Augmented reality in firefighting environments using embedded deep learning system

Sushmitha Kasimsetty Ramesh, Akhil Penninti
ECE 528: Embedded Systems and Machine learning
Prof Sudeep Pasricha
Colorado State University
Fort Collins, CO, USA
sush1997@colostate.edu, akhil.penninti@colostate.edu

‘

***Abstract*—A firefighter will have extensive training in firefighting, and one of their main responsibilities will be putting out dangerous fires that harm lives, property, and the environment. In some circumstances or jurisdictions, they may also be tasked with saving animals. Maintaining situational awareness is essential for firefighters to make the precise decisions needed to move through a fire environment safely and successfully. The uncertainty brought on by threats like smoke and high heat may result in injury or even death. It could be useful for creating an augmented reality for the responding firefighter. A technology-based method of engagement called augmented reality overlays virtual objects on the real world. Augmented reality has expanded as a result of the development of cameras, improvements in silicon processing effectiveness, and more advanced artificial intelligence algorithms. We have developed a prototype embedded system that can analyze incoming data in real-time using deep learning models that have already been generated and data streaming from cameras placed into a firefighter's personal equipment. Therefore, by employing our modified Mask R-CNN model, which is an extension of the Faster R-CNN network structure, we use instance segmentation for a sharper emphasis on specific targets of the same class through object localisation and finally achieve a pruned and quantized accuracy of 98.39%. [6]**

***Index Terms*—Firefighter, Embedded system, Augmented reality;**

I. Introduction

[10]Today's firemen face more complex challenges than merely the risk of fire. Even in the most commonplace structure fire, there are a number of concerns to consider, such as exposure to hazardous gases and explosions. Combustion can result in the production of harmful toxic gases. Understanding these dangers will improve fire scene security and reduce accident rates. Being a firefighter is a very hazardous and difficult job. During firefighting operations, members were observed utilizing easily accessible, personal, and optimal monitoring instruments to assess the types and levels of exposure they experienced while doing their routine job activities. The results demonstrate that dangerous elements including HCl, CO, SO2, and many others are frequently present in high amounts in the air that firefighters breathe. Since respiratory protection equipment was not worn in many cases with the highest exposure to these compounds because it seemed that the smoke was not very intense, these numbers accurately reflect the firefighters' actual direct exposure. Smoke is frequently detected at a wide distance and without physical contact. Smoke detection can spot flames and identify fire conditions more quickly and effectively in order to monitor and prevent fires. Over the past few years, these applications have grown to now encompass wireless networks for early fire detection and response as well as autonomous VR platforms for firefighter training. These solutions do not adequately address the numerous challenges that fighting a fire entail for the firemen themselves. Living in a continually shifting, the life-threatening scenario can be extremely stressful and confusing, which can impair judgment and decrease monitoring. Studies across the board have revealed a gap between the knowledge that is available in fire scenarios and the knowledge required for crucial situational awareness, as well as the importance of precise information for making decisions without error. A firefighter's ability to maintain situational awareness under these conditions is essential for making accurate decisions, which require real-time information on the rapidly changing conditions at the scene. The issue we'll be looking into has to do with using augmented reality in firefighting situations to raise the bar for worker safety. We thought this to be noteworthy because they work in very dangerous circumstances, and it is crucial for them to maintain situational awareness in order to make wise decisions. [5]For the responding firefighter, it would be exciting to employ augmented reality. A wide range of intriguing scenarios can use this technology. For example, firefighters may encounter situations when they are unable to see because of thick smoke or when there is no visible light, in which case they will need to use a flashlight to illuminate the scene. For better visual acuity in these circumstances, thermal cameras are often helpful. Examining thermal images, however, is more challenging than doing so with visible-light photographs. It is possible to incorporate the capacity for the firefighters to reconstruct the situation using augmented reality goggles. But here, using an existing dataset, we intend to use a deep learning model to distinguish between distinct types of thermal/infrared images. We will initially divide the data and train the model using a Fusion of a depth-wise separable image classifier and our custom-built model (based on CNN with pooling, dense, and dropout layers) is used to classify images. Then, based on the findings from our research, we will choose the optimum mix/dual quantization strategy to shrink

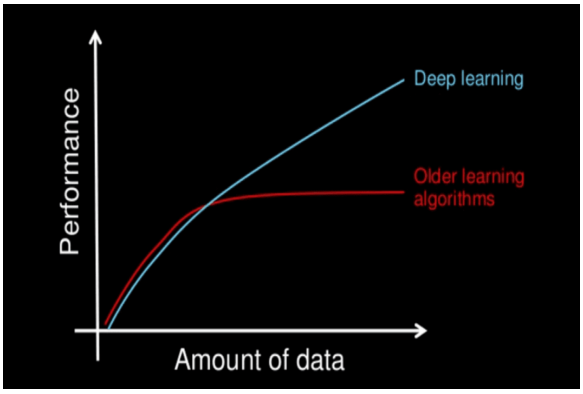

Fig. 1. Deep learning model

the model size and boost overall effectiveness. Regarding the hardware, the proposed software is combined with low-cost thermal imagers, an NVIDIA Jetson unit, and a buzzer where, after the image is taken by the camera and pre-processed using the Jetson Nano unit and a notification is sent to the user.

