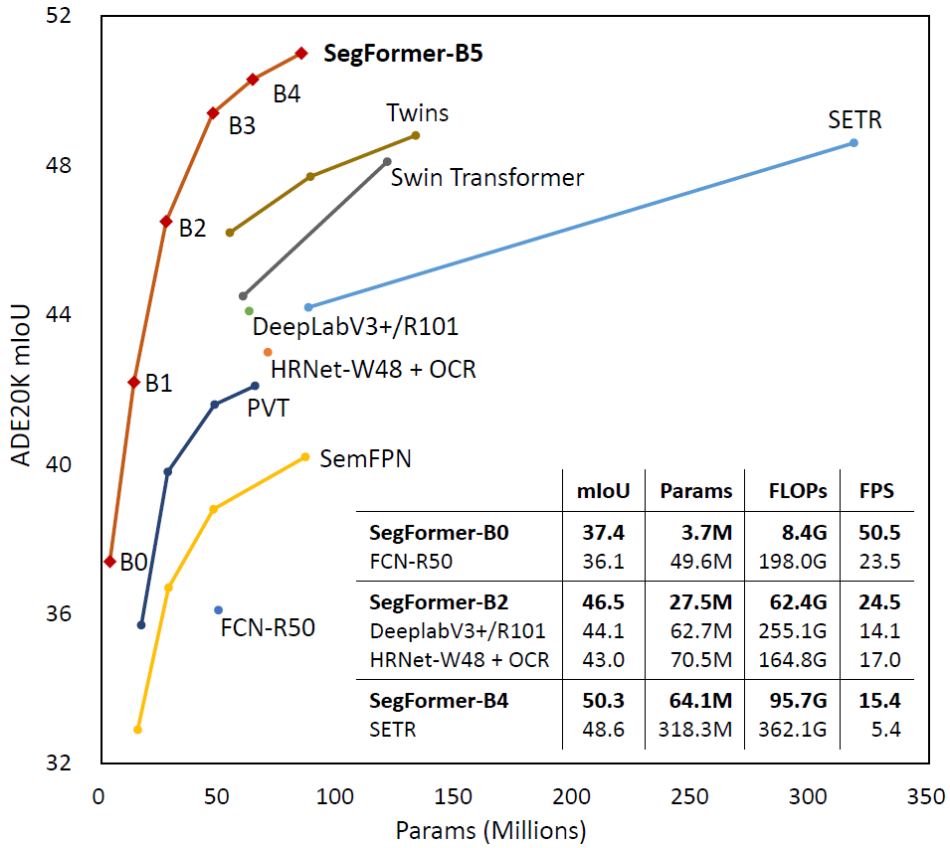
Figure 2: Example of Segments.ai segmentation with 4 unique images, including the base layer and overlaid layer for all images

Figure 3: Labeling interface on Segments.ai

Once we labeled the images, we reviewed them to make sure that they were labeled adequately, as some images were very difficult to label with great accuracy. After reviewing all of our labeled images, we ended up with a total of **167 images** to work with that were of the correct file extension (.jpg) and labeled properly. Going into training, we know that we were limited from our small dataset size, the quality of our data, and limited computing power. These labeled images were exported from Segments.ai to HuggingFace-an open-source platform for building, deploying, and testing machine learning (ML) models-for model training, which can be seen (here).

2.3 Segmentation Model Training

Before labeling on the EV charger images started, we first experimented with a sidewalk dataset from HuggingFace to experiment if this segmentation method was a feasible solution to our problem. After experimentation, the NVIDIA MIT model (MIT-B0) proved to be robust, despite being the lightest model out of the MIT family—sitting at only 3.7 million parameters—which can be seen in this (documentation). Training MIT-B0 on the sidewalk dataset achieved 74.8% accuracy and a mean IoU of 0.159 across classes with significantly limited training. Here is also a graph of how well the MIT-B model performs compared to other segmentation models on the ADE20K dataset, where the y-axis the segmentation mean IoU, and the x-axis is the number of parameters each model has 4.



	mIoU	Params	FLOPs	FPS
SegFormer-B0	**37.4**	**3.7M**	**8.4G**	**50.5**
FCN-R50	36.1	49.6M	198.0G	23.5
SegFormer-B2	**46.5**	**27.5M**	**62.4G**	**24.5**
DeeplabV3+/R101	44.1	62.7M	255.1G	14.1
HRNet-W48 + OCR	43.0	70.5M	164.8G	17.0
SegFormer-B4	**50.3**	**64.1M**	**95.7G**	**15.4**
SETR	48.6	318.3M	362.1G	5.4

