IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

CLEVER: Stream-based Active Learning for Robust Semantic Perception from Human Instructions

Jongseok Lee[1,2], Timo Birr[2], Rudolph Triebel[1,2] and Tamim Asfour[2]

Fig. 1: Imagine a robot encountering an unseen object, e.g., trained for apples but tested on a T-shirt. This problem of distribution shifts induce learning algorithms to typically fail. With the proposed system, CLEVER, the robot queries a human when the model is uncertain, and adapt itself to reduce that uncertainty. With human instructions, we demonstrate that such query-and-adaptation capabilities can improve the robustness of DNN-based semantic perception against distribution shifts.

***Abstract*— We propose CLEVER, an active learning system for robust semantic perception with Deep Neural Networks (DNNs). For data arriving in streams, our system seeks human support when encountering failures and adapts DNNs online based on human instructions. In this way, CLEVER can eventually accomplish the given semantic perception tasks. Our main contribution is the design of a system that meets several desiderata of realizing the aforementioned capabilities. The key enabler herein is our Bayesian formulation that encodes domain knowledge through priors. Empirically, we not only motivate CLEVER's design but further demonstrate its capabilities with a user validation study as well as experiments on humanoid and deformable objects. To our knowledge, we are the first to realize stream-based active learning on a real robot, providing evidence that the robustness of the DNN-based semantic perception can be improved in practice. The project website can be accessed at `https://sites.google.com/view/thecleversystem`.**

I. Introduction

With deep neural networks (DNNs), the performance of computer vision increased dramatically, achieving impressive results in the semantic perception tasks, such as object classification, detection, and segmentation [1], [2]. However, such advancements in computer vision may not directly translate to the robotic semantic perception. This is because, in contrast to standard computer vision benchmarks, robots are situated in the physical world, where unpredictable events routinely occur and affect the robustness of the robot's understanding of its own environments. An example is distributional shift scenarios, where DNNs often make unexpected errors due to test conditions being underrepresented in the training data [3], [4]. Thus, to achieve robust semantic perception, several probabilistic techniques have been investigated so far, so that with uncertainty estimates, robots can reason when to trust the predictions from DNNs and when not [5]–[9].

In this paper, we build upon such probabilistic techniques and propose a system called CLEVER. The main idea behind CLEVER is to not only obtain uncertainty estimates from DNNs for probabilistic predictions, but to further reduce the model's uncertainty by asking support from humans and adapting the model online. We achieve this query and adaptation through so-called stream-based active learning (AL) – an autonomous learning paradigm that involves continuously selecting and labeling new data as they arrive in a stream, allowing for adaptation of the model online to changes in data distribution [10]. The outcome of CLEVER is an adaptable DNN for semantic perception. For such capabilities, we equip CLEVER with a continuously adaptable DNN, a Bayesian learning algorithm, and an AL with temporal information. In particular, our Bayesian algorithm learns informative prior – a probability distribution over the model parameters that incorporates domain knolwedge.

CLEVER meets several desiderata of stream-based AL with DNNs in practice. By learning priors, CLEVER is designed to generalize and estimate well-calibrated uncertainty, even with limited data availability. Such prediction capabilities are crucial to request support from humans ("query") only when necessary. CLEVER also addresses the issue of catastrophic forgetting, i.e., the tendency of DNNs to abruptly forget

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Weßling, Germany. [2]Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany.

This work is supported by euROBIN (grant agreement 101070596), Inverse (grant agreement 101136067) and AiSac (Grant No. RS-2024-00441872).

Correspondence to `jongseok.lee@dlr.de`

TABLE I: The proposed system addresses several desiderata for demonstrating a stream-based AL with real robots.

	Both uncertainty and generalization	Ability to ask help and select samples	Addresses catastrophic forgetting in DNNs	Update DNNs fast, e.g., less than 1 minute
Continual learning (i.e., [11], [12])	✗	✗	✓	✓
Existing stream-based AL (i.e., [13]–[15])	✗	✓	✗	✗
Interactive learning (i.e., [16]–[18])	✗	✗	✗	✓
Our system CLEVER	✓	✓	✓	✓

about previously learned tasks when continuously learning a new task [19]. Furthermore, CLEVER can learn a new task in one minute by updating only the relevant parameters of DNNs online while using only fewer but most informative and diverse training data. In the experiments, we provide several ablation studies and comparative assessments to motivate our design choices. Finally, through a user validation study with 13 participants and the deployment of CLEVER on the humanoid robot ARMAR-6 [20], we demonstrate the enhanced robustness in semantic perception with robots.

Contributions and major claims. To the best of our knowledge, CLEVER is the first stream-based active learning system with DNNs, shown in a physical system for robotic perception tasks. Moreover, unlike existing works, we apply stream-based active learning for securing robustness in semantic perception tasks. To enable this novel capability, we identify new system requirements and challenges (Section III), followed by CLEVER's design that meets these requirements within a single framework (Section IV). CLEVER is evaluated in response to these requirements. In particular, we show that our Bayesian formulation with learning-based priors enhances the practicality of CLEVER (Section V-A). Through a user validation study that involves arbitrary objects (Section V-B), and demonstrations on a humanoid robot for deformable object perception (Section V-C), we create distributional shift scenarios for evaluation. Even under these challenging scenarios, we show that CLEVER can eventually accomplish the given perception tasks, improving the robustness of the DNN-based semantic perception in the real world.

II. Related Work

Our primary contribution is in the area of AL. For this, we bring Bayesian methods for neural networks and interactions with humans for robot learning. Thus, we locate our work within these areas. Tab. I summarizes our main novelty.

Stream-based active learning. Active Learning (AL) is a paradigm in which a learning algorithm identifies the most useful unlabeled instances to learn from [10]. In the literature, a pool of unlabeled instances is mostly assumed, resulting in the so-called pool-based AL [21]–[23]. In contrast, we focus on a setting in which data arrive in the stream, which is an underexplored area of AL [10]. For robotic perception, the early attempts for stream-based AL relied on classical learning techniques such as Gaussian Processes [24], boosting [25] and bagging [15]. Yet, the current de-facto standard in object recognition relies on deep learning, urging for extensions of stream-based AL to DNNs. Although current extensions [13], [14] study the feasibility of stream-based AL using DNNs, the discussions therein are centered on strategies for informative data selection from the data stream.