II. Related Work

The journal was made by the author as a way to use Raspberry Pi and artificial neural network methods to put out fires. Raspberry Pi performs as a data processing/training data image that generates information/output, regardless of whether there is a fire. According to the data, if there is an actual fire or fire, the person in charge of the nearby warehouse and fire department will be informed. It is commonly known that the study's final results demonstrate that more learning samples will yield more accurate results.
A smaller image will require more iterations, thus the author chooses an image that is 40 by 40 pixels in size because, in his opinion, that is the perfect size

In the fourth piece of literature, the authors employed the Convolutional Neural Network (CNN) method to recognize images of fire. This study uses a dataset of 651 online photographs. In this study, CNN VGG16, Resnet50, and fundamental architecture were compared. The deep CNN architecture works better on the fire detection system, per the test results. Resnet50 is 91.18 percent accurate, VGG16 is 90.19 percent accurate, and CNN is just 50 percent accurate. [7]

If fire is not put out right away, it can seriously harm the environment and the social economy. As the economy and science have advanced, attention to the harm that fire causes to society and the environment has grown. It is essential to have real-time fire detection for both active and impending fires. However, due to technology limitations, reliable and real-time fire detection had many difficulties before then. With the development of computer vision and video surveillance technologies, more accurate fire detection with a need for real-time performance is becoming more feasible and significant. Therefore, an effective fire detection method is significant and helpful. In order to identify fire and smoke, a deep learning-based fire detection approach is proposed in this study. First, the depth feature of the image is extracted using the residual network structure. With upgraded ResNet-50, the issue of shallow features readily dissipating is resolved. For the purposes of fusing and enhancing multi-scale features, a network made up of several BiFPN modules is formed. Using intersection over union and cross entropy, the range and type of border boxes are predicted. Then, by comparing the bounding box's confidence, the prediction results are obtained. The outcomes of the tests show that our approach beats the present detection networks in terms of accuracy and processing speed. The efficacy of this method has been validated in the context of fire detection. [11]

Fire can gravely destroy the environment and the social economy if it is not extinguished quickly away. A reliable fire detection method is so important and useful. In this paper, a deep learning-based fire detection method is suggested for distinguishing between fire and smoke. The residual network structure is first used to recover the depth feature of the picture. The problem of shallow features easily dissipating is overcome with updated ResNet-50. A network made up of many BiFPN modules is created with the aim of fusing and boosting multi-scale properties.This paper proposes a new type of forest firefighting method. When one or more suspected fire spots are found in the forest, the firefighting Unmanned Aerial Vehicles (UAVs) can safely and quickly reach the desired location, conduct a thorough investigation of the suspected fire spots. If a fire occurs, it is necessary to drop firefighting materials to put out the fire. At the same time, considering the characteristics of limited endurance and low load capability have always limited the applications of UAVs, Unmanned Ground Vehicles (UGVs) are considered in this paper as an additional and comprehensive problem solution to provide energy and supplement materials for UAVs.A new enhanced D*Lite algorithm is recommended for the mission team's UAVs in order to enable firefighting UAVs to design a safe and efficient course in real time in the challenging forest environment. Even if the 3D workspace is only partially or completely known, the system may nevertheless calculate a suitable path so that the firefighting UAV can move to the fire zones and UGV in real-time. [8]

The development of information and communication technology (ICT) has removed financial and technical barriers and opened up entirely new avenues for action. This results in developments that were simply unthinkable in the past. The MIT Center for Digital Business defines this procedure as "digital transformation," which it defines as "the use of new digital technologies (such as social media, smartphones, analysis, and integrated devices) to enable significant improvements and to increase the performance of the processes in the business environment (such as improve the customer experience, streamline operations, or create new business models)." One of the fun-