Figure 4: Graph showing the performance of MIT-B compared to other segmentation models

Using the same framework for testing segmentation with MIT-B0 on a sidewalk dataset, we used an upgraded model (MIT B3) for our labeled dataset of EV chargers for better performance. The dataset from the HuggingFace repository was saved under dskong07/chargers-full-v0.1, and contains pixel-wise annotations distinguishing different components such as the screen, body, cable, and plug. The MIT B3 model (segformer-b3-finetuned-segments-chargers-full-v3.1) is a pre-trained SegFormer model fine-tuned for EV charger segmentation, imported from the transformers library. This model provides state-of-the-art performance in semantic segmentation by leveraging transformer-based feature extraction. Here is a visualization of the model's architecture 5.

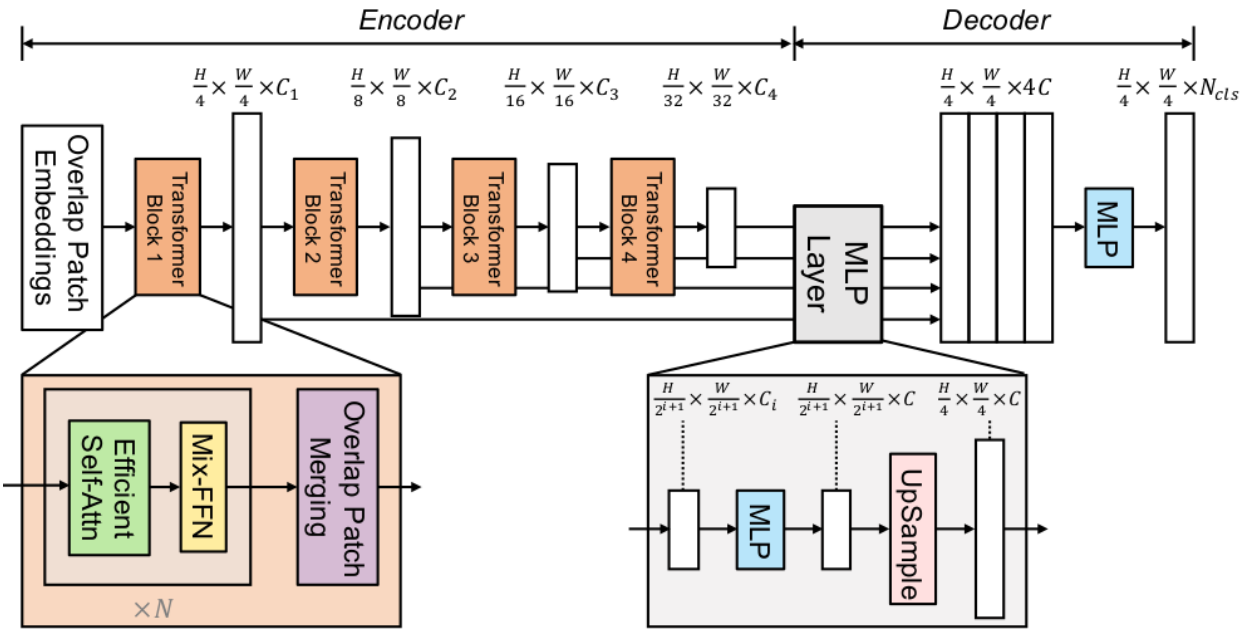


Figure 5: Visual representation of the SegFormer Architecture

For dataset training, the collected images were first randomly shuffled and split into 80% training and 20% testing subsets using train_test_split(). The model leverages a learning rate of 0.00006, a batch size of 2, and 50 training epochs. Data augmentation was applied using ColorJitter, which adjusted brightness, contrast, saturation, and hue to enhance model generalization. The images and their corresponding labels were preprocessed using Hugging Face's SegformerImageProcessor, which resized and normalized the inputs before training. Training was conducted using the Hugging Face Trainer API and the model was initiated with SegformerForSemanticSegmentation.from_pretrained(pretrained_model="nvidia/mit-b0"), with performance monitored via Intersection over Union (IoU) and accuracy metrics, computed using the evaluate library. The model's predictions were interpolated using bilinear scaling, and performance was assessed per category to ensure accurate segmentation across all charger components. The best-performing model (MIT-B3) was automatically saved and uploaded to the HuggingFace Model Hub for future deployment.

