AI / Machine Learning – On the Edge

We built a **Helmet Detection App** for a client for installation onto an eBike. We delivered a working product that met their requirements and gave them the peace of mind they needed.

The App was designed for an **eBike Fleet Manager in India** that needed to protect their business owners from the liability of riders being irresponsible and not wearing helmets. In India, cameras are installed beside the roads and check riders are wearing helmets. If they are not, then a ticket is sent to the owner of that eBike. As a fleet operator of eBikes, they get access (and the bill), and without direct evidence, the rider may not be liable to pay the fine, but the fleet operator has to.

We built the App from scratch using **TensorFlow**, Google's Machine Learning platform, coded in **Python,** being one of the best platforms. The App was built on **Android**, which has a 70% market share, and low-cost IP-rated phones are available globally.
We collected the images by asking people we knew to take videos of themselves while wearing a bike helmet and while not wearing a helmet. We then compiled the videos into two separate collections for each condition.

We used a program called FFmpeg toconvert a video into images needed for our model. This program takes a video and splits it up into individual frames. It then saves these frames as image files. Each print is labeled as wearing or not wearing a helmet.
We split the images into two groups to build a model and check its predictive power. We randomly selected 70% of the ideas for training the model, and the remaining 30% were used as the test set.

Working with TensorFlow, we built a model to predict bikers wearing and not wearing their helmet. TensorFlow is a software library for data analysis and machine learning. It is used to create and train neural networks, which are algorithms that can learn to recognize patterns in data. TensorFlow works by transforming a description of a neural network, described in Python (code), into a mathematical function that can be

executed on a computer. This function is called a “graph,” It can calculate the network’s output for any given set of inputs.

TensorFlow used the training data to optimize the model by adjusting the weights of the neurons in the network. This allowed it to learn how to best predict the output given the input data. It also used the testing data to verify that the model could accurately predict the outcome for new data sets.

To change our model in TensorFlow, we used back regression. Back regression is a technique for fitting a linear/logistic regression model to data that contains unknown (but potentially correlated) regressors. We used it to determine the essential features of our data to predict the output.
We programmed the model into our app to check the rider’s progress every few seconds. This allowed us to field-test the model. The App collected all the images during our field testing and labeled them as wearing or not wearing.

At first, the model produced many Type 1 and Type 2 errors. But by using the collected images as training data with correct labels, we improved the model. The labeling was done manually at first, but to scale, our team used Amazon’s Mechanical Turk to label images correctly. We created a task on the site that asked workers to identify the objects in each image. We then used the results of those tasks to train our machine-learning model to label images automatically. The model improved its accuracy the more testing we did.

Of concern was the many camera artifacts we discovered that would ruin the image and create errors. We found that the artifacts were caused by lens flare, over-exposure, saturation, exposure, and noise. These artifacts exist in all camera images to some degree. But worse in our case, as the camera was only a low-cost cellphone camera and was pointed upwards toward the rider and the background, including the bright sky and overhead lighting. These are difficult to remove from the image.

Our workaround was to eliminate these images from the “data set.” In other words, we would not count the result when we detected these artifacts. To do this, we used multiple techniques:

1. The App took multiple images in a row; if they all agreed with the result, then and only then would we post the result.
2. The App processed the images for image quality. One metric is how much high-frequency data is in the picture. Low frequency often means a low-quality image.
3. We checked the image for saturation, especially in the lower or center part of the image where the face and helmet would be.
4. We developed a face recognition model to process the image and ensure it could see a face. If it could recognize a look, then we knew the image quality was good enough to determine a helmet (or not) accurately.

In the field, we expect helmet detection to improve over time as we collect more training data. This will overcome the problem of “overfitting.”

Overall the project was a great success, and the App worked as expected.

(*) The numbers in the Case Study are illustrative only and not intended to be accurate.

Pete Cooper is a CEO and Program Manager with 20+ years of diverse experience as a Program Manager and eight years as a CEO. His career started as a design engineer and grew to the executive level. He has worked in various fields, including Software Development, AI/ML, Product Design Aviation, App development, RF design, Electronics Design, Mechanical Design, Telehealth, Semiconductors, IoT, and more.

Pete is a thought leader in applying Program Management methodology as a CEO. He has received recognition for overseeing complicated projects in various sectors. He holds an Engineering Degree, MBA, an Airline Pilot's Licence, and multiple Program Management Certifications, including FAIPM.

At Skillion, where Pete is the CEO, we pride ourselves on our ability to implement and educate Program Management woven into our customer projects. If you need more than just a technical solution but need it managed end to end, don't hesitate to get in touch with us today to learn more.

info@skillion.tech

Pete Cooper