Operationalize Deep Learning-Based Helmet Detection on Edge Devices

Jane Hung, Gerardo Mejia, Krishnaprasad Kongoorppalli, Pradeep Chelpati

Abstract

According to Occupational Safety and Health Administration (OSHA) 5,333 workers died on the job in 2019 (3.5 per 100,000 full-time equivalent workers) — on average, more than 100 a week or about 15 deaths every day. In 2019, about 20% (1,061) of worker fatalities in private industry were in construction – accounting for one in five worker deaths for the year. In very large industries, such as construction, oil, and gas, workers are constantly exposed to various health hazards. Injuries and death not only affect the workers and their families but also have significant delays and costs to the projects. Personal protective equipment (PPE) is equipment worn to minimize exposure to hazards that cause serious workplace injuries and illnesses. Personal protective equipment may include items, such as gloves, safety glasses and shoes, earplugs or muffs, hard hats, respirators, or coveralls, vests, and full bodysuits. In this paper, we focus only on hard hats/helmets. There is significant effort to ensure PPE equipment is used properly and employers also provide training to the workers. To ensure compliance of PPE equipment, a lot of large industries have CCTV cameras monitoring continuously and there are some challenges with the approach. In this paper, we present a deep learning model to verify PPE compliance of workers, i.e., if a worker is wearing a hard hat/helmet or not in real-time. We have used MobileNet-SSD v1 and VGG16-SSD as part of the project. While we can detect a “helmet,” we would not operationalize any of these models; we don't always detect that it’s being worn. Additional helmets detected in this manner leads to a system that thinks people are being safer than they actually are. Outside of the model training, good video quality and reduced prediction latency are required to be effective.

Introduction

In large industries such as construction, oil, and gas, many people work at job sites that have unsafe working conditions, and thousands lose their lives each year. According to Occupational Safety and Health Administration (OSHA) 5,333 workers died on the job in 2019 alone and this has been an uptrend from the past few decades. The fatality and injury rates in developing and Asian countries are much higher given the volume of construction activity. The leading causes of the construction site fatalities were falls, slips, being struck by objects, electrocution, and being caught in/between objects. In most fall incidents, the workers fall from heights and hit their heads on hard floors. The origins of the fall incidents are from ladders, roofs, buildings under construction, platforms, or scaffolding. Hard hats/helmets are designed to resist shock and penetration by objects as well as contact with electrical hazards. If workers wear hard hats properly, half of the fatalities due to falls and a significant number of fatalities due to slips, trips, and being struck by falling objects could be expected to decrease.

Figure 1. Typical construction site

To decrease injuries by enforcing work-related safety rules, the US government established OSHA in 1970. OSHA prepares guidelines for workplace safety and offers grants for safety training to construction workers in order to train them about the importance of using PPEs properly. In addition, OSHA monitors construction sites to make sure that contractors and owners follow the safety rules to avoid injuries at the site. Due to various reasons, workers at construction sites sometimes fail to obey the OSHA rules and regulations, for example, under extreme weather conditions or due to stress in meeting deadlines. Construction executives are looking for more modern solutions to enforce compliance of PPE detection, mainly hard hats in real-time.

For this project we did the following:

- We started with the Single Shot MultiBox Detection (SSD) model. We investigated the possibility of using model architectures, like MobileNetv1 and VGG16-SSD.
- We re-trained these models with our hardhat/helmet dataset and compared the performance of the different models after being re-trained.
- We trained the model on AWS using the NVIDIA Deep Learning image and PyTorch.
- We operationalized the models on an IOT device (Jetson Nano) and a webcam (Figure 2).

Figure 2. A schematic of a helmet detection model used in construction safety monitoring systems.

The next sections talk about the dataset we used, our model architecture, and results. We end the paper with our lessons learned during this whole project.