We made a color palette to visually differentiate each segmented class in the output images (red for screen, green for plug, yellow for body, blue for cord, and purple for background). The segmented outputs were then visualized using matplotlib.pyplot and PIL.ImageDraw to overlay segmentation masks on the original images. These visualizations were used to evaluate the accuracy of the model in segmenting various charger components with intersection over union (IoU) and accuracy measurements. IoU measures the interaction between predicted output and actual value, while accuracy measures how much of the prediction was correctly classifying each component. Here are examples of how well each MIT model classified a particular charger image in the results section 8.

2.4 Classification Model Training

After successfully segmenting the components of EV chargers, we trained Vision Transformers (ViT) from HuggingFace to classify images of healthy and broken EV charger condition. For each condition (screen, plug, cable, and out of order) we trained a ViT model with vit-base-patch16-224-in21k. Here is a graph visualization of how Vision Transformers work using a transformation encoder on the ImageNet dataset, performing even better than traditional convolution architectures 6. More information can be found (here). The dataset consisted of images categorized into two classes: healthy with a label=0, and broken with a label=1. Since some images were not in a uniform format, they were first converted to RGB mode and saved as PNG files. Images were also renamed systematically for consistency. A metadata.csv file was created to store file names and their corresponding labels.

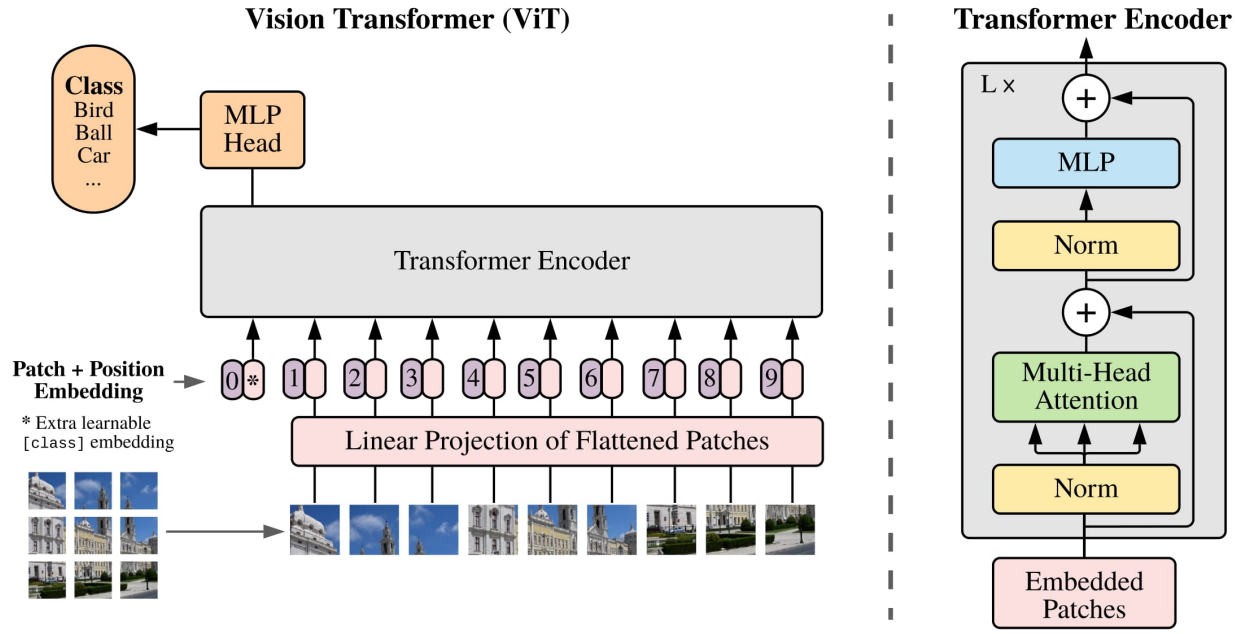


Figure 6: Here is the Vision Transformer Architecture

The dataset was loaded into a HuggingFace dataset and then divided into 80-20 train and test splits. After being loaded in, the data was transformed using PyTorch's torchvision: with RandomResizedCrop, Normalization, and ToTensor, images were cropped to a standard size, normalized based on mean and standard deviation, and then converted into tensors for model training.