damental technologies enabling the digital transformation of both industrial and non-industrial sectors is augmented reality technology. Applications for virtual and augmented reality are already being created that were previously only viable in huge research laboratories, thanks to the quick development of display hardware, new interface devices, and tracking systems. In fact, modern smartphones and tablet computers are now highly helpful with their cameras, sensors, and computational performance as the foundation for AR systems, despite the fact that most users did not purchase them primarily for AR. For the creation of AR applications, big businesses like Google or Apple offer SDK (Software Development Kit). As a result, there are thousands of brand-new augmented reality (AR) applications out there right now, ready for millions of potential customers who already possess the essentials. The use of augmented reality (AR) in the industrial field is discussed in this paper. Two augmented reality (AR) applications will be implemented, and their potential to digitize product lifecycle processes will be examined, based on industrial use cases. The use cases include the following areas: maintenance engineering and product configuration management. They were created as part of collaborative initiatives between the University of Applied Sciences Karlsruhe and a manufacturer of firefighting apparatus and fire trucks. The examination of augmented reality's use in the sector and its potential as a digital transformation tool throughout the product lifecycle is the goal driving the development of the use cases. The offered use cases provide insights and recommendations about the praxis-oriented development and application of augmented reality technology in the engineering area, which can be used for upcoming augmented reality research projects. [2]

III. Data-set

[2]The data that is been used is Fire and Smoke from Kaggle. More than 7000 photos are included in the dataset Fire and Smoke. It's incredibly convenient that some of the photographs were taken expressly by crowdsourcing contributors because we may include at least one image in each sub-dataset (train, validation, and test). It is largely recorded in 400+ cities and has a resolution of 1920*1080. In order to analyze and obtain photographs from various scenarios, this will help. After that, it is photographed using fire and smoke detection in a variety of lighting situations, including day and night, different distances, and different viewpoints. Some of the pictures are attached here:

IV. Software Methodologies

A. Transfer Learning Fine Tuning

[7] The images from the aforementioned data-set are now identified from a pre-trained network utilizing transfer learning and fine tuning. A stored network that was previously trained on a big data-set, generally on a large-scale image-classification job, for image classification is that if a model is trained on a big and diverse data-set, it may successfully serve as a generic model of the visual world. Thus, transfer learning aids in the instantiation of pre-trained base models and the loading of pre-trained weights onto them. It then freezes all of the layers and builds a new model on top of the output of one or more layers from the base model.//

These outputs are utilized as input data for new data, a portion of the basic model is unfrozen, and the model is completely retrained with a very low learning rate (lr=0.0001). This step must be performed only after the model with frozen layers has been trained to convergence. When randomly-initialized trainable layers are combined with trainable layers

Fig. 2. Fig: Sample image1

Fig. 3. Fig: Sample image2

Fig. 4. Fig: Sample image3

Fig. 5. Fig: Sample image4

Fig. 6. Fig: Sample image5

that include pre-trained features, the randomly-initialized layers generate very significant gradient changes during training, destroying the pre-trained features. It's also crucial to utilize a very modest learning rate at this step because we're developing a considerably larger model than in the first round of training on a generally small dataset.

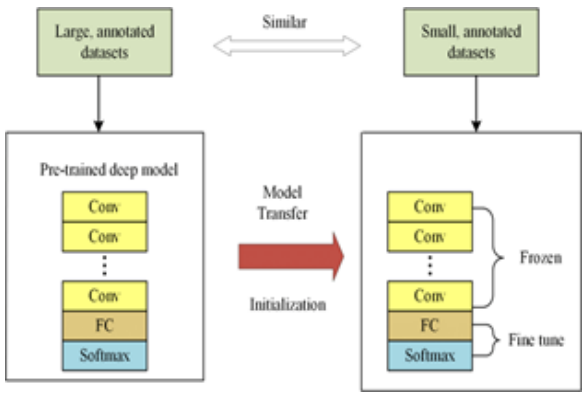

Fig. 7. Transfer Learning Fine Tuning

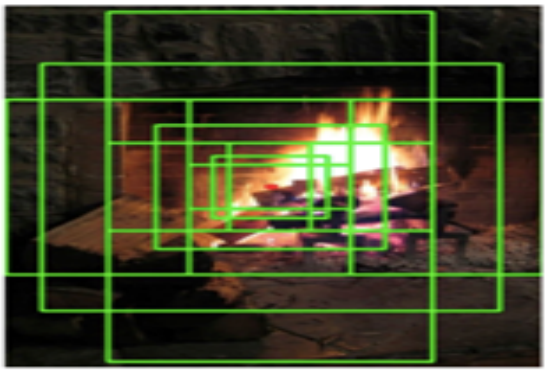

Fig. 8. Generation of anchor boxes

B. Faster-CNN

[9]The Faster R-CNN model creates detection boxes based on anchors, which are the initial set of candidate boxes created during the object detection process. As a result, the Faster-CNN has two phases. The first step, known as a Region Proposal Network (RPN), suggests prospective object bounding boxes.