Indeed, the central objective of AL is to reduce the cost of labeling by querying and selecting the most informative data [10]. In contrast, our focus is applying AL to enhance the robustness of robotic perception by seeking human support and updating DNNs online. A similar use case was also previously mentioned by Triebel et al. [15], [24]. However, due to a different focus, no real system was therein developed and evaluations were limited to showing sample efficiency, i.e., accuracy increase per newly added data points. Instead, we take a systems approach to the problem, thereby developing CLEVER that meets various requirements reported in Tab. I.

Bayesian adaptation of neural networks. Learning semantics from new streams of observation is a crucial capability for our system. In robotics, such adaptations with DNN have previously been investigated [11], [12]. Their findings suggest that continual learning during deployment improves the accuracy of the robot's perception when compared to fixed, pretrained DNNs. However, their applicability to stream-based AL is limited, since no uncertainty estimates are available for the query and selection step of AL. In contrast, our work explores Bayesian methods that are well suited for stream-based AL within a unified framework. For this, we extend our previous work [26] to stream-based AL. We previously showed how continual learning can be performed while obtaining well-calibrated uncertainty estimates and generalization with DNNs using few data samples [26]. We point out that such properties are desired for developing a complete stream-based AL system with real robots.

Robot learning from humans. In robotics, many works on learning from demonstration focus on mapping the states of robots to actions [27]. Yet, a recent work [7] shares a similar spirit to ours, i.e., an algorithm is designed to ask for human help using uncertainty estimates. However, their focus is on robot planning using language models. An idea on the correction of DNNs with language instructions from humans is also being explored for policy learning [28]. Many researchers have investigated incremental and interactive learning of new objects using human instructions [16]–[18]. Among them, we extend the works on a humanoid robot, ARMAR-III, where the robot demonstrated interactive learning of unknown objects from human instructions [18], [29]. These early attempts showed that new rigid objects, such as books, can be learned using hand-crafted visual features and a k-nearest-neighbor classifier. Our extensions provide (a) the use of DNNs, (b) stream-based AL that reduces model uncertainty, and (c) fewer prior assumptions about the object.

III. System Concept and Challenges

Whenever uncertain, our system seeks human support and improves itself online based on human instructions. The

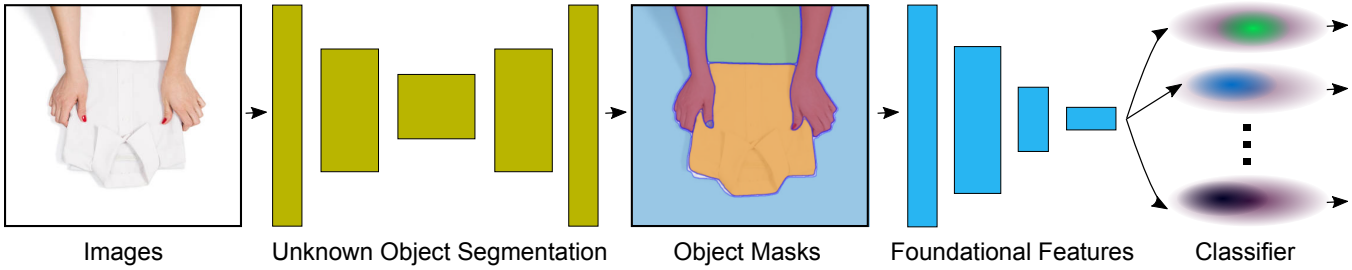


Fig. 2: Our prediction model for stream-based AL using DNNs. For semantic segmentation, our pipeline combines an unknown object segmentation (Segment-Anything), foundational representations (DinoV2) and a classifier based on Bayesian Neural Networks (BNNs) with multilayer perceptron. BNNs are represented as Gaussian distribution over the model parameters.

main problem encountered is distribution shifts that the robot inevitably encounters during the deployment of a DNN-based model, which ultimately produces misleading and overconfident predictions. For example, a robot may face unknown objects. Imagine a robot trained to classify apples but spots bananas during deployment. Deformable objects present similar challenges if the induced deformation in the object's shape is underrepresented in the training data. Given this problem, the concept of our system is to ask for help from humans and adapt the model online (see Fig. 1).

There are several challenges (see Tab. I). First, the system should know when the model is uncertain. This requires a probabilistic treatment that provides well-calibrated uncertainty estimates under distribution shifts. Given a training data $\mathcal{D}$ and a DNN, $f_{\boldsymbol{\theta}}$, where all learnable parameters are stacked as a vector $\boldsymbol{\theta}$, probabilistic treatments of the given problem infer the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ and compute a predictive distribution $p(\mathbf{y}^*|\mathbf{x}^*,\mathcal{D})$ for new test datum $(\mathbf{x}^*,\mathbf{y}^*) \notin \mathcal{D}$. Here, acquired data cannot be of large amounts as we rely on manual human instructions. Thus, we need DNNs that generalize with small amount of data. Second, the system should adequately support queries from the human and a decision-making framework that enables effective instruction of the robot by a human. The former minimizes the need for repeated human querying. We select a subset of data $\overline{\mathcal{D}}^* \subseteq \mathcal{D}^*$ that is the most informative to learn. This reduces the training time. Third, the model should continually learn and produce accurate predictions without forgetting. Lastly, for demonstrations on a humanoid, DNNs are to be trained fast, e.g., under one minute. To achieve this, CLEVER trains only a subset of model parameters $\overline{\boldsymbol{\theta}} \subseteq \boldsymbol{\theta}$ and selected data $\overline{\mathcal{D}}^*$.

IV. CLEVER – A Stream-based Active Learner

Having discussed the system concept and challenges, we describe the design of CLEVER in more detail.

A. Underlying prediction model for continual adaptation

Our first component is a prediction model for semantic perception (see Fig. 2). The overall model takes images as input and outputs segmentation masks with their associated output labels. For this, we rely on an unknown object segmentation such as Segment-Anything [1] to generate the segmentation masks. A tracker can be combined to achieve a faster runtime of the model [1]. Then, we stack the obtained masks and extract features using foundational models such as DinoV2 [2]. The obtained features are used as input of a classifier, which generates output labels for each segmentation mask. In contrast to existing models such as Mask-RCNN, our construction allows open-set extraction of segmentation masks and state-of-the-art visual features using pre-trained models. This means we only need to train a classifier. In lieu of sophisticated pipelines for semantic segmentation such as Mask-RCNN, we focus on AL for classification tasks while also obtaining detection and segmentation results.