Google's Vision Transformer model used for image classification was initialized using AutoModelForImageClassification.from_pretrained(checkpoint, num_labels=2). Then, a data collator (DefaultDataCollator) was used to batch and pad inputs properly. The Trainer API was used to train the model with the following hyperparameters: Batch size: 2 per device, Learning rate: 5e-5, Epoch: 10, Gradient accumulation: 2 steps, Evaluation strategy: step-based evaluation, Best model selection: based on accuracy. For model training, Trainer.train() was used and handled the gradient updates, loss calculations, and back-

propagation. After training, the model was uploaded to the HuggingFace Model Hub using trianer.push_to_hub.

The trained model is then tested using the HuggingFace pipeline on a sample image from the dataset, but not in the training set. The classification output provides confidence scores for each class (healthy or broken) Here are some examples of classifications on different EV charger conditions in the results section 10.

2.5 App Design

The FLARE platform is designed as a mobile application that enables users to report charger faults by submitting images and selecting predefined issue categories. The app streamlines fault reporting by integrating image classification with structured user input.

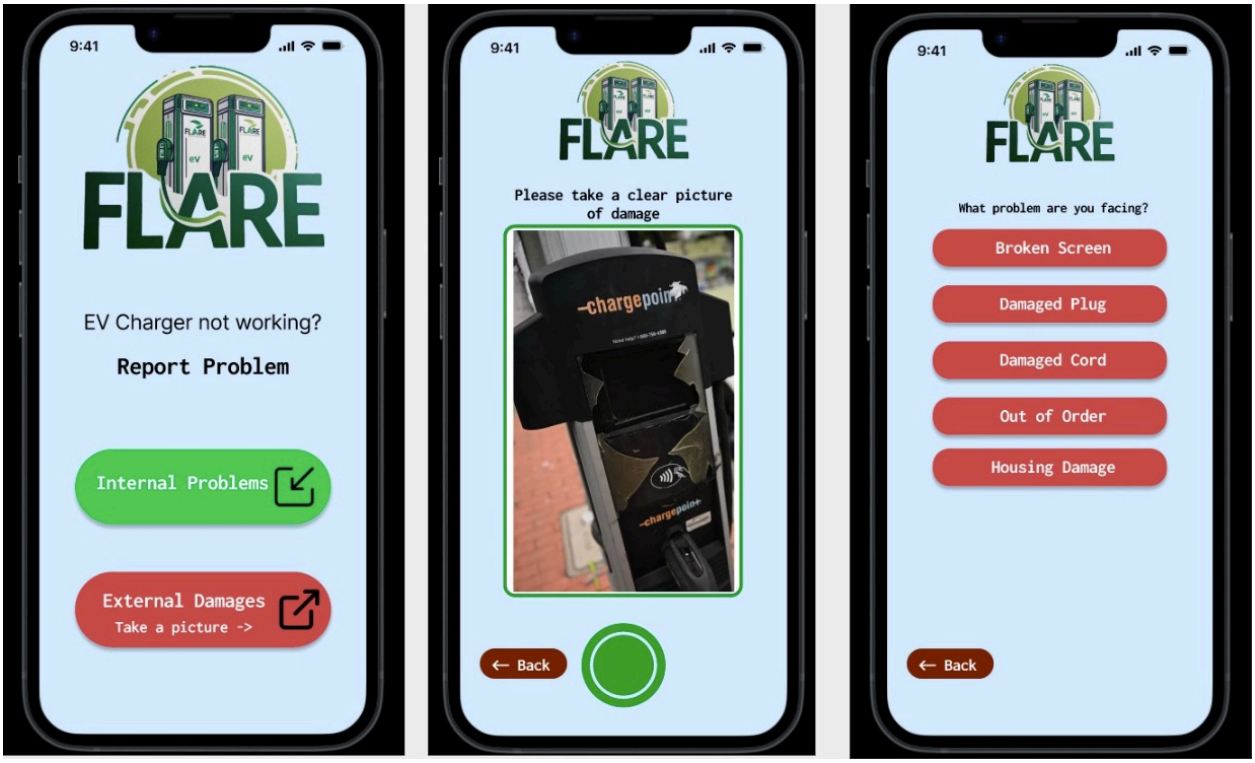


Figure 7: Sample images of FLARE app prototype

The user interface consists of several key components. The home screen displays two reporting options for the user to select, ”Internal Problems” or ”External Damages”. The internal problems report screen enables users to select a text description of the internal charger issue (network connectivity, slow charging speeds, payment issues, unusual noise, burning smell) for the relevant party responsible for repair. The external report submission screen enables users to upload an image and select an issue category (broken screen, damaged plug, damaged cord, out of order, housing damage). A categorized fault menu standardizes user input, ensuring uniform data collection. The charger status dashboard presents

