IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 June 2023, accepted 22 August 2023, date of publication 28 August 2023, date of current version 5 September 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3309050*

APPLIED RESEARCH

# Optimizing Video Analytics Deployment for In-Flight Cabin Readiness Verification

**UNAI ELORDI[1,2], NEREA ARANJUELO[1,2], LUIS UNZUETA[1], JOSE LUIS APELLANIZ[1], AND IGNACIO ARGANDA-CARRERAS[2,3,4]**

[1]Fundación Vicomtech, Basque Research and Technology Alliance (BRTA), 20009 Donostia-San Sebastian, Spain
[2]Computer Science and Artificial Intelligence Department, University of the Basque Country (UPV/EHU), 20018 Donostia-San Sebastian, Spain
[3]Ikerbasque, Basque Foundation for Science, 48009 Bilbao, Spain
[4]Donostia International Physics Center (DIPC), 20018 Donostia-San Sebastian, Spain

Corresponding author: Unai Elordi (uelordi@vicomtech.org)

This work was supported in part by the Clean Sky 2 Joint Undertaking through the European Union's Horizon 2020 Research and Innovation Program under Grant 865162; in part by Smart Cabin System for cabin readiness COVID Amendment (SmaCS) (https://www.smacs.eu/), from the University of the Basque Country (UPV/EHU) under Grant GIU19/027; and in part by the Ministerio de Ciencia, Innovacion y Universidades, Agencia Estatal de Investigación (AEI), under Grant PID2021-126701OB-I00.

**ABSTRACT** This paper proposes an approach to optimize the deployment of on-board video analytics for checking the correct positioning of luggage in aircraft cabins. The system consists of embedded cameras installed on top of the cabin and a heterogeneous embedded processor. Each camera covers multiple regions of interest (i.e., multiple seats or aisle sections) to minimize the number of cameras required. Each image region is processed by a separate image classification algorithm trained with the expected kind of visual appearance considering the effect of perspective and lens distortion. They classify these regions as correct or incorrect for cabin readiness by exploiting the hierarchical structure of classes composed of different configurations of passengers' and objects' presence or absence and the objects' location. Our approach leverages semantic distances between classes to guide prototypical neural networks for multi-tasking between the main classification (i.e., correct or incorrect status) and auxiliary attributes (i.e., scene configurations), learning robust features from different data domains (i.e., various cabins, real or synthetic). The processing pipeline optimizes response delay and power consumption by leveraging embedded processors' computing capabilities. We carried out experiments in a cabin mockup with a Jetson AGX Xavier, efficiently obtaining better-quality descriptive information from the scene to improve the system's accuracy compared to alternative state-of-the-art methods.

**INDEX TERMS** Aircraft, computer vision, deep learning, optimal deployment, pattern recognition, video analytics.

## I. INTRODUCTION

Aircrafts require crew members to ensure that all safety rules are complied with and to attend to the needs of passengers. However, some of their tasks could be improved with the assistance of intelligent systems [1]. To that end, Unzueta et al. [2] presented a conceptual video analytics solution to alert crew members when passengers put luggage in places where safety could be compromised during critical flight phases such as taxi, takeoff and landing (TTL). In this system, embedded cameras are installed over the seats and the aisle, capturing images like those shown in Figure 1. Each camera should cover the maximum possible area, e.g., two seat-rows, to minimize the number of cameras and the cost. An embedded artificial intelligence (AI) processor should receive image streams from all the cameras, apply computer vision algorithms to determine whether the luggage is correctly placed, and send alerts to the crew members when required, specifying where they should manually check. This setup could also be used for other applications such

The associate editor coordinating the review of this manuscript and approving it for publication was Rosario Pecora.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

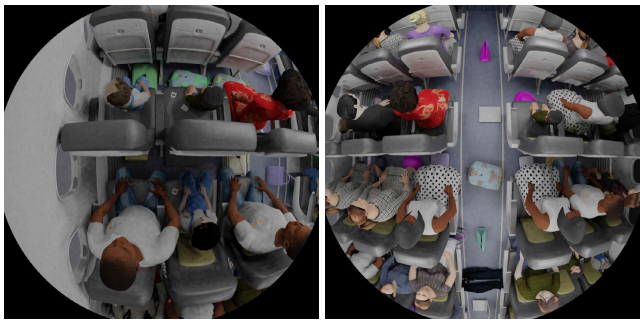


**FIGURE 1.** Examples of the kind of images captured from cameras installed over the seats and the aisle in a simulated virtual cabin.

**FIGURE 2.** Examples of image crops to be processed by multiple DNNs.

as checking the seat occupancy, whether tray tables are in stowed positions, dangerous behaviors of passengers, etc.

Currently, the most advanced computer vision algorithms rely on deep neural networks (DNNs) for tasks such as object detection [3], image classification [4], etc. Object detection DNNs can be used to obtain bounding boxes of objects and passengers in an image, as well as distinguish between different kinds of object classes. On the other hand, image classification DNNs can be applied to image regions containing one seat or aisle section to label the image content with learned concepts (Figure 2). This can help determine whether luggage positioning in that region is correct or not. The advantage of using image classification DNNs in this context is that training data can be labeled more easily, and that DNNs for classification are also much more efficient.

To obtain accurate classification results, samples from the same class should have similar visual appearances and be significantly different from the other class. However, as shown in Figure 2, this is not the case in our problem as there could be many different kinds of objects and people involved, with very varied appearances and spatial relations. Besides, there are objects whose presence is not a problem for TTL cabin readiness, such as magazines, books, food, smartphones, tablets, jackets, wallets, etc., which can be referred to as ''non-cabin luggage''. On the contrary, ''cabin luggage'' would include backpacks, bottles, briefcases, camcorders, hats, laptops, shopping bags, suitcases, tote bags, etc.

Thus, considering this kind of variability, these two classes could be divided into fine-grained subclasses grouped according to different visual appearances. Figure 3 shows

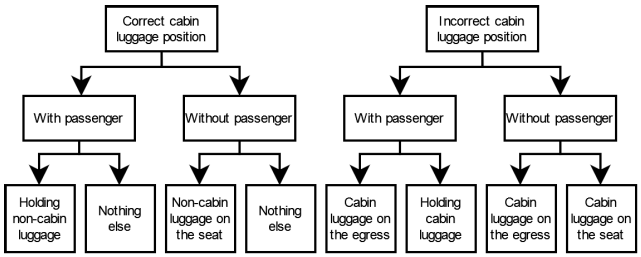



**FIGURE 3.** Example of subdivision in the positioning of luggage for TTL cabin readiness as a three-level hierarchy of classes.

an example of this subdivision for seats in standard seat rows of the aircraft, represented as a three-level hierarchy of classes. The top level contains the two-goal classes (correct or incorrect). The middle level considers the presence or absence of a passenger on the seat, and the lower level considers the absence or presence of objects, their type, and their positioning.

Going through the hierarchy from the top level to each class of the lower level yields more specific and richer descriptions of the correct and incorrect situations. A classification DNN could be trained by taking these lower-level subclasses independently. However, a better approach is to leverage this extra information to improve the accuracy and reliability of the system through learning paradigms such as multi-task learning [5], [6] and metric-guided prototype learning [7]. This work analyzes how this could be done in our context.

Another problem to be tackled in this system is that as the appearances of seats vary depending on their placement on the image—due to perspective and the image distortion—it would need multiple DNNs to analyze the whole scene. Each DNN would learn the specific kind of appearances expected in each image region. For that, how datasets are designed and used for training is another key factor to be considered, especially in this kind of context with visual specificities normally not present in generalist datasets, like ImageNet [8], COCO [9] or OpenImages [10]. For this purpose, we also present in this work the SmaCS dataset [11] to facilitate future research. Considering that in practice, data could usually come from different data sources (i.e., various cabins, real or synthetic), this approach considers domain adaptation techniques to enhance generalization performance [12].