1) Stage 1: Architecture of the Region Proposal Network: The region proposal network (RPN) initially starts with the input image being fed into the backbone convolutional neural network.

• Depending on the stride of the backbone network, the output features (marked by H x W) are frequently significantly smaller than the input picture.

• The network must learn if an item is present in the input picture at its associated position and estimate its size for each point in the output feature map.

• The network must learn if an item is present in the input picture at its associated position and estimate its size for each point in the output feature map. Each mini-batch for RPN training is generated from a single picture.

• [3]Because sampling all of the anchors from the image would bias the learning process toward negative samples, the positive and negative examples are chosen at random to construct the batch, with extra negative samples added if there are insufficient positives.

• In this case, i represents the index of the anchor in the mini-batch. The log loss over two classes is represented by Lcls(pi, pi*) (object vs not object). Pi is the classification branch's output score for anchor I and pi* is the ground-truth label.

• The regression loss L(t, t) is only active if the anchor includes an item, i.e., the ground truth p* is 1. The word ti denotes the regression layer's output prediction and consists of four variables [t, t, t w, t].

• The four offsets are txr, tyr, twr, and thr, where txr and

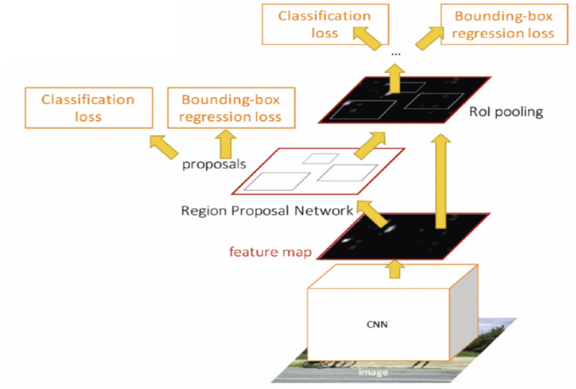

Fig. 9. Structure of Faster R-CNN

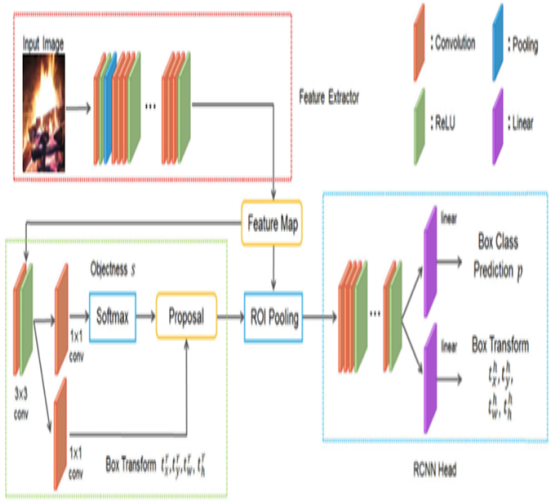

Fig. 10. : Structure of a Regional Proposed Network

tyr represent the offset of the anchor center posi- tion in the X and Y directions of the image, respectively, and twr and thr represent the change in the anchor width and length, respectively. The term ti refers to the output prediction of the regression layer and is made up of four variables.

• Here, x, y, w, and h correspond to the (x, y) coordinates of the box center, as well as the box's height h and width w. The coordinates of the anchor box and its related ground-truth bounding box are denoted by xa, x*.

One thing to be kept in mind is that the regressors in each anchor box have distinct weights. So, given an anchor I the regression loss is applied to its associated regressor (if it is a positive sample).At test time, the learned regression Output ti may be applied to its projected positive anchor box, and the x, y, w, and h parameters for the expected item proposal bounding box can be derived backwards.

2) Stage 2: Architecture of the Fast R-CNN: The second critical stage is the Fast R-CNN, which pulls features from each candidate box using the RoIPool and conducts classification and bounding-box regression. Both stages' characteristics can be shared for quicker inference.

• A CNN backbone, a ROI pooling layer, and fully connected layers are followed by two sibling branches for classification and bounding box regression in the Fast R-CNN detector.

• RoIPool first quantizes a floating-number RoI to the discrete granularity of the feature map, then subdivides it into spatial bins, which are also quantized, and lastly aggregates the feature values covered by each bin.

• To get the feature map, the input picture is first routed through the backbone CNN. Aside from test time efficiency, another important rationale for utilizing an RPN as a proposal generator is to account for weight sharing between the RPM and fast R-CNN detector backbones.

• Following that, the RPN's bounding box suggestion are utilized to pool features from the backbone feature map The ROI pooling layer handles this. In essence, the ROI pooling layer works by,

i) Taking the region corresponding to a proposal from the backbone feature map. ii) Subdividing this area into a set number of sub-windows. iii) Applying max-pooling to these sub-windows to get a fixed-size output.