reported issues in an accessible format and ideally would integrate with PlugShare's data about the respective charger.

When a user submits a report, the system processes the uploaded image through our fault detection model. The detected fault type is then recorded in the database and displayed on the dashboard for utility companies to review. This structured approach reduces ambiguity in fault reports and improves response efficiency. User inputs are necessary for cross validating the model's prediction with the actual problem users experience.

To enhance data reliability, the platform is designed to interface with PlugShare's API. This integration allows real-time updates on charger conditions, synchronizing user-reported faults with crowdsourced feedback. By incorporating PlugShare's data, the platform ensures that charger status reflects actual conditions, benefiting both users and maintenance teams.

3 Results

3.1 Segmentation Results

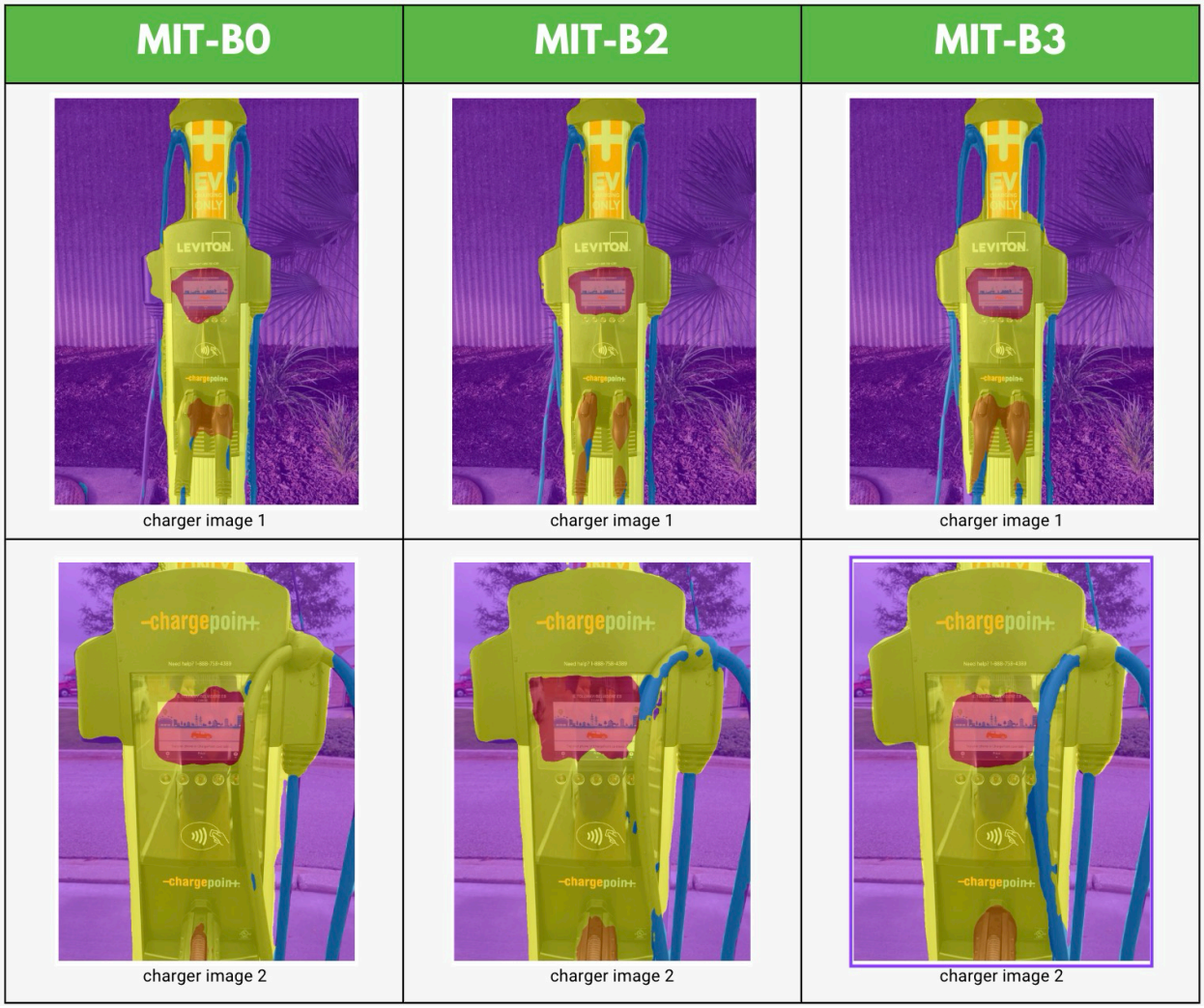


Figure 8: Performance of 3 different levels of NVIDIA's MIT-B SegFormer Segmentation model

Looking at the figure above 8, we can see that the segmentation performance increases as we use models with more parameters, (MIT-B0 has 3.7 million, MIT-B2 has 25.4 million, and MIT-B3 has 45.2 million). The MIT-B3 model achieved 87.1% mean accuracy across classes, 93.5% overall accuracy, and a mean IoU of 0.804 across classes after fine-tuning. Here is also a table showing the accuracy of our MIT-B3 segmentation for each component of an EV charger 9. Accuracy in this context refers to the overall percentage of pixels that are correctly classified across all image segments.

Accuracy Screen	Accuracy Body	Accuracy Cable	Accuracy Plug
0.769	0.944	0.771	0.900

Figure 9: Table showing the segmentation accuracy for each EV charger component

3.2 Classification Results

In the figure below 10, we can see an example of how well each problem was classified regarding the screen, cord, plug, out-of-service chargers, and healthy chargers. The Vision Transformer (ViT) model demonstrated strong performance in identifying damaged and functional components, particularly in high-contrast scenarios where defects were visually distinct. Screens with visible cracks, plugs with broken connectors, and frayed cables were classified with high confidence, contributing to an effective fault detection pipeline.