Deploying a multi-DNN-based multi-camera system on an onboard embedded processor for aircraft cabin readiness verification, consuming minimal power, poses significant challenges in achieving accuracy, robustness, and responsiveness. This work addresses this challenging problem by making the following contributions:

1) An approach to deploy on-board video analytics for in-flight TTL cabin readiness verification with an optimal trade-off between response delay and power consumption.
2) Metric-guided multi-task domain-adversarial prototypical networks (MMDAPNs) for efficient and robust image classification, exploiting the hierarchical structure of classes.

3) An optimal multi-MMDAPN processing pipeline tailored to the embedded processor's heterogeneous computing capabilities.
4) Experimental results with the SmaCS dataset [11], comprising around 30K images captured in a cabin mockup and 7K generated with computer graphics depicting various situations involving passengers and objects.

The rest of the paper is organized as follows: Section II describes prior related work; Section III explains MMDAPNs and the optimal processing pipeline to check the correct positioning of luggage for aircraft TTL cabin readiness; Section IV presents experimental results with data obtained from a cabin mockup and processed with a Jetson AGX Xavier. Finally, Section V presents the conclusions and future lines of work.

## II. RELATED WORK

Better classification accuracies tend to be obtained by more complex DNN architectures that require significant computation resources and energy costs. As a result, there have been recent efforts to achieve efficient real-time deployment of DNNs in processors with limited computational resources. One way to do this is by improving the structure of DNNs to achieve better accuracy with fewer parameters. In the last decade, convolutional neural networks (CNNs) have become the dominant type of DNNs for visual object recognition. With CNNs, the exploration of different connectivity patterns has resurged, leading to notable network architecture innovations such as DenseNet [13], EfficientNet [14], and EfficientNetV2 [15].

DenseNets [13] use direct connections between any two layers with the same feature-map size to scale naturally to hundreds of layers while exhibiting no optimization difficulties that traditional CNNs have. EfficientNets [14] are a family of models optimized for floating-point operations per second (FLOPs) and parameter efficiency through a scaling method that uniformly scales all depth/width/resolution dimensions using a simple yet highly effective compound coefficient. EfficientNetV2 [15] improves the previous models' size and speed, thanks to the combination of training-aware neural architecture search and scaling. Vision transformers (ViTs) [16] are another type of DNNs that have recently received attention from the computer vision community as they have demonstrated superiority in accuracy over CNNs. However, they currently have higher computational costs and therefore require further research on optimization techniques to efficiently deploy them in resource-constrained processors [17].

Another way to improve the DNN's inference efficiency and size is through network compression techniques such as pruning and quantization, used independently or in combination [18]. Pruning removes redundant computations that have limited contribution to a result. Pruning can be performed element-wise, channel-wise, shape-wise, filter-wise, layer-wise, and even network-wise. Each has trade-offs in compression, accuracy, and speedup. On the contrary, quantization reduces computations by reducing the data type's precision (typically from 32-bit to 16 or 8-bit). It can significantly improve performance (usually 2-3x) and reduce storage requirements.

Moreover, to accelerate DNN inference, new kinds of embedded processors have been equipped with neural processing units (NPUs), apart from a CPU and a GPU. For example, NVIDIA's NVDLAs (NVIDIA Deep Learning Accelerators) included in Jetson AGX Xavier and Jetson Xavier NX computing platforms, or Intel's VPUs (Vision Processing Units) and Google's Coral Edge TPUs (Tensor Processing Units) that could be integrated into embedded PCs. Each of these processors requires a specific deep learning framework for optimal DNN inference: e.g., NVIDIA's TensorRT, Intel's OpenVINO, or Google's TensorFlow Lite for their respective processors.

Some companies also provide toolkits to build optimal end-to-end DNN-based streaming analytics pipelines. For example, NVIDIA's DeepStream and Intel's DL Streamer, both based on the open-source GStreamer multimedia framework, provide hardware-accelerated modules that encompass decode, preprocessing, and DNN inference of input video streams. They consider the pipeline for object detection, attribute classification, and tracking but also allow users to build more complex pipelines if required.

However, deploying multiple concurrent classification DNNs as in our case is challenging in current off-the-shelf DNN accelerators and deep learning frameworks as they are not designed for that. They only provide a single-level priority, one-DNN-per-process execution model, sequential inference interfaces, and assume that the DNN inference is executed on a single processing element (CPU, GPU or NPU), no more than one simultaneously.

Some works have focused on leveraging the heterogeneous computing capabilities of embedded processors. For instance, Xiang et al. [19] proposed DART, a CPU-GPU scheduling framework for DNNs that offers deterministic response time to real-time tasks and increased throughput to best-effort tasks. It employs a pipeline-based scheduling architecture with data parallelism, where heterogeneous CPUs and GPUs are arranged into nodes with different parallelism levels. Similarly, Lim et al. proposed ODMDEF [20], an on-device CPU-GPU co-scheduling framework to remove the performance barrier precluding DNN executions from being bound by the GPU. Experiments with the NVIDIA Jetson AGX Xavier platform show that it speeds up the execution time by up to 46.6% over the GPU-only solution. Jeong et al. [21] proposed a parallelization methodology to maximize the throughput of a single DNN-based application using GPU and NPU by exploiting various types of parallelism on TensorRT: multi-threading, multi-stream, pipelining of the inference network, and partial network duplication. They devised a heuristic to determine the pipeline cut-points achieving 81%-391% throughput improvement over the baseline inference that uses the GPU only in six real-life object detection

networks on an NVIDIA Jetson AGX Xavier board. The same authors also introduced a tool called JEDI (Jetson-aware Embedded Deep learning Inference acceleration) [22] for implementing the proposed optimizations on NVIDIA Jetson boards.

Other works have focused on optimizing the deployment of multiple DNNs in a single processing element. For example, Kim [23] presented a methodology to allow higher priority DNNs to occupy the GPU preferentially. Every decomposed DNN layer job is stacked in a priority order inside the layer queue. Higher priority layer jobs can preempt lower-priority ones using stream prioritization, reducing the execution time of DNNs by up to 60.4%. Cox et al. proposed MASA [24], a responsive memory-aware multi-DNN execution framework on CPU, an on-device middleware featuring modeling inter- and intra-network dependency and leveraging complementary memory usage of each layer. It can consistently ensure the average response time when deterministically and stochastically executing multiple DNNs. Finally, Yu et al. [25] proposed a graph- and runtime-level cross-layer scheduling framework for multi-tenant inference optimization in GPU, which automatically coordinates concurrent DNN computing at different execution levels. It achieves 1.3x 1.7x speedup compared to regular DNN runtime libraries (e.g., CuDNN, TVM) and concurrent scheduling methods (e.g., NVIDIA Multi-Stream).

## III. METHODOLOGY

### A. MMDAPN DESIGN AND TRAINING

DNNs rely on data to learn and make predictions. The accuracy and reliability of DNNs depend on how the data is handled during training, such as identifying commonalities between classes and the relationships between labels. Multi-task learning [5] aims to improve the learning performance of multiple related tasks by sharing valuable information between them. In our case, the primary task is to classify the top-level classes, while auxiliary tasks involve classifying the subclasses. However, identifying commonalities between different subclasses could make it challenging to distinguish between the main classes. Metric-guided prototype learning [7] can help address this issue by making DNNs focus on relevant image features to distinguish between classes based on semantic hierarchical priors.