Next, we present the design of our classifier for stream-based AL. The classifier learns one multilayer perceptron (MLP) per object class. We call each of these MLPs the heads of our classifier. If we had an apple and a banana, our classifier would have two heads, each responsible for only one object class. In this way, a multi-class classification task is tackled using a combination of binary classifiers with calibrated uncertainties. The proposed construction brings two advantages. First, we can mitigate catastrophic forgetting by design. As we create a new head for a new incoming class, learning new objects does not affect the previously learned heads. Moreover, in each learning cycle, we can update only a head of our classifier. For example, if a head responsible for an apple exhibits high uncertainty, we can only learn to classify an apple better. Once the classifier is confronted with an unknown object, we can add and learn one new head. Training a smaller model can be more efficient. Achieving these results with a multi-class classifier might be difficult.

Within this construction, we apply probabilistic inference on all MLPs to obtain Bayesian Neural Networks (BNNs), which can provide well-calibrated uncertainty under distribution shifts. Let us define the training data for the classifier as $\mathcal{D} = ((\mathbf{x}_i, \mathbf{y}_i))_{i=1}^N$ where the inputs $\mathbf{x}_i$ are the extracted features. We use superscripts (c,t) to denote the available C classes and T tasks, respectively. Then, a classifier $f_{\boldsymbol{\theta}}$ is now divided into several binary classifiers $f_{\boldsymbol{\theta}^{(c,t)}}$ that output $\mathbf{y}_c \in \{0,1\}$. The corresponding DNN's learnable parameters $\boldsymbol{\theta}^{(c,t)}$ belong to the head c for the task t, which is obtained using the relevant training data $\mathcal{D}^{(c,t)}$. BNNs apply probabilistic inference to DNNs, e.g., models are represented by the posteriors $\rho^{(c,t)} = p(\boldsymbol{\theta}^{(c,t)}|\mathcal{D}^{(c,t)})$. BNNs predict through marginalization:

$$p(\mathbf{y}_c^* = 1|\mathbf{x}^*, \mathcal{D}^{(c,t)}) = \int p(\mathbf{y}_c^*|\mathbf{x}^*, \boldsymbol{\theta}^{(c,t)}) p(\boldsymbol{\theta}^{(c,t)}|\mathcal{D}^{(c,t)}) d\boldsymbol{\theta}^{(c,t)}. \tag{1}$$

Intuitively, instead of one model that best fit the data, BNNs

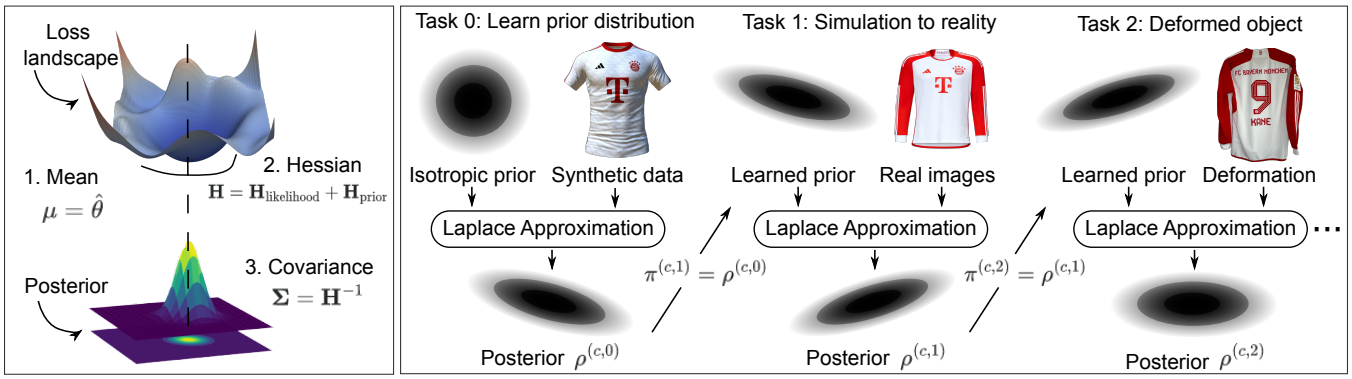


Fig. 3: The proposed pipeline to learn a prediction model for stream-based AL. Left: Laplace Approximation (LA) infers the posterior distribution of a DNN as a Gaussian distribution. Right: Using LA to obtain BNNs and further exploiting training data that encodes our domain knowledge, we learn an informative prior from a posterior of a previous task. Bayesian learning in sequence addresses the potential prior misalignment where humans provide the relevant task and data to learn the prior.

pay tribute to the model uncertainty for predictions. The probabilities obtained better reflect the true belief in the class $\mathbf{y}_c^*$ than the often used softmax scores that tend to bias towards higher probabilities, or overconfident predictions [26], [30].

We then combine the calibrated binary classifiers using BNNs for handling the given multi-class classification task. Given a single test input $\mathbf{x}^*$, we obtain a vector of probabilities from all heads: $\boldsymbol{p} = (p_1, p_2, ..., p_C)$ where $p_c = p(\mathbf{y}_c^* = 1|\mathbf{x}^*, \mathcal{D}^{(c,t)})$. Then, we pick the predicted class label by:

$$\mathbf{y}^* = \text{argmax}_c(\boldsymbol{p}) \text{ with } p(\mathbf{y}^* = c|\mathbf{x}^*) = p_{c=\mathbf{y}^*}. \tag{2}$$

In this way, we select one output of a head with the highest probabilities and choose the corresponding class label and probability score. We do not consider one vs. rest multi-class strategy. Hence, the vector $\boldsymbol{p}$ itself is not a valid probability distribution. For querying from humans and selecting the most informative data using uncertainty estimates, a valid probability distribution for the most likely class is sufficient. Thus, we do not require probabilities for all classes at once. Moreover, the given design choice enables our system to be more efficient when training, because we do not need to update all the heads for training a single head – an assumption in one vs. rest multi-class strategy. Instead, we can update each head individually for efficiency.