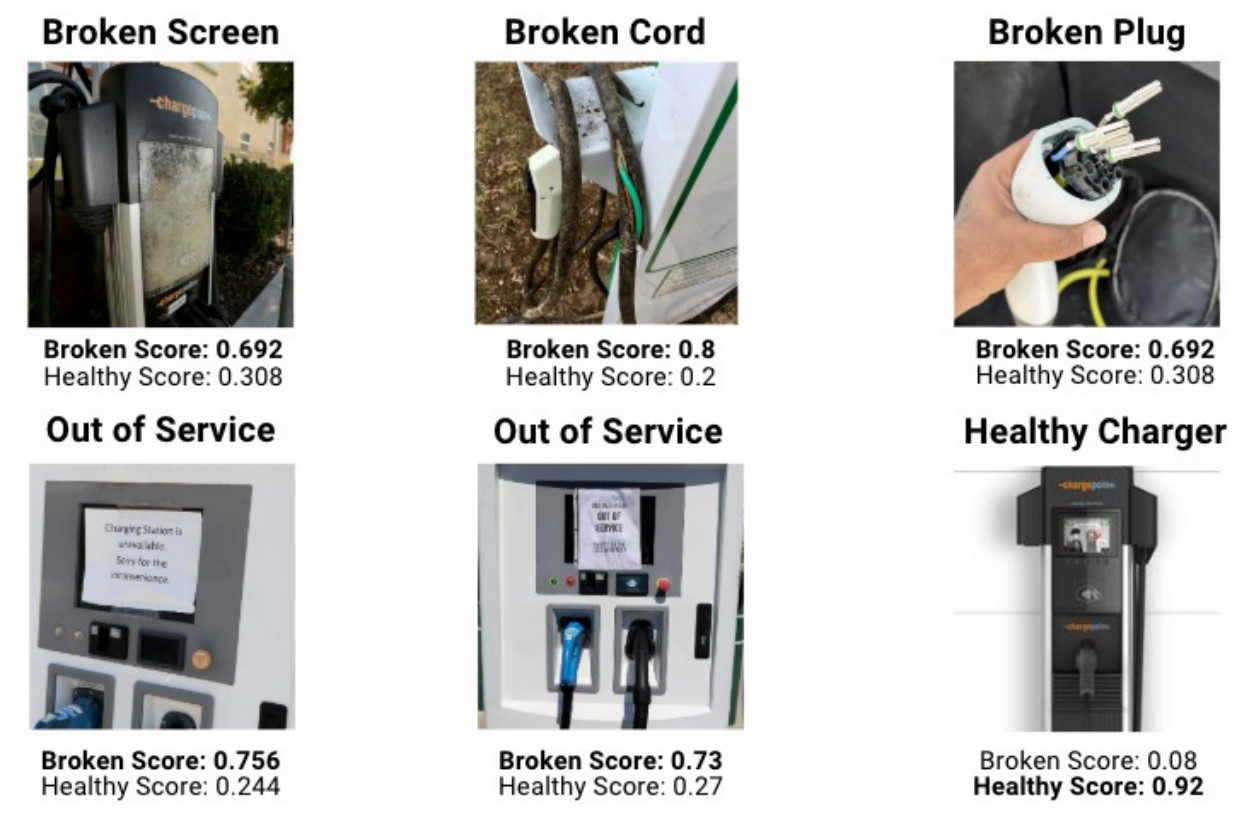


Figure 10: Classification scores on EV chargers from Hugging Face Vision Transformer model

However, classification accuracy was challenged in cases where damage was subtle, such as slight screen discoloration, minor abrasions on plugs, or partially detached cables. Low-light environments and obstructed views occasionally led to misclassifications, as the model relied heavily on clear structural differences to distinguish between healthy and broken components. Reflection from charger screens or cables intertwined with external objects also contributed to variability in classification accuracy.

Despite these limitations, the ViT model significantly improved fault detection by leveraging attention-based feature extraction, capturing both global and local dependencies within images. This resulted in a more precise classification framework compared to traditional convolutional architectures, which often struggle with spatial relationships in complex scenes. The ability to generate confidence scores for each class also enabled a more granular understanding of model performance, allowing for potential threshold tuning to optimize false positive and false negative rates.

4 Discussion

The results of this study demonstrate how effective semantic segmentation and classification models can be in automating EV charger fault detection. By leveraging Nvidia's MIT-B SegFormer models for segmentation and Vision Transformers for classification, we successfully identified and categorized charger faults with a relatively high accuracy, given our limited dataset of 167 labeled images.

4.1 Segmentation Discussion

The segmentation results (Figure 8 indicate that model performance improves as we scale up the Segformer MIT model to more complex architectures. The MIT-B3 model, which has the highest number of parameters (45.2M), achieved a mean accuracy of 87.1% across classes, an overall accuracy of 93.5%, and a mean IoU of 0.804. This performance exceeded our expectations as our dataset was extremely