Designing a system that uses DNNs requires careful consideration of the amount and quality of data available to train the model. To ensure better generalization, training datasets should be designed with care, gathering as many samples of each subclass from different real cabins to avoid bias among classes. Synthetic data can help with this process [26], [27], but it's important to handle the domain gap between different data sources (i.e., various cabins, real or synthetic). Domain adaptation methods [12] can help DNNs extract robust cross-domain features.

Our proposed MMDAPN architecture improves the classification accuracy and reliability of DNNs by combining multi-task, metric-guided prototype, and domain adversarial learning paradigms. Figure 4 shows the architecture of MMDAPNs used to train classification models that exploit the semantic information present in the hierarchy of classes for aircraft cabin readiness verification with data from different sources (e.g., synthetic and real cabins). The deployment of multiple MMDAPNs to analyze images like those shown in Figure 2 enables efficient on-board video analytics of in-flight TTL cabin readiness verification.

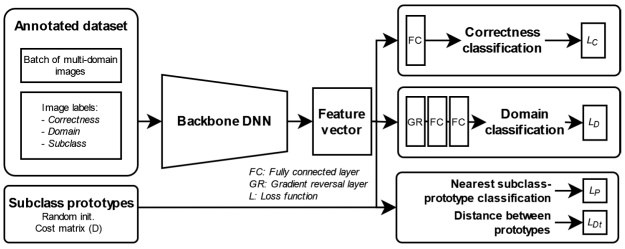



**FIGURE 4. The architecture of metric-guided multi-task domain adversarial prototypical networks (MMDAPNs).**

In our context, MMDAPNs are constructed around a backbone classification deep neural network (DNN). Given the limited computational resources of the onboard embedded processor, this approach considers efficient DNNs such as EfficientNetV2. During training, the MMDAPN architecture aims to optimize three tasks: (1) classifying the main top-level classes (i.e., correct or incorrect luggage positioning), (2) classifying the source domain (e.g., cabin 1, cabin 2, cabin 3, etc.), and (3) classifying the fine-grained scene descriptive subclasses (Figure 3).

The input data comes from two sources: (1) an annotated dataset of images gathered from various cabins (real or synthetic), and (2) subclass prototypes generated by the user-defined semantic hierarchical priors that guide the learning process. The former is introduced batch by batch in each iteration of the training process, equally distributing the data from each data source (i.e., cabin 1, cabin 2, etc.). Data samples include labels for three tasks: luggage positioning correctness (0 or 1), domain ID (integer value), and scene descriptive subclass ID (integer value). The latter uses a finite metric on the hierarchical class set to supervise a prototypical network that associates each class with a representation or prototype and classifies observations according to the nearest prototype [7]. More specifically, the metric is defined by semantic distances among subclasses under the form of a cost matrix that acts as a distortion-based regularizer for the prototypical network.

Eq. (1) shows the loss function of MMPADNs. $\mathcal{L}_C$ is a binary cross-entropy loss for the luggage positioning correctness classification, $\mathcal{L}_D$ is a binary or softmax cross-entropy loss for the domain classification (depending on whether there are two or more data source domains), $\mathcal{L}_P$ is the loss for the subclass-prototype classification and $\mathcal{L}_{Dt}$ the loss for the metric-distortion-based regularization. The loss weights $\lambda_C$, $\lambda_D$, $\lambda_P$ and $\lambda_{Dt}$ are floating-point values between 0 and 1,

empirically selected to balance the contribution of each loss.

$$\mathcal{L} = \lambda_C \mathcal{L}_C + \lambda_D \mathcal{L}_D + \lambda_P \mathcal{L}_P + \lambda_{Dt} \mathcal{L}_{Dt} \quad (1)$$

By including $\mathcal{L}_D$ in the loss function, the MMPADN learns how to distinguish each domain during training and adjusts the backbone DNN's weights to maximize the outcome of the rest of the learning tasks in all domains. This helps the DNN learn the most robust features for all domains simultaneously.

This work relies on the approach proposed in [7] for the definitions of $\mathcal{L}_P$ and $\mathcal{L}_{Dt}$. The proposed training dataset considers $\mathcal{N}$ of $N$ elements $x \in \mathcal{X}^N$ with ground truth classes $z \in \mathcal{K}^N$, hierarchically organized. A user-defined cost matrix $D$ defines a finite metric that considers the shortest path between nodes of the hierarchical $\mathcal{K}$ classes. $\Omega$ is the embedding space that forms a continuous metric space when equipped with the distance function $d : \Omega \times \Omega \mapsto \mathbb{R}_+$. Eq. (2) shows an example of $D$ for the class hierarchy example of Figure 3, where rows and columns follow the same order as the lower-level subclasses, i.e., 0: *Correct_with_passenger_holding_non-cabin_luggage*, 1: *Correct_with_passenger_nothing_else*, 2: *Correct_no_passenger_non-cabin_luggage_on_the_seat*, etc.

$$D = \begin{bmatrix} 0 & 2 & 4 & 4 & 6 & 6 & 6 & 6 \\ 2 & 0 & 2 & 4 & 6 & 6 & 6 & 6 \\ 4 & 2 & 0 & 2 & 6 & 6 & 6 & 6 \\ 4 & 4 & 2 & 0 & 6 & 6 & 6 & 6 \\ 6 & 6 & 6 & 6 & 0 & 2 & 4 & 4 \\ 6 & 6 & 6 & 6 & 2 & 0 & 2 & 4 \\ 6 & 6 & 6 & 6 & 4 & 2 & 0 & 2 \\ 6 & 6 & 6 & 6 & 4 & 4 & 2 & 0 \end{bmatrix} \quad (2)$$

In this example, there are four semantic distance values: 0 for the lower-level subclass with respect to itself, 2 for lower-level subclasses of the same top-level and mid-level groups, 4 for subclasses of the same top-level group and different mid-level class, and 6 for subclasses of different top-level groups.

Following this nomenclature, a prototypical network is characterized by an embedding function $f : \mathcal{X} \to \Omega$ and a set $\pi \in \Omega^{\mathcal{K}}$ of $K$ prototypes. Based on [28], a prototypical network associates with an observation $x_n$ the posterior probability over its class $z_n$, as shown in Eq. (3).

$$p(z_n = k|x_n) = \frac{\exp(-d(f(x_n), \pi_k))}{\sum_{l \in \mathcal{K}} \exp(-d(f(x_n), \pi_l))}, \forall k \in \mathcal{K} \quad (3)$$

Thus, $\mathcal{L}_P$ is defined as the normalized negative log-likelihood of the true classes, as shown in Eq. (4). This loss encourages the embedding function $f(x_n)$ to be close to the prototype $\pi_{z_n}$ and far from the other prototypes.

$$\mathcal{L}_{\mathcal{P}}(f, \pi) = \frac{1}{N} \sum_{n \in \mathcal{N}} \left( d(f(x_n), \pi_{z_n}) + \log \left( \sum_{l \in \mathcal{K}} \exp(-d(f(x_n), \pi_l)) \right) \right) \quad (4)$$

Finally, $\mathcal{L}_{D_t}$ is defined as shown in Eq. (5). This loss enforces a metric-consistent prototype arrangement to avoid conflicting with the second term of $\mathcal{L}_P$ by scaling prototype coordinates in $\Omega$ by a scalar factor $s$. Thus, it encourages a low distortion between $s \cdot \pi$ scaled prototypes and $D$.

$$\mathcal{L}_{Dt}(\pi) = \frac{1}{K(K-1)} \min_{s \in \mathbb{R}_+} \sum_{k,l \in \mathcal{K}^2, k \neq l} \left( \frac{sd(\pi_k, \pi_l) - D[k,l]}{D[k,l]} \right)^2 \quad (5)$$

### B. OPTIMAL MULTI-MMDAPN PROCESSING PIPELINE

The installation's characteristics are determined by the cabin areas to be covered and the minimum cabin luggage size. These characteristics include the number of cameras, their placement, their lens characteristics, the number of image regions of interest (ROIs) analyzed (Figure 2), and their resolution. However, deploying the multi-MMDAPN-based approach optimally in a heterogeneous embedded processor onboard that satisfies the onboard response latency and power consumption constraints is not straightforward.