Literature Review

Existing techniques of automated monitoring of PPE compliance can be broadly categorized into sensor and vision-based. Sensor-based techniques involve installing RFID tags on each PPE component (i.e., on hard hats) and checking these tags with a scanner at the job site, which does not involve continuous monitoring. Vision-based includes collecting images at a job site to verify PPE compliance and is not done on a real-time basis. In this paper, we looked at real-time monitoring of hard hats using MobileNet and VGG models.

Database

Kaggle dataset

Our 5000 image dataset[6] from Kaggle is 416x415 pixels and is auto-oriented. This dataset is a collection of images of workers in workplace settings that require a hard hat and includes example pictures taken from worksites where people wore hard-hat during their work.

Figure 3. A sample training image (left) and the XML file with class annotations (right).

An annotation file accompanied with images contains details of the object (helmet/head/person) being detected and pixel details of the bounding box.

We have a healthy dataset as all images provided had annotations accompanied and we get an average of 2 bounding boxes for the images. Three classes (helmet, head, and person) were used to annotate the images and detect objects present in the image.

Figure 4. Examples of true annotated labels.

Our dataset had the following characteristics:

- The backgrounds of the images were diverse, including various construction-site scenes
- Different poses and head alignment were exhibited in the image (e.g., standing, walking, sitting, bending, squatting, among others)
- Images captured person from different viewing angles (e.g., front, back, side, top, and bottom views)
- Images were from both outdoors and indoor and with different lighting

- The number of people wearing hard hats varied in an image

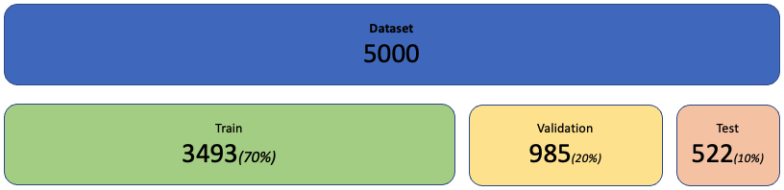

Figure 5. Train, validation, and test dataset breakdown.

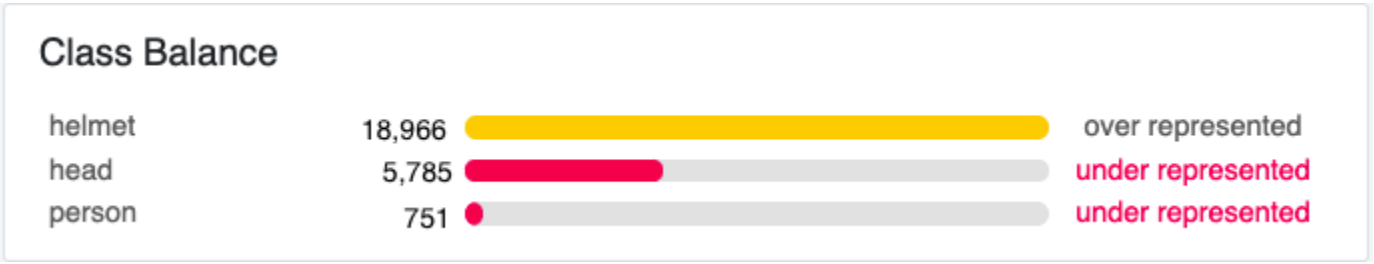

Figure 6. Class imbalance across different objects.

Pascal VOC format[7]
Pascal VOC annotation format was used to generate the annotations for the images in the dataset. This format originally created for the Visual Object Challenge (VOC) has become a common interchange format for object detection labels. Unfortunately, no models can directly consume VOC format labels directly hence we had to generate labeled text data using the Pascal VOC input XML.

Methodology

Methodology - Algorithms
For this project, we chose to use the Single Shot MultiBox Detection (SSD) model. This model combines image classification and object detection in one model and is more efficient than other algorithms like R-CNN and Fast R-CNN which are composed of two steps. This SSD approach is similar to the YOLO (You Only Look Once) model which also does everything in one step, but its mAP performance is slightly inferior when compared to SSD. The difference in the YOLO and SSD approach is that YOLO divides the image into a grid and tries to find the classification score for every box whereas in the SSD approach the neural network does all of the work. We also investigated the possibility of using models architectures, like VGG16-SSD and MobileNet.