limited compared to the scope of what we were trying to classify. There are well over 10 different EV charger companies, with Chargepoint, EVgo, Electrify America, and Blink just to name a few, and each company has a few different charger types to classify. Despite this wide range of images, MIT-B3 model proved to be robust for our segmentation needs. However, there are limitations with our approach because each one of our images needed to be manually collected and labeled before model training could begin, leaving only a small subset of images for testing that were not used in the training model. Minor misclassifications were observed in low-light or cluttered backgrounds, suggesting that further improvements could be made through data augmentation and additional fine-tuning.

4.2 Classification Discussion

The classification models (Figure 10) demonstrate a relatively high confidence in distinguishing healthy from broken charger components. Each Vision Transformer model was trained specifically for one component (screen, plug, cable, or charger status), resulting in accurate predictions. The models performed particularly well on high-contrast images where damage was clearly visible. However, misclassifications occurred in cases where charger damage was subtle or partially obstructed, indicating a need for additional labeled data, especially in challenging environmental conditions.

One key advantage of the classification approach is its ability to complement segmentation results. By first segmenting the charger components and then classifying their condition, the system ensures that fault detection remains both granular and accurate. This dual-step approach minimizes false positives, ensuring that only relevant components are analyzed for damage classification.

Again, one of the key limitations to this classification is our limited-sized dataset, as finding many images of broken charger components is difficult. However, this Vision Transformer model still gave promising results when classifying a variety of EV charger defects.