B. Bayesian learning for uncertainty and generalization

We now present our training pipeline for the aforementioned probabilistic model. Our goal is to obtain the posteriors $\rho^{(c,t)}$ for the estimation of the uncertainty. We also aim to address the challenge of generalization under a small data regime with DNNs. For this, our main idea is to learn the prior distribution over the parameters of DNNs. To explain, according to Bayes rule, the posteriors are proportional to the likelihood $\prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\theta}^{(c,t)})$ and the prior $\pi^{(c,t)} = p(\boldsymbol{\theta}^{(c,t)})$, i.e., $\rho^{(c,t)} = p(\boldsymbol{\theta}^{(c,t)}|\mathcal{D}^{(c,t)}) \propto \pi^{(c,t)} \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\theta}^{(c,t)})$. Due to the non-linearity of DNNs, no closed-form solution exists for the posteriors. Thus, we need approximate Bayesian inference – a set of algorithms that approximate the intractable posteriors. For this, the first step is to define the priors over the DNN parameters. Traditionally, a zero-mean isotropic Gaussian prior – an uninformative prior that regularizes the overall model – was seen to be sufficient for DNNs. However, when no large amounts of data are available, the likelihood no longer dominates the posterior. Thus, specifying an informative prior can improve the approximate Bayesian inference [26]. Fig. 3 shows our pipeline, which we later combine with a generalization framework called the PAC-Bayes theory.

The first step of our pipeline is to learn an initial prior distribution offline using synthetic data (task 0 in Fig. 3). Synthetic data for object recognition can be generated by either photorealistic synthesizers such as BlenderProc2 or generative models such as StableDiffusion with relevant prompts like "A jersey on a table". Synthetic data has the advantage that large amounts of annotated training data can be generated in a cost-effective manner. Hence, for all the known classes of objects, we generate training data $\mathcal{D}^{(c,t=0)}$. Now, in order to learn an initial prior, we apply Laplace Approximation (LA). LA is an approximation inference method that imposes Gaussian distribution on the DNN parameters around a local mode [30]. The obtained posterior has its mean as the maximum-a-posterior (MAP) estimates $\boldsymbol{\mu}^{(c,t)} = \hat{\boldsymbol{\theta}}^{(c,t)}$, which can be obtained using the standard DNN training procedure with a cross-entropy loss. In LA, the covariance of the posterior $\mathbf{\Sigma}^{(c,t)}$ is estimated by an inverse of a loss landscape's Hessian $\mathbf{H}^{(c,t)}$, i.e., a second order derivative of log posterior w.r.t the DNN parameters $\boldsymbol{\theta}^{(c,t)}$. By definition, $\mathbf{H}^{(c,t)} = \mathbf{H}^{(c,t)}_{\text{likelihood}} + \mathbf{H}^{(c,t)}_{\text{prior}}$. Assuming an isotropic prior,

$$\begin{aligned} &\text{Prior:} \quad \pi^{(c,0)} = \mathcal{N}(\mathbf{0}, \gamma\mathbf{I}), \\ &\text{Posterior:} \quad \rho^{(c,0)} \approx \mathcal{N}(\hat{\boldsymbol{\theta}}^{(c,0)}, (\mathbf{H}^{(c,0)}_{\text{likelihood}} + \gamma\mathbf{I})^{-1}), \end{aligned} \tag{3}$$

are the prior-and-posterior pairs. The posteriors at $t = 0$ is then used as priors for task $t = 1$. We use a layer-wise Kronecker factorization [26] to compute the Hessian or the covariance.

Having learned the priors, we now iterate the learning process online. Examples of incoming tasks are semantic segmentation on real images, or recognition of deformable objects, as depicted Fig. 3. In each step, the approximated posteriors from the previous tasks are used as the priors for

the current learning tasks. Intuitively, as we keep learning one object class per head, such Bayesian learning results in positive transfers across the tasks, i.e., posteriors that classify folded clothes helps in learning unfolded clothes, even with small amounts of data. For this, we repeat LA and obtain:

$$\text{Prior:} \quad \pi^{(c,t)} = \mathcal{N}(\hat{\boldsymbol{\theta}}^{(c,t-1)}, (\mathbf{H}^{(c,t-1)})^{-1}), \tag{4}$$
$$\text{Posterior:} \quad \rho^{(c,t)} \approx \mathcal{N}(\hat{\boldsymbol{\theta}}^{(c,t)}, (\mathbf{H}^{(c,t)}_{\text{likelihood}} + \mathbf{H}^{(c,t-1)})^{-1}).$$

The prior-and-posterior pairs are obtained with small modifications to LA. First, the maximum-a-posterior estimates are computed using a more expressive prior, instead of initializing around zero with one variance for all DNN parameters. Second, for approximating the posteriors, the Hessian from the posterior of previous task is used instead of an isotropic term $\gamma\mathbf{I}$. Our pipeline adapts our previous method [26] to better fit our use case. Unlike previously, we do not attempt to transfer across heads. This enables class-independent learning. Moreover, we embed the domain knowledge with synthetic data at task 0, instead of foundational models.

For these steps of Bayesian learning, we can explicitly design for improving generalization of the model under small data regime. We achieve this by introducing hyperparameters τ, α and β s.t. $\mathbf{H} = \tau(\beta\mathbf{H}_{\text{likelihood}} + \alpha\mathbf{H}_{\text{prior}})$, and optimizing for a generalization objective called PAC-Bayes bounds. Note that we dropped the superscript (c,t) for notation simplicity when explaining PAC-Bayes theory. The hyperparameters α and β decide how much weight should be given to the previous task (prior) and the current data at hand (likelihood). If more weight is given to the current data, the model may overfit, while more weight to the prior may result in underfitting. The tempering term τ scales the overall posterior. We pick these hyperparameters by minimizing an upper bound to the expected loss $\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathcal{L}^l_P(f_{\boldsymbol{\theta}})]$ on the true distribution P. A true expected loss incurs over the true data distribution P – not only on training and test data. Such generalization bounds depend on empirical loss on the training data $\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\hat{\mathcal{L}}^l_{\mathcal{D}}(f_{\boldsymbol{\theta}})] = \frac{1}{N}\sum_{i=1}^{N}\hat{\mathcal{L}}^l_{\mathcal{D}}(f_{\boldsymbol{\theta}})$ and the KL-divergence of the priors and the posteriors. Our KL-divergence is dependent on τ, α and β. For $\epsilon > 0$, the so-called PAC-Bayes bounds are defined as:

$$P_{\mathcal{D}\sim P}(\forall\rho \ll \pi : \ \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathcal{L}^l_P(f_{\boldsymbol{\theta}})] \tag{5}$$
$$\leq \delta(\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\hat{\mathcal{L}}^l_{\mathcal{D}}(f_{\boldsymbol{\theta}})], \mathrm{KL}(\rho\|\pi), N, \varepsilon)) \geq 1-\varepsilon.$$

For more details, we refer to our previous work [26] where we devised a computationally tractable method for LA.