We also made use of pre-trained models for our project that were originally trained on the ILSVRC-2012 and COCO datasets. We re-trained these models with our helmet dataset and compared the performance of the different models after being re-trained. The metrics we used were the Jaccard overlap, which is the intersection over the union, or in other words, the ground truth box will be matched to the default box with the best Jaccard overlap.

Methodology - Tools
For this project, we used a variety of tools, technologies, and services. In terms of physical hardware, all the team members had the following:

- Laptop/Computer
- NVIDIA Jetson Nano 4GB
- USB Camera

In terms of software, we made use of the following:

- Tensorboard
- NVIDIA Deep Learning AMI
- OpenCV
- Jetson-Inference
- Pytorch-SSD

We also used the following AWS services to speed up our training of the models:

- AWS EC2
- AWS S3

We downloaded the Kaggle dataset to our laptops, which we then processed and saved into S3. We then loaded the dataset into a g4dn.2xlarge EC2 machine running Nvidia Linux and trained using PyTorch. The EC2 instance had hardware consisting of 1 GPU, 8 virtual x86_64 CPUs and 32GB of RAM. When the training was done, we saved the model weights back into S3. Finally, we downloaded the model weights from S3 into the Jetson, which we controlled using a VNC remote display client in our laptops and a VNC server in the Jetson Nano, which has inferior hardware when compared to the EC2 instance. The specs of the Jetson Nano are 4GB of RAM, 4 ARM A57 CPUs running at 1.47GHz, and 1 Maxwell GPU with 128-cores. This hardware was just enough to run inference on the models we trained.

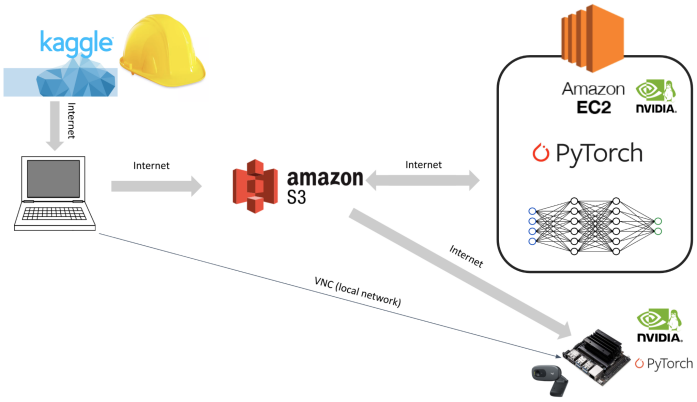

Figure 7. Project implementation architecture.

Results

Model training

According to the figures below, the VGG model seemed most promising during model training compared to the updated MobileNet model. Juxtaposed with the default MobileNet model, the VGG model reached comparable loss values (~2.5), which suggests that the VGG model may outperform the default model in practice (Figure 8).