The proposed end-to-end processing pipeline for this purpose is shown in Figure 5 and is composed of four modules: (1) multithreaded image capture, (2) batched image preprocessing, (3) multi-MMPADN inference, and (4) response post-processing. The first module produces a full-image frame pool in the GPU by capturing images from $n$ cameras scheduled by $n$ CPU threads and decoding them in the GPU. The second module, composed of $x$ preprocessing workers, consumes $x$ batches of $k$ images from this frame pool to produce a pool of batched cropped images corresponding to $p$ ROIs that are padded and resized to match the MMPADN's input resolution in GPU. Then, the third module, composed of $y$ multi-MMPADN inference workers in GPU-NPU, consumes y multi-ROI batches from this pool and delivers the classification results to the fourth module that maps them in CPU to cabin placements to be shown in the monitoring system interface.

The values of $k$, $p$, $x$, and $y$ must be carefully selected, as well as the XPU (typically, GPU and/or NPU) where the MMPADN models will be deployed for inference to obtain the optimal trade-off between latency and power consumption for a targeted embedded processor that complies with onboard constraints. To help achieve this goal, this approach proposes the algorithm 1.

Algorithm 1 evaluates the performance of various MMPADN-XPU deployment configurations **c** using a testing dataset **I** of $n$ videos processed by **m** MMDAPNs. The algorithm measures the average latency $\Delta_{\text{avg}}$ and maximum power consumption $P_{\text{max}}$ for each deployment candidate configuration. It then selects all the configurations that meet the maximum acceptable $P_{\text{thr}}$ and $\Delta_{\text{thr}}$ values. The optimal deployment configuration is selected based on the minimum Euclidean distance $S_{\text{min}}$ between the latency and power values and the origin (0,0) on the latency-power plane.

## IV. EXPERIMENTS

To evaluate the efficacy of the proposed approach, we performed a series of experiments utilizing the SmaCS dataset [11]. The MMDAPN classification accuracy and

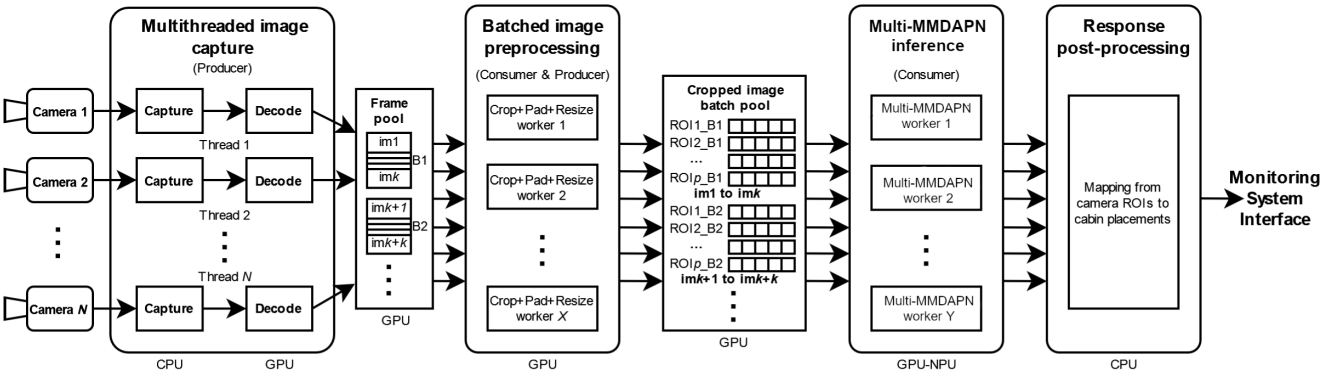



**FIGURE 5.** Multi-camera multi-MMDAPN-based end-to-end processing pipeline.

**Algorithm 1** Multi-MMDAPN Processing Pipeline Optimization for the Multi-Camera Onboard System

**procedure** PipeOpt($m, I, c, k_{\max}, x_{\max}, y_{\max}, P_{thr}, \Delta_{thr}$)
  $c_{\text{opt}} = \text{None}$
  $k_{\text{opt}} = x_{\text{opt}} = y_{\text{opt}} = -1$
  $S_{\min} = \text{BigNumber}$
  **for** $c$ in **c do**
    **for** $k \leftarrow 1$ to $k_{\max}$ **do**
      **for** $x \leftarrow 1$ to $x_{\max}$ **do**
        **for** $y \leftarrow 1$ to $y_{\max}$ **do**
          $P_{\max}, \Delta_{\text{avg}} \leftarrow \text{eval}(m, I, c, k, x, y)$
          **if** $P_{\max} < P_{\text{thr}} \& \Delta_{\text{avg}} < \Delta_{\text{thr}}$ **then**
            $S \leftarrow d_{\min}(P_{\max}, \Delta_{\text{avg}})$
            **if** $S < S_{\min}$ **then**
              $c_{opt} = c$
              $k_{opt} = k$
              $x_{opt} = x$
              $y_{opt} = y$
            **end if**
          **end if**
        **end for**
      **end for**
    **end for**
  **end for**
  **return** $c_{\text{opt}}, k_{\text{opt}}, x_{\text{opt}}, y_{\text{opt}}$
**end procedure**

reliability experiment demonstrates how its components enhance classification accuracy compared to state-of-the-art alternatives. Furthermore, this experiment aims to show the impact of employing different learning paradigms separately on classification accuracy and reliability. Finally, the optimal processing pipeline experiment evaluates the potential of the proposed deployment approach with an optimal trade-off between response delay and power consumption qualitatively and quantitatively.

### A. THE SmaCS DATASET

The SmaCS dataset required the construction of a cabin mockup to gather the necessary real data and 3D graphic

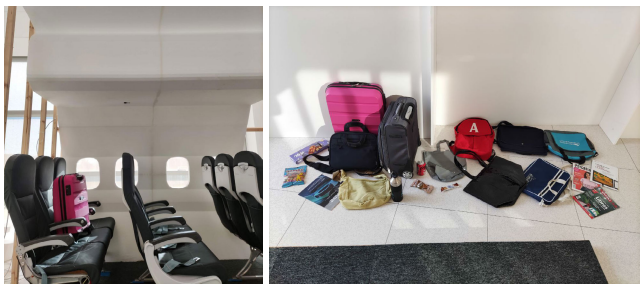


**FIGURE 6.** The cabin mockup and objects used to create the SmaCS dataset.

engine for synthetic data generation. The mockup comprises one side of three rows of cabin seats, the aisle, and the exterior of the cabin covered in polystyrene (see Figure 6). It is illuminated by three different light sources: (1) natural light entering through the adjacent windows, (2) artificial overhead lighting within the room, and (3) a spotlight located near the cabin window to simulate directional sunlight. The data capture process took place on different days, deliberately altering the lighting conditions. To monitor two rows of seats, two cameras were positioned above the four seats closest to the window, and another camera above the aisle to observe the aisle and the two adjacent seats.