C. A temporal active learning system with humans

The remaining challenges are to develop a system that (a) asks for help from humans and (b) selects the most informative samples to learn from. For both components, we combine the temporal information inherent in data streams.

Our query strategy involves a recursive rule that keeps the current probabilities about an object class given all measurements until step k: $p(\mathbf{y}^*_c = 1|\mathbf{x}^*_{1:k})$. Defining $l(\bullet) = \log p(\bullet)[1-p(\bullet)]^{-1}$ from which we can retrieve the current probabilities $p(\bullet) = 1-(1+\exp[l(\bullet)])^{-1}$, our recursive form

begin
  // Initialization
  $\rho^{(c,0)} \leftarrow \text{prior}(\mathcal{D}^{(c,0)}, \pi^{(c,0)})\ \forall c$; // Eq. 3
  $\alpha^{(c,0)}, \beta^{(c,0)}, \tau^{(c,0)} \leftarrow \text{pac-bayes}(\bullet)\ \forall c$; // Eq. 5
  // Main Loop
  while incoming image stream **do**
    $p_c|\mathbf{x}^*_k \leftarrow \text{marginalization}(\mathbf{x}^*_k)\ \forall c$; // Eq. 1
    $p_c|\mathbf{x}^*_{1:k-1} \leftarrow \text{filter}(p_c|\mathbf{x}^*_k)\ \forall c$; // Eq. 6 (option).
    $\mathbf{y}^*, p_{c=\mathbf{y}^*} \leftarrow \text{prediction}(\boldsymbol{p}|\mathbf{x}^*_{1:k-1})$; // Eq. 2
    if query humans for head c **then**
      $\mathcal{D}^{(c,t)}_{new} \leftarrow$ human-instruction() ; // Fig. 1
      $\bar{\mathcal{D}}^{(c,t)} \leftarrow \text{selection}(\mathcal{D}^{(c,t)}_{new})$; // Eq. 7
      $\rho^{(c,t)} \leftarrow \text{posterior}(\bar{\mathcal{D}}^{(c,t)}, \pi^{(c,t)})$; // Eq. 4
      $\alpha^{(c,t)}, \beta^{(c,t)}, \tau^{(c,t)} \leftarrow \text{pac-bayes}(\bullet)$; // Eq. 5
    end
  end
end

Algorithm 1: CLEVER - stream-based active learner.

based on a binary state Bayes filter is given by:

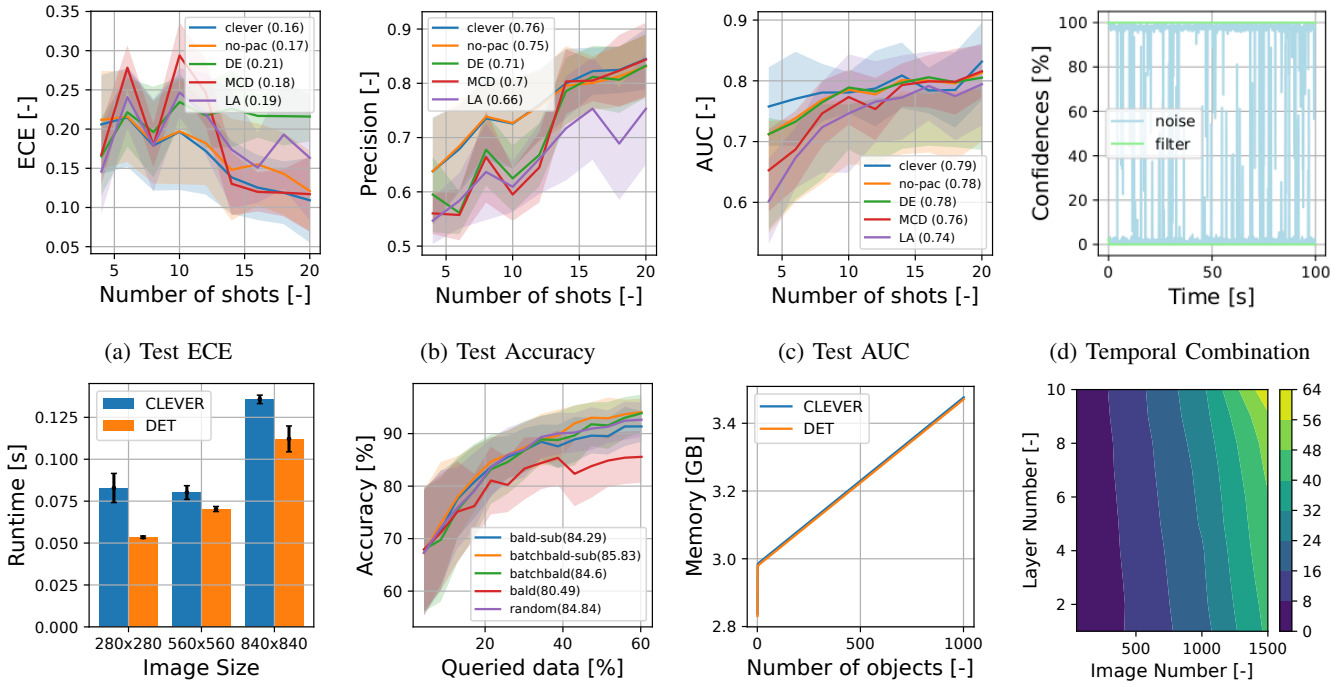
$$l(\mathbf{y}^*_c = 1|\mathbf{x}^*_{1:k}) = l(\mathbf{y}^*_c = 1|\mathbf{x}^*_k) + l(\mathbf{y}^*_c = 1|\mathbf{x}^*_{1:k-1}) - l(\mathbf{y}^*_c = 1). \tag{6}$$

The obtained probabilities are converted to a normalized entropy as a measure of uncertainty, and we use a pre-defined threshold for query decisions [24]. Here, temporal information may filter the outliers and augment the tracking of object segmentation. Our design choice on utilizing binary classifiers per head also enables us to simplify the incorporation of temporal information. Instead of complex models such as Dirichlet, we only modified the algorithm behind the well-known probabilistic grid map [31] that tracks uncertainty on binary states such as occupied or non-occupied space.