• The candidate regions are next subjected to non-maximum suppression, in which the region with the greatest object score is screened out. Following non-maximum suppression, each zone is sent into the ROI pooling layer which produces a fixed shape feature map for each region. Finally, the R-CNN Head's linear layer provides the classification score for each area as well as any additional offsets: txh, tyh, twh, and th.

C. Mask R-CNN

[10]Over a short period of time, the Artificial Intelligence community has dramatically improved object detection and semantic segmentation outcomes. These advancements have been fueled in large part by strong baseline systems, such as the Fast/Faster R-CNN and Fully Convolutional Network frameworks for object identification and semantic segmentation, respectively. These approaches are theoretically simple and provide flexibility and resilience, as well as quick training and inference times. As a result, our objective in this effort is to provide a comparable enabling architecture for example segmentation.

Instance segmentation is difficult because it involves accurate recognition of all objects in a picture as well as exact segmentation of each instance. It thus combines elements from the traditional computer vision tasks of object detection, in which the goal is to classify individual objects and localize each using a bounding box, and semantic segmentation, in which the goal is to classify each pixel into a fixed set of categories without distinguishing object instances. While our method detects objects in an image efficiently while also generating a high-quality segmentation mask for

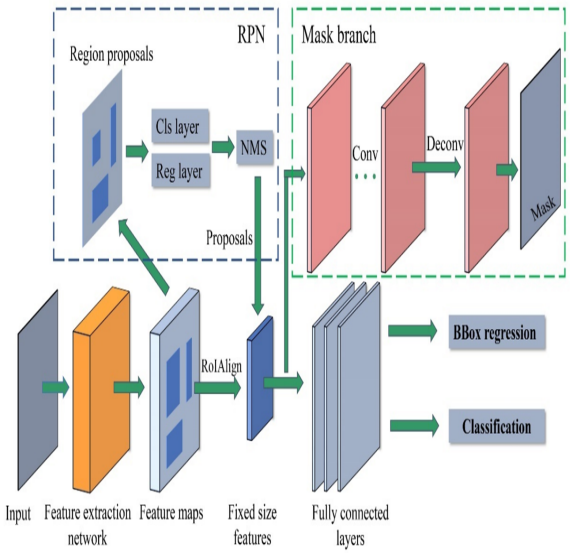

Fig. 11. Instance Segmentation

each instance, Mask R-CNN acts as an intuitive extension of Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is used because it is easy to train and only adds a minor overhead to Faster R-CNN. Most crucially, Faster R-CNN was not designed to align network inputs and outputs pixel by pixel.

D. Procedural Mechanism of the Mask R-CNN:

Mask R-CNN uses the same two-stage technique, with the same first step (which is RPN).

• In the second stage, Mask R-CNN predicts the class and box offset while simultaneously producing a binary mask for every ROI during training, we define a multi-task loss on each sampled RoI as follows:

L = Lcls + Lbox + Lmask.

• The classification loss Lcls and bounding-box loss Lbox are pretty much identical. For each RoI, the mask branch produces a Km2-dimensional output that encodes K binary masks of resolution m*m, one for each of the K classes. We apply a per-pixel sigmoid to this, and Lmask is defined as the average binary cross-entropy loss.

E. Pruning

[4]Pruning is a simple model compression approach in which a huge trained network is pruned of weights, neurons, blocks, and so on. Deep learning models with high performance are large in size. However, the larger the model, the more storage space it requires, making deployment on resource-constrained devices challenging. Furthermore, a larger model requires longer inference time and consumes more energy during inference. While these models performed admirably in the lab, they are unsuitable for many real-world

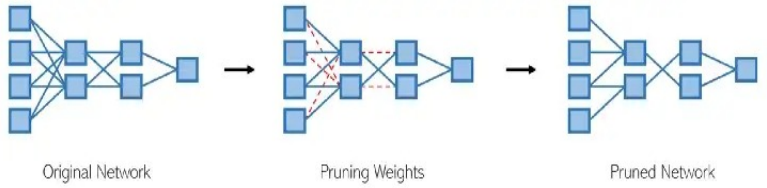

Fig. 12. Fig: Before Pruning

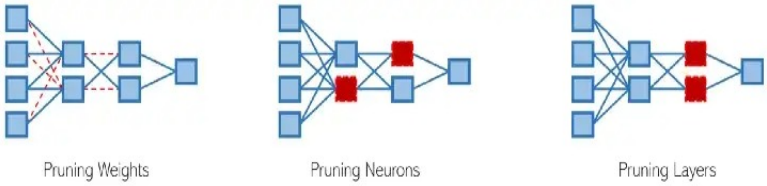

Fig. 13. Fig: After Pruning

applications. Pruning eliminates duplicate parameters or neurons that do not add significantly to the accuracy of outcomes. When the weight coefficients are zero, near to zero, or repeated, this situation occurs.