This dataset established a recording protocol that was adhered to by 18 participants in order to simulate various scenarios involving cabin and non-cabin luggage objects (see Figure 6). The protocol outlined guidelines for the participants, including the duration of their seating in specific seats and the appropriate placement of any carried objects. The aim of the recording protocol was to recreate a diverse range of situations, encompassing both correct and incorrect luggage placement. This approach ensured a balanced representation of scenarios within the dataset, such as correctly and incorrectly placed luggage.

In contrast, the synthetic data was generated using the methodology outlined in [26], employing 22 distinct object types to replicate common cabin and non-cabin luggage items, as well as a cabin model that accurately represents a Boeing 737 aircraft. Additionally, we incorporated

a collection of human models exhibiting various poses and appearances to represent the seated passengers. To enhance variability, the appearance of the cabin and objects was randomized during the generation process.

Our previous publication [11] released the data for one specific seat–the one nearest to the cabin window in the rear row. This seat yielded approximately 30, 000 real samples and 7, 000 synthetic samples. Figure 7 showcases a selection of real and synthetic ROI images. Additionally, Figure 8 illustrates the distribution of the collected data across the defined subclasses. The number of samples per subclass in the synthetic data is balanced, whereas this balance is not observed in the real data. This discrepancy arises because, despite participants following the recording protocol, capturing ''empty seat'' samples (subclass 3) during the intermediate periods between different scenarios is unavoidable.

### B. ACCURACY AND RELIABILITY EVALUATION

To test the accuracy and reliability of the proposed MMDAPN, we separated a subset of approximately 2, 700 images for testing (real samples) and used the rest of them (real and synthetic) to train the model. The testing images do not contain visually similar situations (e.g., the same person with the same object and illumination) as it happens in the training data.

The backbone of the MMDAPN network is EfficientNet-V2-B0 [15] with pre-trained weights from the ImageNet dataset [8]. The resolution of the input images is $300 \times 300$ pixels, as it is the minimum size that allows visualizing small objects for classification, and empirically set the values of $\lambda_C = 0.8$, $\lambda_D = 0.6$, $\lambda_P = 1$, $\lambda_{Dt} = 1$. We used $D$ as shown in Eq. 2.

Table 1 presents a comparison of MMDAPN's performance on the test set against nine state-of-the-art alternatives. These alternatives include: (1) an image classifier based on EfficientNetV2-B0 [15], trained solely on overall correctness classes, (2) DANN with EfficientNetV2-B0 as the backbone [27], trained in a similar manner, (3) EfficientNetV2-B0 trained using subclasses and inferring overall correctness from them, (4) DANN with subclasses, (5) EfficientNetV2-B0 with multi-tasking, where the main task is overall correctness and subclasses serve as auxiliary tasks [6], (6) DANN with multi-tasking, (7) EfficientNetV2-B0 with prototypes of subclasses [7], used for inferring overall correctness, and (8) DANN with prototypes, following the same approach as (7), and (9) EfficientNetV2-B0 with multi-tasking and prototypes of subclasses.

The results reveal that all methods achieve perfect detection accuracy for an empty seat (subclass 3), with a 100% accuracy rate across all cases. However, the most challenging scenarios arise in subclass 0 (passenger holding non-cabin luggage), reaching a maximum accuracy of 79%, and subclass 4 (cabin luggage on the egress with a passenger), reaching a maximum accuracy of 77%. This outcome is to be expected since an empty seat provides visual consistency, whereas the presence of a passenger introduces increased visual variability. Additionally, distinguishing between non-cabin and cabin luggage can be difficult, particularly when the luggage is partially occluded by passengers. Similarly, detecting luggage on the egress can present challenges for the same reason.

Among all the methods considered, MMDAPN achieves the highest accuracy in overall correctness classification, achieving a remarkable accuracy of 96.9%. This result surpasses the second-best method (DANN + multi-task) by 1.9% and outperforms the least accurate method (EfficientNetV2) by a substantial margin of 11.61%. Furthermore, MMDAPN also achieves the highest scores in four out of the subclasses, which is the highest number of top rankings among all the methods. However, it is worth emphasizing that the most significant outcome in this particular use case is the overall correctness classification. The closest competitor to MMDAPN is DANN with multi-tasking, but MMDAPN consistently yields superior results across all categories except for three subclasses (subclasses 4, 5, and 6), where multi-task DANN achieves marginally better performance.

The results further demonstrate that incorporating both the multi-task approach, which integrates subclass information, and the matrix-guided prototypical component, which influences the positioning of subclasses in the feature space, leads to improved classification accuracy. These findings conclude that the MMDAPN architecture effectively classifies whether cabin luggage is correctly or incorrectly placed on the monitored seat.

To observe the impact of the metric-guided prototypical network component on the arrangement of features in the feature space, the features were extracted from the training images and employed principal component analysis (PCA) for dimensionality reduction, aiming to visualize the results. Figure 9 presents the extracted features using MMDAPN with and without the metric-guided prototypical network component.

In the case of training without the prototypical metric guidance (Figure 9, left), the features of different subclasses appear more entangled in the space, as exemplified by subclasses 4, 5, and 7. Conversely, when the metric-guided prototypical network component is incorporated (Figure 9, right), the features of each subclass exhibit clearer separation, and subclasses that are more similar to each other, such as subclass 6 and 7, are situated closer in the feature space. Consequently, in situations where misclassification occurs, the outcomes would be more meaningful, and the system would be more reliable.

### C. OPTIMAL PROCESSING PIPELINE EVALUATION

Both qualitative and quantitative evaluations assess the effectiveness of the optimal processing pipeline. The qualitative evaluation involved comparing various features of our approach with state-of-the-art alternatives. On the other hand, the quantitative evaluation presents experimental results of deploying multi-MMDAPN using a practical example on an

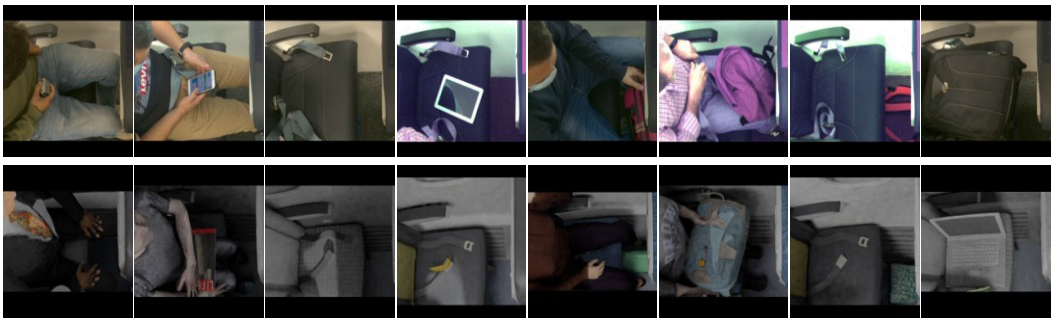


**FIGURE 7.** Examples of image ROIs from captured images (top) and synthetic images (bottom) for the different fine-grained subclasses.