Next, given a human demonstration, AL selects the most informative data. For CLEVER, such a selection results in smaller training data, which makes the continual adaptation of the underlying model more efficient. We utilize the so-called BatchBald [32] to select the top $\bar{B} \subset B^{(c,t)}$ data points:

$$\mathcal{A}_{\text{batchbald}}(\mathbf{x}^*_{1:B}, \rho^{(c,t)}) = \mathbb{I}(\mathbf{y}^*_{c,1:B}, \boldsymbol{\theta}^{(c,t)}|\mathbf{x}^*_{1:B}, \mathcal{D}^{(c,t)}_{\text{new}})$$
$$= \mathbb{H}(\mathbf{y}^*_{c,1:B}|\mathbf{x}^*_{1:B}, \mathcal{D}^{(c,t)}_{\text{new}}) - \mathbb{E}_{\rho^{(c,t)}}\mathbb{H}(\mathbf{y}^*_{c,1:B}|\mathbf{x}^*_{1:B}, \boldsymbol{\theta}^{(c,t)}, \mathcal{D}^{(c,t)}_{\text{new}}). \tag{7}$$

Intuitively, this acquisition function examines the mutual information $\mathbb{I}$ between the multiple model predictions and the model parameters. Such mutual information is obtained by approximations to the entropy terms $\mathbb{H}$. The coupling between the model predictions for a batch of data points and the model parameters is captured, and data points with high mutual information would inform us more about the true model parameters. This allows us to again combine temporal information by considering a batch of data points. We further combine a subsampling strategy [22], that is, we randomly select a subset from the demonstration data before applying BatchBald. This ensures the diversity of the samples while speeding up the computations of BatchBald.



(a) Test ECE (b) Test Accuracy (c) Test AUC (d) Temporal Combination

(e) Run-time Analysis (f) Acquisition Functions (g) Memory Analysis (h) Training Analysis

Fig. 4: Results from ablation studies and comparative assessments of various design choices. The unit for (h) is seconds.

D. System overview and implementation details

Alg. 1 provides an overview of CLEVER, which shows how all the components are integrated into a single algorithm. For each task, humans instruct the semantic information about an object using speech and demonstrate the given object from different viewpoints. Negative examples from the previous demonstrations are also provided for training each head. For unknown objects, we add a new head and start the learning process with an isotropic prior. In Alg. 1, • are used to indicate any arbitrary but relevant inputs to a function.

V. Results

A. Ablation studies and comparative assessments

We provide ablation studies and comparative assessments to provide insights into the design of CLEVER. The analysis is focused on the continual learning architecture, the impact of using posteriors as informative priors, and combining temporal information. In particular, we focus on how our design choices address the challenges listed in Tab. I. For this, we collect a dataset from ARMAR-6, which consists of images from ten household objects. CAD models of these objects are obtained using an Android app called MagiScan. We also used StableDiffusion for synthetic data generation. For each object, we provided nine sequences of human demonstrations with 250 images each. We varied the difficulty level by providing more deformations to the objects, for example. We note that the dataset will be released to the public. There were no open-source datasets which fit our stream-based AL scenario due to the human element. We assumed that segmentation is given by foundational models like SAM, and mainly evaluate the underlying classifier.

Continuously adaptable model. First, we present our analysis on CLEVER's ability to perform continual learning. We claimed that the proposed design enables training of DNNs in less than one minute, while addressing the catastrophic forgetting problem. To evaluate the training time, we vary the number of layers from one to ten, and also vary the number of training data points up to 1500. Because we only train the MLP while fixing the representations from a foundational model, the results show that our classifiers can be updated in less than 1 minute (Fig. 4h). We note that a three-layered MLP is used for all other experiments. Regarding the catastrophic forgetting, CLEVER adapts a progressive architecture where new heads are trained for new incoming object categories. By design, such architecture-based approaches mitigate the forgetting. However, growing the DNN architecture may hurt the computational scalability. To evaluate such computational scalability, we grow the DNN architecture to accommodate 1000 object categories. Comparisons in terms of memory and runtime are provided with CLEVER without probabilistic treatments (denoted DET). The results are depicted in Fig. 4g and 4e. Without an elaborated mechanism to mitigate catastrophic forgetting, CLEVER can learn 1000 object categories with less than 4GB of GPU peak memory and interactive frame rates. We also note that forgetting mechanisms can be introduced when an application scenario demands many more object categories.

Bayesian learning algorithm. Secondly, we examine the influence of prior learning method for both uncertainty and generalization under small data regime. For this, we split the dataset into training and test set with a ratio of 8:2. Then, we randomly pick N-shot images per object category

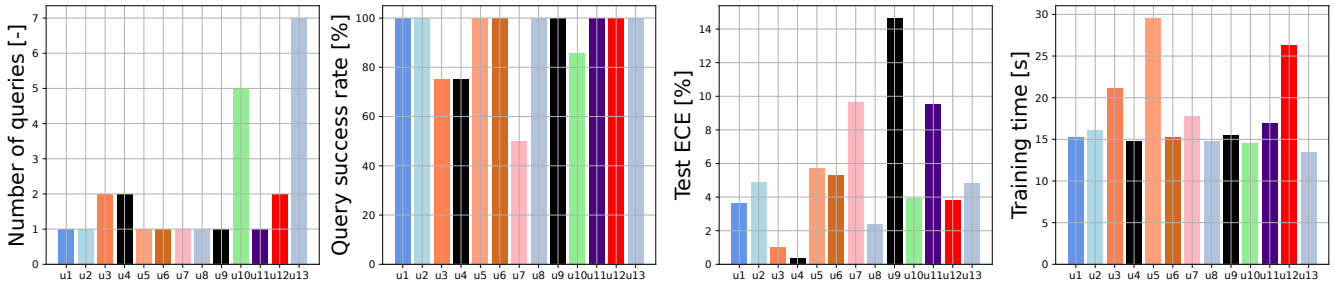


Fig. 5: Results of a complete open-set evaluation with 13 users (x-axis). Number of queries to semantically segment the objects with more than 85% confidence (lower the better), query success rate (higher the better), ECE as a measure of uncertainty (lower the better), and training time (lower the better) are reported from the experiments per user. These experiments validate that various users can perform active learning with CLEVER for semantic segmentation under open-set conditions.