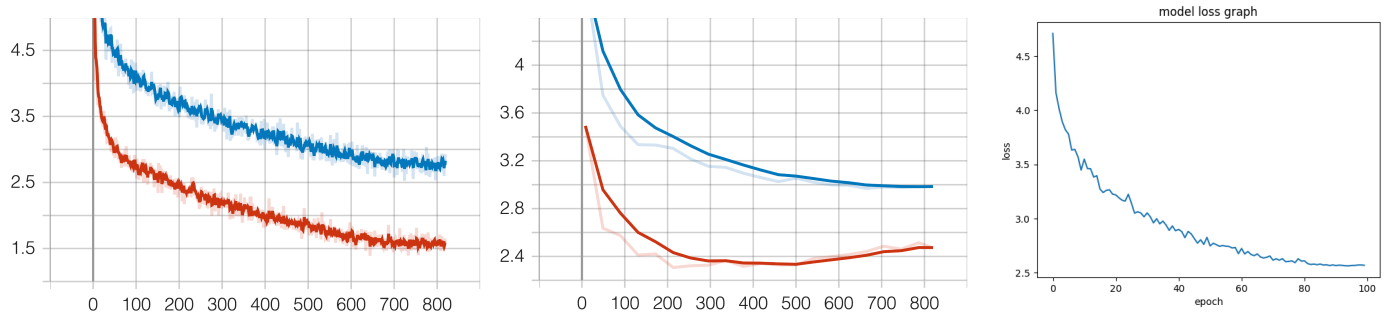

Figure 8. Tensorboard visualizing training (left) and validation loss (middle) across our two final valid experiments – updated MobileNet (blue) and VGG (red). Validation loss for the default MobileNet model (right).

Model evaluation

Table 1 below demonstrates that the VGG model outperforms both sets of MobileNet models in average precision in all cases except predicting “person”. Average precision for the “person” class is extremely low in comparison with the other objects since those labels only make up 3% of all annotations. Focusing on helmet precision allowed us to reduce false positives detected, which would encourage higher compliance with safety measures.

Table 1. Average precision across classes when predicting on a test set of data and detection time on a single helmet image.

Model	Average Precision - helmet	Average Precision - person	Average Precision - head	Average Precision - all classes	Detection Time of One Helmet (sec)
Default MobileNet	8.7358e-06	0.0	0.0043	0.0014	1.2756
Updated MobileNet	0.5744	**0.0101**	0.3726	0.3190	**1.2786**
VGG	**0.7986**	0.0058	**0.6971**	**0.5005**	1.2907

As shown in Table 1, the updated MobileNet model (27MB) has the fastest detection time when detecting a single helmet and the VGG model has the slowest detection time, which suggests that we may experience issues when required to extend the model onto streaming data. While video and prediction quality may be affected by the use of the larger VGG model (92MB), practically speaking, this construction safety use case may allow for a degree of delay. Indeed, the VGG model is the most confident in our helmet prediction compared to the default MobileNet, which gives further evidence that this model would operationalize well (Figure 9). However, the VGG model required twice as much training time over 100 epochs compared with the MobileNet model, which demonstrates that the model is much more costly to train.

Figure 9. Default MobileNet (left), updated MobileNet (middle), and VGG (right) predicting correct helmet wearing.

Jetson operationalization

When the models were deployed onto the Jetson device and applied onto a live video stream, we were better able to identify how useful these models are in construction systems. There does not seem to be a noticeable difference between the models when there is someone wearing a helmet as shown by the high probability of “helmet” detections. However, we can see that there is less confidence in detecting “head” across all the models. Recorded video demos can be found here: updated MobileNet and VGG.

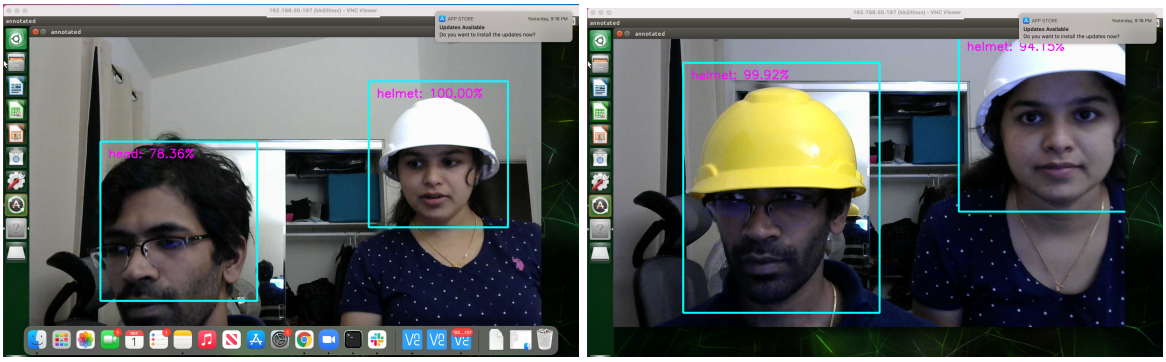

Figure 10. Default model predicting “head” (i.e., no helmet is worn) and “helmet” (helmet is worn) (left). Default model predicting “helmet” for multiple people captured on camera (right).