**TABLE 1.** Comparison of MMDAPN on the test set with state-of-the-art alternatives (Sc: subclass).

| | Correct cabin luggage position | | | | Incorrect cabin luggage position | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | **Sc0** | **Sc1** | **Sc2** | **Sc3** | **Sc4** | **Sc5** | **Sc6** | **Sc7** | **Correctness** |
| EfficientNetV2 [15] | - | - | - | - | - | - | - | - | 85.29% |
| EfficientNetV2 [15] subclasses | 76.09% | **100%** | 94.93% | **100%** | 73.19% | 66.67% | **100%** | 72.46% | 89.02% |
| EfficientNetV2 [15] + multi-task [6] | 77% | 96% | **100%** | **100%** | 74% | 65% | **100%** | **100%** | 92% |
| EfficientNetV2 [15] + prototypes [7] | 68% | 97% | 54% | **100%** | 45% | 66% | **100%** | **100%** | 88% |
| EfficientNetV2 [15] + multi-task [6] + prototypes [7] | 76% | 99% | 96% | **100%** | 57% | 66% | **100%** | **100%** | 90% |
| DANN [27] | - | - | - | - | - | - | - | - | 94.4% |
| DANN [27] subclasses | **79%** | 94% | 96% | **100%** | **77%** | 70% | **100%** | 99% | 91% |
| DANN [27] + multi-task [6] | 60.7% | 99.7% | 97.9% | **100%** | 76.3% | **100%** | **100%** | 99.3% | 95.0% |
| DANN [27] + prototypes [7] | 62% | **100%** | 99% | **100%** | 54% | 79% | **100%** | 99% | 90% |
| MMDAPN (ours) | 65.4% | **100%** | **100%** | **100%** | 75.2% | 97.7% | 99.0% | **100%** | **96.9%** |

**TABLE 2.** Qualitative comparison of our processing pipeline with respect to alternative state-of-the-art approaches.

| Method | DNN types | DNN workload type | Optimization criteria | XPUs involved | Deployment config. |
|---|---|---|---|---|---|
| [19] | Classifiers | Variable | $\Delta_{\text{min}}$ | CPU-GPU | Manual |
| [20] | Classifiers | Variable | $\Delta_{\text{min}}$ | CPU-GPU | Manual |
| [21] [22] | Detectors | Constant | Max throughput | GPU-NPU | Manual |
| [23] | Classifiers | Variable | $\Delta_{\text{min}}$ | GPU | Manual |
| [24] | Detectors and classifiers | Variable | $\Delta$ and RAM trade-off | CPU | Manual |
| [25] | Classifiers | Variable | $\Delta_{\text{min}}$ | GPU | Manual |
| Ours | Classifiers | Constant | $\Delta$ and $P$ trade-off | CPU-GPU-NPU | Automatic |

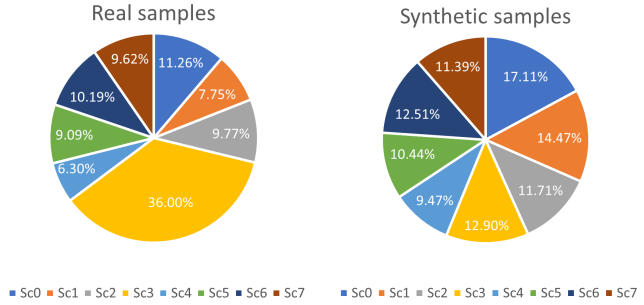



**FIGURE 8.** Distribution of the eight fine-grained subclasses in the captured and generated samples. Subclasses with index from 0 to 3 represent correct situations and from 4 to 7 incorrect ones (cabin luggage incorrectly placed).

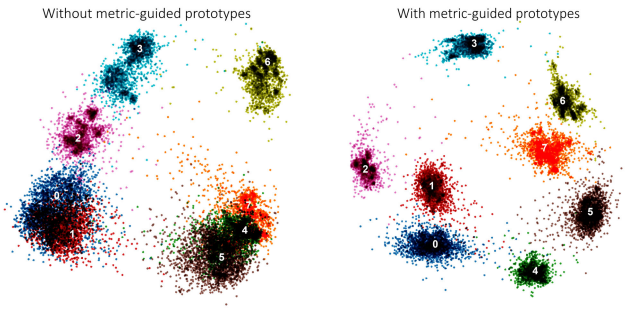



**FIGURE 9.** Image features visualization after PCA. Each subclass is represented using a different color. Features are extracted using the trained MMDAPN (right) and the model without the metric-guided prototypical network component (left).

NVIDIA Jetson AGX Xavier platform. Table 2 summarizes the qualitative comparison across five different categories.

As described previously, our approach focuses on constantly operating classification DNNs. While other works may employ detection DNNs or a combination of both types with variable workloads, our multi-DNN deployment aims to handle different situations specific to our use case. Unlike deployment methodologies that primarily prioritize inference speed and memory management, our deployment strategy also takes into account power consumption, optimizing the processing pipeline across the CPU-GPU-NPU subsystems. As illustrated in Figure 5, the CPU is responsible for

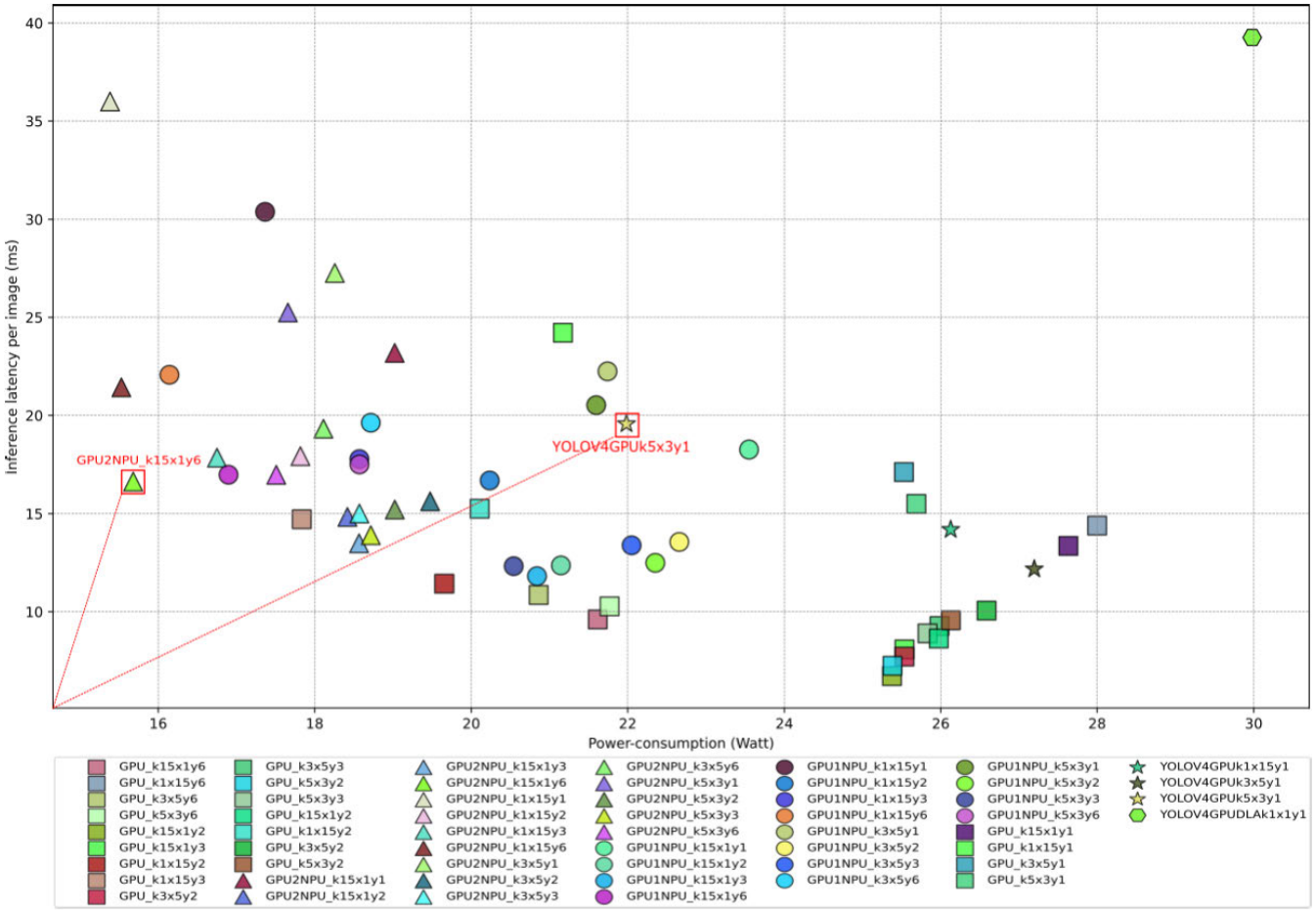



**FIGURE 10.** The Multi-MMDAPN processing pipeline evaluation to find the optimal deployment configuration with Jetson Xavier AGX 32GB. The optimal configuration is selected in red rectangle.