Pruning connections, on the other hand, results in sparse matrices, which causes some computing complexity. Given the quantity of connections, trimming them is likewise a computationally expensive job. These issues can be avoided by pruning other areas of neural networks.

• Pruning neurons: We may use a magnitude-based strategy to completely eliminate superfluous neurons by taking the average of incoming or outgoing weights. We may also use entropy to do more advanced studies. Pruning whole neurons is easy and frequently successful.

• Pruning blocks: Block-sparse formats store blocks in memory in a contiguous manner to decrease irregular memory access. Pruning memory blocks is comparable to pruning neurons as clumps of network pieces, but it is primarily concerned with hardware performance and energy economy.

• Pruning layers: Layers can be pruned by rule; for example, during training, every third layer is pruned progressively such that the model shrinks slowly yet has time to condense its learning. Pruning can also be based on a mathematical examination of the layer's influence on the model's output.

In our experiments using pruning, we train the model first, then remove unneeded weights to get a substantially smaller yet effective model. We seek to develop a model that can operate efficiently on resource-constrained devices without sacrificing accuracy. It also lowers the computational cost of network training and shrinks the total model size. It also saves computing time and energy, which is important.

F. Quantization

[1]Quantization is a simple but efficient approach of model compression that maintains weights in lower bit representations. In this case, we apply Post-training

quantization, a conversion approach that may minimize model size while also decreasing CPU and hardware accelerator latency with little loss in model accuracy. Because the model compression strategies are complimentary, they may be used as a post-processing step to pre-trained models to reduce model size and boost inference speed. While pruning gets rid of ”unimportant” weights, quantization tries to minimize the number of bits needed to hold the weights. Quantized weights, on the other hand, make neural networks more difficult to converge and make back-propagation impossible. By ensuring that all model math is integer quantized, we were able to achieve significant latency savings, lower peak memory consumption, and compatibility with integer only hardware devices or accelerators.

V. Hardware Components

A. Jetson Nano

A small, powerful computer with modern AI capabilities, the Jetson Nano is ideal for IoT and embedded applications. Jetson Nano offers a quick and simple solution to include cutting-edge AI into your next product since it provides the performance and capabilities required to run contemporary AI workloads.

For the creation of cutting-edge AI solutions for a range of markets, developers use the NVIDIA Jetson series of advanced embedding systems. Jetson is one of the best hardware platforms in the world for cutting-edge AI, making it a great resource for both students and tech enthusiasts to gain practical experience with AI through a variety of innovative projects. Modules are high-performance edge processors with a small physical size that make up the platform. Along with a complete ecosystem to aid in the quick development of original AI applications, it also includes JetPack SDK for software acceleration. The NVIDIA Jetson system is among the best and most popular platforms for developing edge devices with AI thanks to its high performance and power economy.

For AI-based computer vision applications and to carry out tasks like image classification, picture segmentation, object recognition, and more, the NVIDIA Jetson Nano module is used. It is appropriate for use with free and open-source machine learning tools like OpenCV.

Fig. 14. Jetson Nano

Fig. 15. Buzzer

B. Buzzer

An electronic gadget called a piezo buzzer is used to create a tone, alarm, or sound. It has a straightforward design and is lightweight. However, depending on the parameters of the piezo ceramic buzzer, it is also dependable and can be built in a variety of sizes that operate over a range of frequencies to provide various sound outputs.

C. Jetson Nano Camera

The camera that we used here is Compatible with NVIDIA Jetson Nano Camera IMX219-160 8MP IR-Cut Infrared Night Vision Camera Module for Jetson Nano and Raspberry Pi Compute Module,162° FOV with IMX219 Sensor

VI. EXPERIMENTS

We begin the experimentation phase by pre-processing a challenging Fire Smoke dataset and then use the keras models which are pre-trained with weights that can be used for prediction, feature extraction, and fine-tuning. We made use of the VGG 16, ResNet 50 MobileNetV2 pre-trained models to train on the data set with a large scale image classification task. To instantiate, the convolutional base is free zed using transfer learning for the purpose of feature