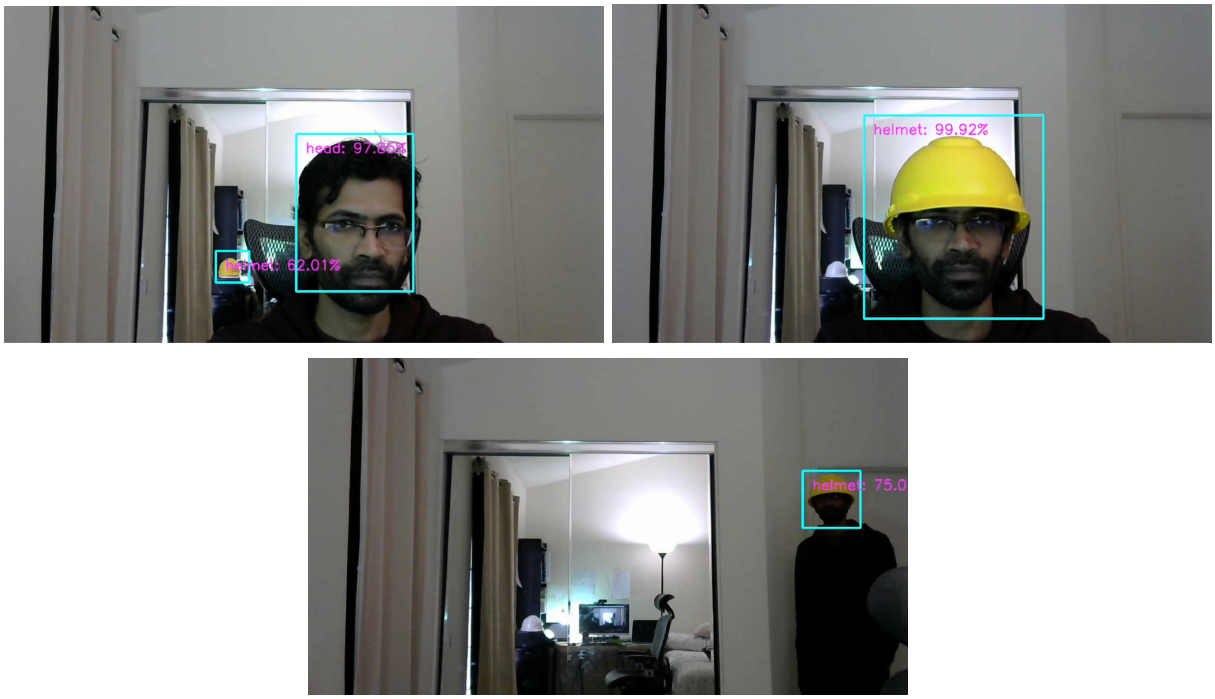

Figure 11. Updated MobileNet model predicting “head” and “helmet” when a helmet is simply shown on screen (top left), “helmet” when someone is wearing a helmet (top right), and “helmet” with less certainty in shadows (bottom).