post-processing and capture tasks, while the preprocessing and multi-DNN inference tasks are offloaded to GPU and NPU hardware accelerators. Considering that the embedded device installed in the aircraft must adhere to low-power specifications, our approach strikes a balance between latency and power consumption measurements. Additionally, in contrast to state-of-the-art alternatives, which require manual configuration of deployment parameters, our approach automatically calculates the optimal configuration parameters for the processing pipeline.

For the quantitative evaluation, JEDI [22] is selected from the methods listed in Table 2 as it shares the same DNN workload type as our approach. However, it is important to note that while JEDI uses DNNs for detection (specifically, YOLOv4 [29]), our approach focuses on classification. This distinction implies that JEDI may require more computational resources compared to our pipeline. Furthermore, both our pipeline and JEDI employ different configurations that influence the trade-off between latency and power consumption. As a result, it becomes necessary to evaluate which approach yields the best results when considering their respective configurations.

In order to compare different deployment configurations, this experiment measures the latency and power consumption using algorithm 1 with the following configuration parameters: $k_{\max} = 15$, $x_{\max} = 15$, $y_{\max} = 6$, $P_{\text{thr}} = 30$W, and $\Delta_{\text{thr}} = 40$ms. Different batch sizes are considered, specifically $k = 1, 3, 5, 15$, $x = 1, 3, 5, 15$, and $y = 1, 2, 3, 6$. The formula $(T_l - T_f)/\text{num\_processed\_frames}$ calculates the average latency per image $\Delta_{\text{avg}}$, where $T_l$ represents the total processing time and $T_f$ denotes the time spent on the framework. Power consumption measurements were taken every second to prevent system saturation. JEDI is evaluated using their publicly available implementation [30]. The methodology employed to measure latency and power consumption was consistent for both our approach and JEDI.

The deployment software used for the NVIDIA Jetson Xavier AGX was JetPack 4.6.2, which incorporated TensorRT 8.2.1 and CUDA 10.2. The system was configured for MAXN power mode, and the Jetson clocks were activated. Our approach employed 15 full-HD cameras as input sources. Specifically, our approach used six ROIs, each trained with an MMDAPN model. This resulted in a total of six MMDAPNs, with EfficientNetV2-B0 [15] serving as the backbone.

The cropped images were resized to dimensions of 300 × 300 pixels. For the purpose of comparison with JEDI, this experiment selected their optimized implementation of YOLOv4 [29] with an input size of 416×416 pixels from their publicly available object detection implementations [30]. The Jetson Xavier AGX is equipped with a CPU, a GPU, and two NPUs (known as NVDLAs). This experiment quantized both the MMDAPN models and YOLOv4 models to FP16 precision, as it offers improved latency without significant accuracy loss. While INT8 quantization is also possible through a calibration process, it can result in a notable decrease in accuracy due to the reduced numeric range compared to FP16 or FP32.

Figure 10 depicts the optimal pipeline configurations for our approach and JEDI. Only candidates that fell within the limits of $\Delta_{\text{thr}}$ and $P_{\text{thr}}$ were considered. Each candidate's hardware deployment configuration is represented by different shapes. For our approach, rectangles represent that all DNNs run on the GPU, triangles represent 4 DNNs on the GPU and 2 on the 2 NPUs (one on each), and circles indicate 5 DNNs on the GPU and 1 on a single NPU. Stars symbolize JEDI's optimized YOLOv4 for GPU configurations $k = 1, 3, 5, 15$ and $x = 1, 3, 5, 15$ that fall within the $\Delta_{\text{thr}}$ and $P_{\text{thr}}$ limits. Since JEDI does not support multi-DNN execution, $y = 1$ was defined for DNN threading. The hexagon denotes JEDI's best YOLOv4 configuration, utilizing 2 NVDLAs and GPU hardware with the PND-A technique and a (82) cutpoint, employing a model configuration of $k = 1$, $x = 1$, and $y = 1$. The graph presents the latency in milliseconds and power consumption in Watts.

The experiment demonstrates that our method's optimal configuration (GPU2NPU_k15 × 1y6) outperforms JEDI's (YOLOv4GPU_k5 × 3y1): ∼16ms/∼15W compared to JEDI's ∼20ms/∼22W. Our method's optimal configuration uses the GPU and 2 NPUs for multi-DNN inference with $k = 15$, $x = 1$, and $y = 6$. This suggests that loading 15 frames to preprocess all crops in the same thread ($k = 15$, $x = 1$), but executing DNN inference in a separate thread, is more efficient ($y = 6$). Notably, 80% of the top 10 candidates employ a GPU-NPU hardware combination (8 out of 10). The experiment also reveals that using two threads for DNN ($y = 2$) yields the fastest latency results with GPU hardware configuration (GPU_k15 × 1y2). However, using $y = 3$ results in the worst performance across all GPU configurations, with higher latencies and power consumption. JEDI's YOLOv4 candidates for GPU configurations indicate that employing multiple preprocessing threads leads to faster inference results for $x = 5$ and $x = 15$, surpassing the MMDAPN approach's performance with a single thread for DNN ($y = 1$).

## V. CONCLUSION

This work presents an approach for verifying the proper placement of luggage in aircraft cabins by deploying on-board video analytics with an optimal balance between response delay and power consumption. Our approach combines metric-guided multi-task domain adversarial prototypical Networks (MMDAPNs) with an optimized processing pipeline that takes advantage of the heterogeneous computing capabilities of embedded processors. To assess the effectiveness of our approach, this work conducted experiments using the SmaCS dataset [11], which was specifically created for research in this domain.

Experimental results demonstrate that the MMDAPN architecture significantly enhances the classification accuracy of overall luggage positioning correctness when compared to existing state-of-the-art alternatives. This notable improvement can be attributed to the incorporation of a subclasses hierarchy, along with the matrix-guided prototypical component, which effectively influences the arrangement of subclasses within the feature space.

In contrast to alternative state-of-the-art multi-DNN processing pipelines, our approach harnesses the heterogeneous computing capabilities available and effectively determines the optimal configuration that strikes a balance between processing latency and power consumption. To evaluate these capabilities, this approach conducted an end-to-end deployment of the multi-MMDAPN system and compared it to an equivalent state-of-the-art approach for object detection [30] on the Jetson AGX platform. Our findings indicate that employing higher batch sizes and increasing the number of worker DNN threads contribute to more efficient pipeline processing in both scenarios. Furthermore, the deployment experiment evinces that the simultaneous multi-DNN classification approach outperforms object detection in terms of optimization and efficiency.