To enhance future classification accuracy, additional training data encompassing a broader range of charger models, lighting conditions, and environmental occlusions should be incorporated. Further fine-tuning through data augmentation techniques, such as adaptive contrast adjustment and synthetic occlusion, could help the model generalize better across diverse real-world conditions. Additionally, integrating multimodal data sources, such as error logs from the chargers themselves, could provide supplementary context to refine predictions and reduce ambiguity in challenging cases.

4.3 Implications for Fault Detection and EV Infrastructure

The results highlight the feasibility of deploying automated fault detection in EV charger networks. Compared to existing reporting methods—where users manually describe issues in PlugShare or call maintenance teams—our model provides structured, real-time, and standardized fault reports. By integrating these results into a mobile app interface (Figure 7), users can submit structured reports with greater accuracy, and utility companies can act on more reliable, data—driven insights.

Furthermore, the segmentation model's high IoU score suggest that real—world deployment is reliable, given proper calibration. However, real—world performance could be refined with more data in diverse conditions, such as varying lighting, weather and charger types. Additionally, incorporating active learning—where the model continuously improves by retaining on new reported faults— could help sustain and even improve performance over time.

5 Conclusions

The FLARE project combines manual and online image collection, computer vision-based fault detection with Vision Transformers, and a structured reporting interface to improve EV charger reliability. The current model development focuses on state-of-the-art segmentation techniques with transformer models, providing a foundation for future deep learning-based improvements. The mobile app is designed to integrate with PlugShare, ensuring real-time updates and improved charger fault visibility.

The results from our project confirm that deep learning-based fault detection with vision transformers for EV chargers is a viable and scalable solution. By integrating semantic segmentation with classification models, we demonstrate a system that automates fault logging and enhances reporting efficiency. With further development, this approach has the potential to improve EV charger maintenance, reduce downtime, and boost consumer confidence in charging infrastructure, ultimately accelerating the adoption of electric vehicles.

5.1 Future Work

Since this project is just the beginning to demonstrate the concept of fault detection for EV chargers, FLARE has many areas for growth to make it truly remarkable.

One of the key areas for expansion is developing a back-end infrastructure to support a fully functional mobile application. While our current prototype demonstrates fault detection capabilities, integrating this technology into a user-friendly app would allow EV owners to report charger malfunctions efficiently. A robust back-end system would manage user submissions, store fault reports, and facilitate communication between users and service providers.

Another critical enhancement is integrating PlugShare's API to expand the app's functionality. PlugShare is one of the most widely used platforms for EV charger availability and status updates, and integrating its API would streamline real-time reporting. This would allow FLARE to not only detect faults but also update charger statuses dynamically, making the app more practical and appealing for widespread adoption.

To address privacy concerns, we plan to implement automated face and license plate blurring in user-submitted images. Since users will be capturing photos of public charging stations, ensuring privacy protection is essential. By integrating computer vision techniques for anonymization, we can prevent the collection of sensitive personal data while still preserving the necessary details for fault detection.

Finally, to ensure that FLARE continuously improves over time, we aim to implement data feedback loops and continuous model training. User-submitted images and reports will be incorporated into our training pipeline, allowing the model to refine its classification accuracy iteratively. This approach will help the system adapt to new types of failures, changing charger designs, and real-world conditions, making FLARE increasingly robust and scalable.

By focusing on these future enhancements, FLARE has the potential to transform EV charger maintenance into an efficient, data-driven process, ensuring greater reliability and user trust in public charging infrastructure.

6 References

- **Strudel, Robin.** 2021. Segmenter: Transformer for Semantic Segmentation. [Link]
- **Nvidia.** 2021. SegFormer (b3-sized) encoder pre-trained-only [Link]
- **Rogge Niels.** 2022 HuggingFace Blog: [Link]
- 2024. Segments.ai Documentation: [Link]
- **Thomton, Mark.** 2018 Wood Mackenzi. EVs up to 67% less emissions intensive than ICE cars: [Link]
- **Google.** 2023. HuggingFace Vision Transformer (base-sized model) [Link]
- 2021. HuggingFace SegFormer [Link]
- Daniel Kong's HuggingFace for Initial Segmentation Model using MIT-B0 [Link]
- Irving Zhao's HuggingFace for Semantic Segmentation using MIT-B3 Models[Link]
- 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale[Link]
- 2021. HuggingFace Vision Transformer (ViT) [Link]