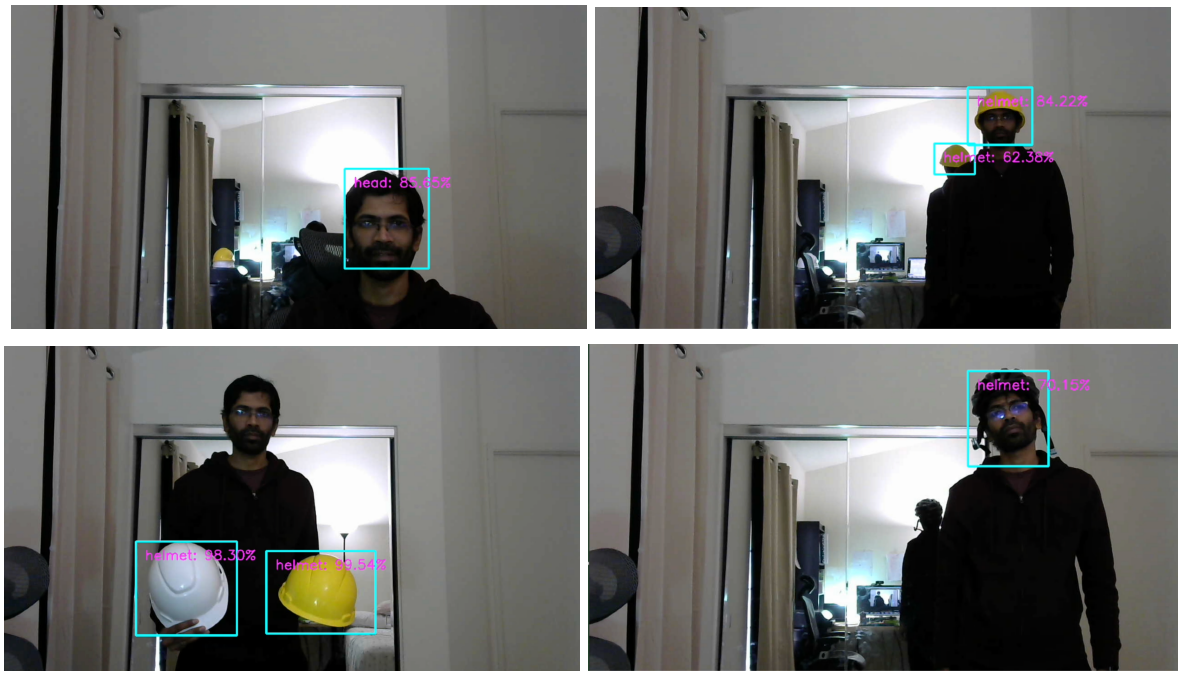

Figure 12. Trained VGG model accurately predicting "head" (top left) and "helmet" (top right). Notice also the helmet detected in the mirror (top right) and when held (bottom left). Trained VGG model not detecting "head" (bottom left) but able to detect other types of helmets (bottom right).

All models provided high fidelity predictions when shown a person without a helmet (i.e., "head") and a person with a helmet (i.e., "helmet"). However, both models also identified helmets that were sitting on tabletops or were being held, such that the helmets were not actively on a person's head, which suggests that these models if implemented, cannot identify whether construction workers are demonstrating correct safety protocols (Figure 13). This major flaw demonstrates that the models would not be effective in real-time construction environments without further manipulation of input data to identify "person with helmet". However, the speed of predictions and the size of the model checkpoint files, when operationalized within the Jetson Nano, are redeemable qualities that suggest that the smaller, agile MobileNet model would be effective on edge devices.

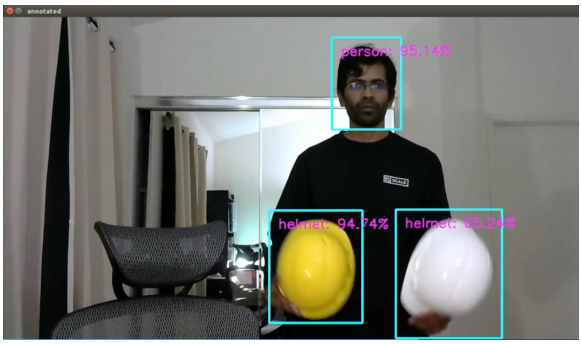

Figure 13. Updated MobileNet model predicting "helmet" when a person is not wearing a helmet.

Table 2. Video demo data rate

Model	Data Rate
Updated MobileNet	**5.99 Mbit/s**
VGG	4.50 Mbit/s

Video bitrate on our trained models demonstrates that the MobileNet model produces a system with better video quality, which may be more conducive to active video monitoring.