The future work includes a plan to investigate incremental learning techniques that enable the system to adapt and readjust on-site using the embedded processor in response to false positives and negatives. Additionally, exploring methods to incorporate spatio-temporal features into the system without compromising its performance would enhance its accuracy and robustness further.

## REFERENCES

[1] C. Álvarez, M. I. González, and A. Gracia, "Flight procedures automation: Towards flight autonomy in manned aircraft," in *Proc. AIAA/IEEE 39th Digit. Avionics Syst. Conf. (DASC)*, Oct. 2020, pp. 1–8.

[2] L. Unzueta, S. Garcia, J. Garcia, V. Corbin, N. Aranjuelo, U. Elordi, O. Otaegui, and M. Danielli, "Building a camera-based smart sensing system for digitalized on-demand aircraft cabin readiness verification," in *Proc. Int. Conf. Robot., Comput. Vis. Intell. Syst.*, Jun. 2020, pp. 98–105.

[3] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Feb. 2020.

[4] W. Wang and Y. Yang, "Development of convolutional neural network and its application in image classification: A survey," *Opt. Eng.*, vol. 58, no. 4, Apr. 2019, Art. no. 040901.

[5] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, Dec. 2022.

[6] F. Wang, H. Han, S. Shan, and X. Chen, "Deep multi-task learning for joint prediction of heterogeneous face attributes," in *Proc. 12th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May 2017, pp. 173–179.

[7] V. S. F. Garnot and L. Landrieu, "Leveraging class hierarchies with metric-guided prototype learning," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Nov. 2021, pp. 22–25.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[9] T. Lin, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, vol. 8693, 2014, pp. 740–755.

[10] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1956–1981, Jul. 2020.

[11] U. Elordi, N. Aranjuelo, L. Unzueta, J. L. Apellaniz, J. García, and O. Otaegui, "SmaCS dataset," Zenodo, OpenAIRE, U.K., CERN, Switzerland, Tech. Rep., Jan. 2023, doi: 10.5281/zenodo.7524808.

[12] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, Oct. 2018, doi: 10.1016/j.neucom.2018.05.083.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[14] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Mach. Learn. Res.*, vol. 97, K. Chaudhuri and R. Salakhutdinov, Eds. Long Beach, CA, USA: PMLR, Jun. 2019, pp. 6105–6114.

[15] M. Tan and Q. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. 38th Int. Conf. Mach. Learn.*, M. Meila and T. Zhang, Eds., vol. 139, Jul. 2021, pp. 10096–10106.

[16] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 87–110, Jan. 2023.

[17] X. Wang, L. L. Zhang, Y. Wang, and M. Yang, "Towards efficient vision transformer inference: A first study of transformers on mobile devices," in *Proc. 23rd Annu. Int. Workshop Mobile Comput. Syst. Appl.*, Mar. 2022, pp. 1–7.

[18] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[19] Y. Xiang and H. Kim, "Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2019, pp. 392–405.

[20] C. Lim and M. Kim, "ODMDEF: On-device multi-DNN execution framework utilizing adaptive layer-allocation on general purpose cores and accelerators," *IEEE Access*, vol. 9, pp. 85403–85417, 2021.

[21] E. Jeong, J. Kim, S. Tan, J. Lee, and S. Ha, "Deep learning inference parallelization on heterogeneous processors with TensorRT," *IEEE Embedded Syst. Lett.*, vol. 14, no. 1, pp. 15–18, Mar. 2022.

[22] E. Jeong, J. Kim, and S. Ha, "TensorRT-based framework and optimization methodology for deep learning inference on Jetson boards," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 5, pp. 51:1–51:26, 2022, doi: 10.1145/3508391.

[23] M. Kim, "Guaranteeing that multilevel prioritized DNN models on an embedded GPU have inference performance proportional to respective priorities," *IEEE Embedded Syst. Lett.*, vol. 14, no. 2, pp. 83–86, Jun. 2022.

[24] B. Cox, J. Galjaard, A. Ghiassi, R. Birke, and L. Y. Chen, "Masa: Responsive multi-DNN inference on the edge," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2021, pp. 1–10.

[25] F. Yu, S. Bray, D. Wang, L. Shangguan, X. Tang, C. Liu, and X. Chen, "Automated runtime-aware scheduling for multi-tenant DNN inference on GPU," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–9.

[26] N. Aranjuelo, J. García, L. Unzueta, S. García, U. Elordi, and O. Otaegui, "Building synthetic simulated environments for configuring and training multi-camera systems for surveillance applications," in *Proc. 16th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, Feb. 2021, pp. 80–91.

[27] N. Aranjuelo, J. L. Apellaniz, L. Unzueta, J. Garcia, S. Garcia, U. Elordi, and O. Otaegui, "Leveraging synthetic data for DNN-based visual analysis of passenger seats," *Social Netw. Comput. Sci.*, vol. 4, no. 1, p. 40, Nov. 2022.

[28] J. Wang and Y. Zhai, "Prototypical Siamese networks for few-shot learning," in *Proc. IEEE 10th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, Jul. 2020, pp. 178–181.

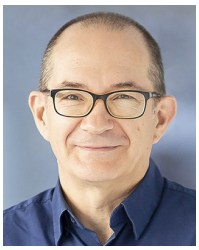
[29] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[30] *Jetson-Aware Embedded Deep Learning Inference Acceleration Framework With TensorRT*. Accessed: May 29, 2023. [Online]. Available: https://github.com/cap-lab/jedi

**UNAI ELORDI** received the degree in computer science from Mondragon University, Spain, in 2009, and the M.S. degree in computational engineering and intelligent systems from the University of the Basque Country (UPV/EHU), Donostia-San Sebastian, Spain, in 2016, where he is currently pursuing the Ph.D. degree. He is a Researcher with the Intelligent Security Video Analytics Department, Fundación Vicomtech, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian. His research interest includes optimizing the deployment of deep neural networks in edge-to-cloud environments for video surveillance applications.

**NEREA ARANJUELO** received the bachelor's and M.S. degrees in industrial technologies engineering from the Tecnun, University of Navarra, Donostia-San Sebastian, Spain, in 2013 and 2015, respectively, and the Ph.D. degree in computer science from the University of the Basque Country (UPV/EHU), in 2023. She is currently a Researcher with Fundación Vicomtech, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian. Her research interests include computer vision, synthetic data generation, and multimodal systems for automotive applications.

**LUIS UNZUETA** received the M.S. and Ph.D. degrees in mechanical engineering from the Tecnun, University of Navarra, Donostia-San Sebastian, Spain, in 2002 and 2009, respectively. He is currently a Principal Researcher with the Intelligent Security Video Analytics Department, Fundación Vicomtech, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian. His current research interests include video-surveillance systems, biometrics, and human–computer interaction.



**JOSE LUIS APELLANIZ** received the M.S. degree in telecommunication engineering from the University of the Basque Country (UPV/EHU), in 1993. He is currently a Research Assistant with Fundación Vicomtech, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain. He is also involved in deep learning and computer vision for automotive applications.

**IGNACIO ARGANDA-CARRERAS** received the M.S. and Ph.D. degrees in computer science and electrical engineering from Universidad Autónoma de Madrid, in 2002 and 2009, respectively. He is currently an Ikerbasque Research Associate with the University of the Basque Country (UPV/EHU), Donostia-San Sebastian, Spain; the Ikerbasque, Basque Foundation for Science, Bilbao, Spain, and the Donostia International Physics Center (DIPC), Donostia-San Sebastian. His research interests include computer vision and bioimage analysis.