TABLE II: Query success rates, ECE, accuracy, and number of queries to reach 85% accuracy are reported.

	Success rate	ECE	Precision	Nr. Queries
CLEVERv1	**0.887±0.049**	**0.060±0.017**	**0.933±0.020**	**1.539±0.544**
CLEVERv2	0.826±0.053	0.092±0.036	0.900±0.034	2.440±0.866
Vanilla	0.801±0.054	0.177±0.021	0.817±0.039	4.320±0.992

up to 20 images. Using Google's uncertainty baseline, we evaluate various models (MC Dropout [9] as MCD, Deep Ensemble as DE [8], Laplace Approximation as LA [30], and no-pac means CLEVER without PAC-Bayes optimization [26]) with expected calibration error (ECE), precision and area under ROC curve (AUC) as the standard evaluation metrics. The findings are shown in Fig. 4c, 4b and 4c, where the chosen metrics are reported by averaging over the object categories. Five random seeds were used. The results show that CLEVER can outperform the chosen baselines. In particular, comparisons to LA and no-pac show that learning the prior from simulation, and optimizing for a generalization bound can improve the performance for the chosen evaluation scenario of stream-based AL.

Temporal information. Third, we examine the idea of combining the temporal information when selecting the subset of images to learn more efficiently, and also deciding to query. For the former, we choose a uniform sampling, BALD, BatchBALD and their combinations with sub-sampling as baseline acquisition functions. For the latter, we train a model to provide noisy confidence estimates, and display filtered output against the raw output from a single query step of 100s. The same train-test split of 8:2 was used for the comparisons on acquisition functions. A total of 15 query steps were generated for all object categories. Five random seeds were used to obtain the results. For the combination of temporal data via filtering, we observe that noises can be removed (Fig. 4d). Moreover, BatchBALD with the sub-sampling strategy, as the acquisition function looks at the batch of data points to measure information gain, outperformed other baselines in Fig. 4f. These findings motivate our design choice of integrating temporal information for performance improvements.

Evaluation. Finally, we examine the final performance (see Tab. II). We assume that images arrive in a batch of streams over nine subsequent demonstrations with increasing levels of difficulty. Metrics of choice were query success rate, i.e., if the model queried correctly, average ECE and precision, and the number of queries required to reach more than 85 % precision. These metrics capture several requirements of a stream-based AL system. 40 data points were selected for training in each demonstration out of 250 data points so that the training terminates in less than a minute. Five random seeds were used. We compared three baselines. Vanilla corresponds to a deterministic model of CLEVER without any priors. Version 1 used the full formulation of the prior with both mean and covariance, while version 2 only utilized the mean by pretraining with the given synthetic data. We observe the gradual increase in performance in all metrics. These results justify our design choices, in particular, the prior learning with the targeted synthetic data for stream-based AL.

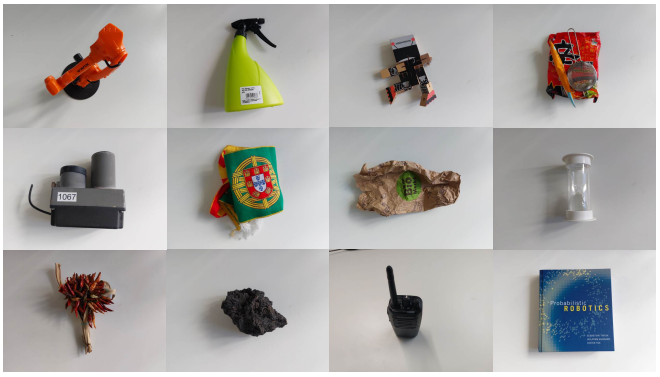


Fig. 6: Examples of arbitrary objects brought by different users. The objects ranged from articulated, transparent, deformable, industrial and planetary objects.

B. A complete open-set evaluation with users

We evaluate the performance of CLEVER in an open-set condition, where the model encounters unseen objects. Furthermore, the goal is to perform user validation in order to show that many users can use the system successfully. To achieve this goal, we randomly invited 13 users and asked them to bring any object of their choice. Fig. 6 shows the example objects that were brought by the users to test the system. Since we did not know these objects a priori, we could create a truly open-set condition. Initial prior from Section V-A was used by pretraining with object and no-object classes.

The users were instructed to use CLEVER in order to perform semantic perception of the objects they brought. Under these conditions, we measured the number of queries to learn the new object with more than 85% confidence, the number of query failures, test ECE, and training time. The users only collected 80 images per query step. Out of 80 images, CLEVER selected 32 images to adapt the model. In each query step, posterior of the previous task was used as prior, along with an optimization for PAC-Bayes bounds.

The results are depicted in Fig. 5. The average number of required queries was 2.0 ± 1.79, while we had a query success rate of $91.20 \pm 15.06\%$ where CLEVER associated well-calibrated confidence and appropriately asked for help. The average ECE was $5.36 \pm 3.75\%$, and CLEVER consumed 17.79 ± 4.717s training time. All users were able to work with CLEVER, and perform stream-based AL under the replicated open-set condition. We also believe that a training time of less than 20s can be practical. Regarding limitations, small distributional shifts seem to be an issue. For example, we found it difficult to obtain well-calibrated uncertainty estimates when training a DNN with an apple but testing with an object similar to an apple in appearance, like a red pear. Nevertheless, all the objects brought by the user could be eventually conquered with CLEVER, which could have been difficult without adaptations at test-time. In this sense, our experiments show the relevance of stream-based AL in developing a persistent vision system.