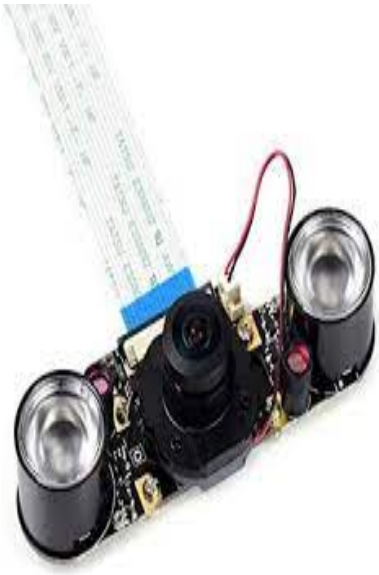

Fig. 16. Jetson Nano Camera

extraction and then new layers are added to turn the features into predictions for training the dataset to convergence. Soon after, fine tuning is performed which unfreezes the entire model with a low learning rate of 0.0001 to avoid the risk of over fitting. The Batch Normalization layers have 2 non-trainable weights which are updated during training. In the next step step, we use the model compression techniques such as pruning post-training quantization to achieve a reduced model size improved speed-up without affecting the obtained accuracy after fine tuning for real-time inference optimization. We initially start pruning the model with 50% sparsity then end with 80% sparsity for the VGG16, ResNet50 models. For the MobileNetV2 model, there's a width multiplier parameter that helps in trade-off accuracy along with the parameters the computational cost, since they're small with low-latency and low-power.

After pruning quantization, the approximate model size of the VGG16 is compressed down from 510 MB to 129 MB with an increase in GPU performance from 29.33 ms to 7.01 ms. Similarly, the approximate model size of the ResNet50 is lowered down from 130 MB to 36.9 MB with an increase in speed-up GPU performance to nearly 6.8 ms. Lastly, the approximate model size of the MobileNetV2 is reduced down from 3.67 MB to 0.76 MB with an increase in speed-up GPU performance to 5.73 ms. The accuracies obtained for VGG16, ResNet50 MobileNetv2 are 91.2,, 94.06, and 93.91 percents respectively after fine tuning and 91.65, 94.2, and 93.76 percents respectively after applying the model compression techniques.

So now, upon configuring customizing the Mask R-CNN which is an extension of the Faster R-CNN model, we prepare the model configuration parameters for object detetction, build the architecture with 5 layers and then load the weights. We use load_dataset() function to accept the directory of the images in the dataset, load_mask() to return the masks and the class ID's of each object, and then extract_boxes() to return the coordinates of each bounding box including the height weight of each image. Then after transfer learning and fine tuning, the accuracy obtained for this model is 98.64 percent. Furthermore, we prune quantize it for a reduced model size without much loss in accuracy, where the original model size sums up to around 4.2 MB, which is eventually compressed to 0.58 MB with an accuracy of 98.39 percent with a GPU speed of 4.84 ms.

So in order to deploy our model on the Jetson Nano, we first run inference on the model where the inference tool helps the Mask R-CNN model to visualize bounding boxes frame-by-frame. Exporting the model separates the training and inference processes and enables translation to TensorRT engines outside of the TLT environment since TensorRT engines are hardware-specific and should be created for each particular inference environment. The exported model format has the extension .etlt now. TensorRT engines can be created in INT8 mode to boost speed, however this necessitates the building of a calibration cache. Pre-generating and storing calibration information eliminates the requirement to calibrate the model on the inference machine. Because it is a considerably smaller file and can be moved with the exported model, relocating the calibration cache is typically far more convenient than moving the calibration tensor file. Using the calibration cache further accelerates engine development because constructing the cache might take several minutes depending on the size of the Tensor file and the model itself. Additionally, we use the Deep Stream SDK streaming analytic toolkit to accelerate the deep learning based models. [1]We employ the TensorRT open source software (OSS) with generate Detection Plugin, multilevel Crop And Resize Plugin, resize Nearest Plugin and multi level Propose ROI plugins from the TensorRT OSS Build. For Jetson Nano, we update the TensorRT OSS using the following lines of code:

```
o sudo apt remove --purge --auto-remove
   cmake
o wget https://github.com/Kitware/CMake
   /releases/download/v3.13.5
   /cmake-3.13.5.tar.gz
o tar xvf cmake-3.13.5.tar.gz
o cd cmake-3.13.5/
o ./configure
o make -j$(nproc)
o sudo make install
o sudo ln -s /usr/local/bin/cmake
    /usr/bin/cmake
o git clone -b 21.03
    https://github.com/nvidia/TensorRT
o cd TensorRT/
o git submodule update --init --recursive
o export TRT_SOURCE=`pwd`
o cd $TRT_SOURCE
```

```
o mkdir -p build && cd build
o sudo mv /usr/lib/aarch64-linux-gnu
  /libnvinfer_plugin.so.7.x.y ${HOME}
  /libnvinfer_plugin.so.7.x.y.bak//backup
  original libnvinfer_plugin.so.x.y
o sudo cp `pwd`/out/
    libnvinfer_plugin.so.7.m.n
    /usr/lib/aarch64-linux-gnu
   /libnvinfer_plugin.so.7.x.y
o sudo ld config
```