Discussion

Limitations

Our dataset format proved to be a limiting obstacle early on in our development. To compare to the baseline model as shown in Hackster.io, we reused the Pytorch-SSD scripts, which forced us to format our data using Pascal VOC despite our preference and familiarity with YOLO v5. During this aspect of our project, we struggled with PNG to JPG conversion, Pascal VOC's preferred file structure, and understanding and creating the ImageSet (i.e., a delineation between train, test, and validation records). Furthermore, Pascal VOC combines additional attributes, such as "truncated" and "difficult," which were necessary for this SSD implementation and made it impossible to use the YOLO v5 format without significant restructuring of the dataset formatting code.

Reflections

We primarily compared validation loss across experiments to identify our most accurate model. However, when operationalizing this model in the Jetson Nano, we found that lower validation losses did not correlate with significantly more robust helmet/head predictions compared with other experiments. In hindsight, the mean average precision (mAP) and helmet precision and recall would have been more impactful comparison metrics because they can better describe how these models would have performed operationally. For example, in our case, predicting the bounding box coordinates may be less useful than correctly classifying the helmet, person, and head classes, so prioritizing classification loss higher than regression loss could have further advantages in the construction workplace. While we were able to calculate these evaluation metrics to compare our two best model iterations, it may have been ideal to extend to all of our model trials to better compare.

Ideally, we would also include metrics detailing prediction latency and system UI latency that were negatively affected by the size of the VGG pre-trained model used for this project. While we could qualitatively identify these metrics when the experiment artifacts were deployed to the Jetson, we did not recognize the importance to overall system performance until further along in our efforts. Indeed, excellent prediction and system UI latency would be prioritized requirements for actual real-time usage of the models. We attempted to answer these questions through the demo data rate (Table 2) and the single helmet prediction time (Table 1), but generalizing this effort across multiple samples would ideally identify the best model for real-time use.

Lastly, the Kaggle dataset annotated three types of objects – helmet, person, and head. As a result, our models were trained to predict these three objects, which we realized would not be as applicable to our overall use case. That is, often the models would predict "helmet," but there was no one wearing the helmet. In practice, we were looking for the model to predict "helmet" and "head" bounding boxes with significant Jaccard overlap, which would better indicate that someone was actually wearing the helmet. The models would also predict "head" and "person" interchangeably when there was a person not wearing a helmet, which suggests that these two classes were not unique enough. Indeed, these two were severely imbalanced classes that would warrant support from a secondary data source. Ideally, we should have reorganized this dataset to predict "head wearing helmet" and "head not wearing a helmet." In doing so, we would better monitor safety protocol adherence in construction zones.

Conclusion

While our models successfully detected helmets when processed on an edge device, there are several scenarios where our models do not effectively monitor and encourage hard hat-wearing in dangerous construction zones. As a result, we need to continue to refine our models using methods to specifically focus on a person wearing a helmet and a person not wearing a helmet in order to improve safety compliance on construction sites.

References

Code can be found here.

[1] https://github.com/smlblr/Real-Time-Detection-of-People-not-Wearing-Hardhat

[2] https://github.com/qfgaohao/pytorch-ssd/

[3] https://aws.amazon.com/ec2/instance-types/g4/

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7472390/

[5] https://developer.nvidia.com/embedded/jetson-modules

[6] https://www.kaggle.com/andrewmvd/hard-hat-detection

[7] https://roboflow.com/formats/pascal-voc-xml

[8] https://ascelibrary.org/doi/abs/10.1061/(ASCE)CO.1943-7862.0000974

[9] https://www.sciencedirect.com/science/article/abs/pii/S0926580517304429

[10] https://www.hindawi.com/journals/ace/2020/9703560/

[11] https://www.hindawi.com/journals/jcen/2015/721380/

[12] https://arxiv.org/pdf/1704.04861.pdf

[13] https://arxiv.org/pdf/1409.1556.pdf