C. Demonstration on a humanoid robot

Finally, we demonstrate CLEVER on the KIT's humanoid robot, ARMAR-6 [20] (see Fig. 1) where we examine the feasibility of stream-based AL on a real robot. Three objects, namely an apple, a banana, and a T-shirt, are considered. ARMAR-6's onboard cameras, speech interface, and NVIDIA GeForce GTX 1080 are utilized [20]. Pre-designed language prompts were used to allow robot-human communication. The videos are on our project website. On ARMAR-6, we showcase CLEVER's ability to perform robust semantic perception. We emphasize that, for all examined scenarios, deploying a standard DNN without any ability to adapt would have failed to complete the given perception tasks. In contrast, CLEVER is able to improve the robustness of deploying DNNs on a real robot. That is, CLEVER estimates uncertainties, asks for help from humans, and adapts itself to finally accomplish the given perception task. With these results, we illustrate robust DNN-based perception by showing the feasibility of stream-based AL on a real robot. [1]

VI. Conclusion

In this work, we propose a stream-based active learner for robust semantic perception with robots. Experimentally, ablation studies provide the insights behind the system's design. We also evaluate CLEVER in an open-set condition with a user validation, in which participants brought various objects that are transparent, deformable, articulated, industrial, and planetary. CLEVER is also integrated into a humanoid robot. These results generally suggest the possibilities of robust semantic perception, while embracing the predictive performance of deep learning. In future, improvements in unknown object segmentation techniques will also help our system. Thus, as a next step, we envision active learning on foundational models directly for open-set recognition tasks.

[1] All implementation details including metrics, synthetic data generation and in-depth discussions are in suplementary materials of our project website.

References

[1] A. Kirillov, *et al.*, "Segment anything," in *ICCV*, 2023, pp. 4015–4026.
[2] M. Oquab, *et al.*, "DINOv2: Learning robust visual features without supervision," *TMLR*, 2024.
[3] S. Ancha, *et al.*, "Deep evidential uncertainty estimation for semantic segmentation under out-of-distribution obstacles," in *ICRA*, 2024.
[4] J. Feng, *et al.*, "Topology-matching normalizing flows for out-of-distribution detection in robot learning," in *CoRL*, 2023.
[5] K. Sirohi, *et al.*, "Uncertainty-aware panoptic segmentation," *RA-L*, vol. 8, no. 5, pp. 2629–2636, 2023.
[6] J. Lee, *et al.*, "Trust your robots! predictive uncertainty estimation of neural networks with sparse gaussian processes," in *CoRL*, 2021.
[7] A. Z. Ren, *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *CoRL*, 2023.
[8] B. Lakshminarayanan, *et al.*, "Simple and scalable predictive uncertainty estimation using deep ensembles," *NeurIPS*, vol. 30, 2017.
[9] Y. Gal *et al*, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.
[10] D. Cacciarelli and M. Kulahci, "Active learning for data streams: a survey," *Machine Learning*, vol. 113, no. 1, pp. 185–239, 2024.
[11] H. Blum, *et al.*, "Self-improving semantic perception for indoor localisation," in *CoRL*, 2022.
[12] J. Frey, *et al.*, "Continual adaptation of semantic segmentation using complementary 2d-3d data representations," *RA-L*, vol. 7, no. 4, 2022.
[13] A. Saran, *et al.*, "Streaming active learning with deep neural networks," in *ICML*, 2023, pp. 30 005–30 021.
[14] S. Schmidt *et. al*, "Stream-based active learning by exploiting temporal properties in perception with temporal predicted loss," *BMVC*, 2023.
[15] A. Narr, *et al.*, "Stream-based active learning for efficient and adaptive classification of 3d objects," in *ICRA*, 2016, pp. 227–233.
[16] P. Azagra, *et al.*, "Incremental learning of object models from natural human–robot interactions," *T-ASE*, vol. 17, no. 4, pp. 1883–1900, 2020.
[17] S. Valipour, *et al.*, "Incremental learning for robot perception through hri," in *IROS*, 2017, pp. 2772–2777.
[18] D. Schiebener, *et al.*, "Segmentation and learning of unknown objects through physical interaction," in *Humanoids*, 2011, pp. 500–506.
[19] L. Wang, *et al.*, "A comprehensive survey of continual learning: theory, method and application," *T-PAMI*, 2024.
[20] T. Asfour, *et al.*, "Armar-6: A high-performance humanoid for human-robot collaboration in real-world scenarios," *RAM*, vol. 26, no. 4, 2019.
[21] M. Noseworthy, *et al.*, "Active learning of abstract plan feasibility," in *RRSS*, 2021.
[22] J. Feng, *et al.*, "Bayesian active learning for sim-to-real robotic perception," in *IROS*, 2022, pp. 10 820–10 827.
[23] O. Akcin, *et al.*, "Fleet active learning: A submodular maximization approach," in *CoRL*, vol. 229, 06–09 Nov 2023, pp. 1378–1399.
[24] R. Triebel, *et al.*, "Driven learning for driving: How introspection improves semantic mapping," in *ISRR*, 2016, pp. 449–465.
[25] D. Mund, *et al.*, "Active online confidence boosting for efficient object classification," in *ICRA*, 2015, pp. 1367–1373.
[26] D. Schnaus, *et al.*, "Learning expressive priors for generalization and uncertainty estimation in neural networks," in *ICML*, 2023.
[27] H. Ravichandar, *et al.*, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, no. 1, pp. 297–330, 2020.
[28] L. X. Shi, *et al.*, "Yell at your robot: Improving on-the-fly from language corrections," *RSS*, 2024.
[29] T. Asfour, *et al.*, "Armar-iii: An integrated humanoid platform for sensory-motor control," in *Humanoids*, 2006, pp. 169–175.
[30] J. Lee, *et al.*, "Estimating model uncertainty of neural networks in sparse information form," in *ICML*, 2020, pp. 5702–5713.
[31] W. Burgard, *et al.*, "Estimating the absolute position of a mobile robot using position probability grids," in *AAAI*, 1996, pp. 896–901.
[32] A. Kirsch, *et al.*, "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning," *NeuRIPS*, vol. 32, 2019.