Next, in order to generate the tlt-converter for the further step and then to use it, we use
d the following lines of code:

```
o $ export TRT_INC_PATH="/usr/include
   /aarch64-linux-gnu"
o tlt-converter [-h] -k <encryption_key>
o -d <input_dimensions>
o -o <comma separated output nodes>
o  [-c <path to calibration cache file>]
o  [-e <path to output engine>]
o  [-b <calibration batch size>]
o  [-m <maximum batch size of the
   TRT engine>]
o  [-t <engine datatype>]
o  [-w <maximum workspace size of
    the TRT Engine>]
o  [-i <input dimension ordering>]
o  [-p <optimization_profiles>]
o  [-s]
o  [-u <DLA_core>]
o  input_file
```

Hence, we generate a device-specific Jetson Nano optimized TensorRT engine using the tlt-converter for direct integration of the model with the DeepStream. Once the model is deployed onto the hardware set-up, we capture the image with a fire background to detect it using our software, and then if the infrared camera senses the problem at task, the Jetson Nano will make sure to send us a signal/indication through the buzzer unit.

VII. CONCLUSION

Augmented reality is a technology-based interaction approach that superimposes virtual elements on the actual environment. Henceforth, we developed an automated system capable of real-time, intelligent object identification and recognition, allowing firemen to increase their situational awareness during an emergency response in this project. We employ a trained Mask R-CNN system with an enhancement to the Faster R-CNN. By collecting, processing, and evaluating key information, this system is capable of effectively guiding the decision-making process of firefighters in the midst of the critical circumstances generated by an accidental fire incident. So, in comparison with the state-of-the-art pre-trained models like VGG 16, ResNet 50 the MobileNet V2 which provided us with an accuracy of 91.65%, 94.2%, and 93.76% respectively, we obtain a final pruned and d accuracy of 98.39% with our customized Mask R-CNN model which is an extension of the Faster R-CNN network structure, which uses instance segmentation for a sharper focus on an individual targets of the same class through object localization. The experimental results also validate the effectiveness of our model, which achieved a significant speedup of almost 6x times over the base models and a model compression rate of 7.2x over its original size after compaction through the commonly used pruning post-training quantization rates, as compared to conventional computations. Also, as compared to the VGG16, ResNet50 the MobileNetV2 which achieve a model compression rate of 3.9x, 3.5x, and 4.82x with a speed-up approximately equivalent to 4.18x, 4.33x, 5.1x, we achieved a higher performance rate with an inter-varietal generalization.

REFERENCES

[1] K. Atanassov, P. Vassilev, I. Georgiev, and H. Tsakov. Generalized net model of the organization of the firefighting activities when dealing with forest fires. In *ANNA '18; Advances in Neural Networks and Applications 2018*, pages 1–5, 2018.

[2] F. Bellalouna. Industrial use cases for augmented reality application. In *2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 000011–000018, 2020.

[3] A. R. Calvão, F. Carvalho, and F. Marques. Decision support system for forest fires firefighting in agueda municipality. In *2015 10th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–5, 2015.

[4] R. H. Eltom, E. A. Hamood, A. A. Mohammed, and A. A. Osman. Early warning firefighting system using internet of things. In *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pages 1–7, 2018.

[5] M. Han and W. Baek. Herti: A reinforcement learning-augmented system for efficient real-time inference on heterogeneous embedded systems. In *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 90–102, 2021.

[6] M. Iqbal, C. Setianingsih, and B. Irawan. Deep learning algorithm for fire detection. In *2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, pages 237–242, 2020.

[7] T. Iwata, T. Yamabe, and T. Nakajima. Augmented reality go: Extending traditional game play with interactive self-learning support. In *2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, volume 1, pages 105–114, 2011.

[8] Z. Luo, Y. Zhang, L. Mu, J. Huang, J. Xin, H. Liu, S. Jiao, G. Xie, and Y. Yi. A uav path planning algorithm based on an improved d lite algorithm for forest firefighting. In *2020 Chinese Automation Congress (CAC)*, pages 4233–4237, 2020.

[9] D. Tate, L. Sibert, and T. King. Using virtual environments to train firefighters. *IEEE Computer Graphics and Applications*, 17(6):23–29, 1997.

[10] A. Tewari, B. Taetz, F. Grandidier, and D. Stricker. [poster] a probabilistic combination of cnn and rnn estimates for hand gesture based interaction in car. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pages 1–6, 2017.

[11] Z. Xiao, E. Dong, and S. Du. Fire detection method based on deep residual network and multi-scale feature fusion. In *2020 Chinese Automation Congress (CAC)*, pages 4810–4815, 2020.