2022 International Conference on Unmanned Aircraft Systems (ICUAS)
Dubrovnik, Croatia. June 21-24, 2022

2022 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-0593-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICUAS54217.2022.9836107

Dynamic Coded Distributed Convolution for UAV-based Networked Airborne Computing

Bingnan Zhou*†, Junfei Xie*, Baoqian Wang*†
* Department of Electrical and Computer Engineering
San Diego State University, San Diego, CA, 92182
† University of California San Diego, San Diego, CA, 92093
Email: jxie4@sdsu.edu

***Abstract*—A single unmanned aerial vehicle (UAV) has limited computing resources and battery capacity, making it difficult to handle computationally intensive tasks such as the convolution operations in many deep learning applications. UAV-based networked airborne computing (NAC) is a promising technique to address this challenge. It allows UAVs within a range to share resources among each other via UAV-to-UAV communication links and carry out computation-intensive tasks in a collaborative manner. This paper investigates the vector convolution problem over the NAC architecture. A novel dynamic coded convolution strategy with privacy awareness is developed to address the unique features of UAV-based NAC, including node heterogeneity, frequently changing network typologies, time-varying communication and computation resources. Simulation results show its high efficiency and resilience to uncertain stragglers.**

I. Introduction

Recent years have witnessed the fast popularization of unmanned aerial vehicle (UAV) in both academia and industry [1]–[3]. The UAVs are often equipped with sensing, communication, and computing capabilities and can generate massive data, which include valuable information that can be used for improving decisions, making scientific discoveries, or supporting new artificial intelligence (AI) applications. To effectively utilize the information from these data, e.g., by using deep learning algorithms, considerable amount of computing resources are often needed. However, due to small payload, a single UAV often has a limited computing capability and battery capacity for carrying out computation-intensive tasks.

To address above issues, the existing solution is to offload data from the UAV to the ground station or remote cloud for processing. However, this solution suffers from many issues such as long transmission latency and data losses, and is thus not suitable for delay-sensitive applications. A better solution is to offload the data to nearby UAVs and leverage their computing resources to perform data processing and analysis. Such a computing system formed by UAVs connected via UAV-to-UAV communication links are often known as the UAV-based networked airborne computing (NAC) [4]. Compared with the traditional cloud- or static server-based computing systems, UAV-based NAC systems are featured by 1) high node mobility; 2) heterogeneous nodes with different computing, communication and sensing capabilities; and 3) dynamic computing and communication resources. These unique features make many existing distributed computing techniques that assume homogeneous and static computing nodes perform poorly in UAV-based NAC systems.

The time-varying communication and computing properties of UAV-based networked airborne computing systems can be modeled as uncertain stragglers that are slow in generating the result or take a long time to transmit data. Topology changes or link/node failures can also be modeled as uncertain stragglers that fail to generate or return any results. To alleviate the effects of stragglers, coded distributed computing (CDC) [5] is a promising technique, which introduces computation redundancy into the system via exploiting the coding theory. Currently, most works on CDC focus on the matrix multiplication problem or assume homogeneous distributed systems with static computing nodes [6]–[9]. However, many data analysis algorithms, especially the filtering or feature extraction techniques like the convolutional neural networks (CNNs), involve convolution operations. How to perform resilient distributed convolution over UAV-based NAC systems formed by heterogeneous moving UAVs has not been investigated, to the best of our knowledge. The state-of-the-art coded convolution strategy introduced in [10] was designed for homogeneous systems with static computing nodes, which performs poorly over the UAV-based NAC system as we will show in the simulation studies. Although there have been some works considering heterogeneous systems [7]–[9] and moving computing nodes [11], [12], these works are centered on the matrix multiplication problem, which has a quite different problem solving procedure from the convolution problem.

In this paper, we aim to fill the aforementioned research gap by making the following main contributions:

- *Dynamic coded distributed convolution strategy*. We

propose an innovative dynamic coded distributed convolution strategy with privacy awareness for UAV-based NAC. It integrates the coding theory with a novel task decomposing and allocation mechanism to dynamically assign tasks to the worker nodes based on their communication and computing performances. Unlike most existing CDC algorithms that have to pre-determine the amount of computation redundancy to be introduced before performing the task, our strategy introduces redundancy dynamically and only when needed. It can thus achieve high resilience with the minimal redundancy. Furthermore, as our strategy encodes the input data, data privacy is protected to some extent.

- *Comprehensive simulation studies.* We conducted comprehensive simulation studies to evaluate the performance of the proposed strategy, in comparison to the uncoded distributed convolution strategy and the state-of-the-art coded distributed convolution strategies. The results demonstrate the high efficiency and resilience of the proposed strategy in face of uncertain stragglers.

In the rest of the paper, we first describe the problem to be solved in Sec. II, and then review the two existing distributed convolution strategies in Sec. III. The proposed dynamic coded distributed convolution strategy is then introduced in Sec. IV. In Sec. V, we present the simulation results on the performance of the proposed strategy, compared to existing distributed convolution strategies. Section VI finally concludes the paper.

II. Problem Description

Consider a UAV-based NAC system formed by multiple UAVs with different computing and/or communication capabilities. Suppose one of the UAV needs to perform a vector convolution task, $\boldsymbol{a}*\boldsymbol{x}$, where $\boldsymbol{a} \in \mathbb{R}^{N_1}$ is a pre-stored vector and $\boldsymbol{x} \in \mathbb{R}^{N_2}$ is the input vector. To save energy and reduce computation time, it decides to offload the task to its neighbors within its communication range.

The problem considered in this paper is how the *master node* (UAV that offloads the task) should decompose the task and distribute subtasks to surrounding *worker nodes* (UAVs that execute the offloaded task collaboratively), such that the task completion time is minimized. To solve this problem, the key technical challenges to conquer include: 1) As all worker nodes in the UAV-based NAC system can move, the network topology may change frequently due to node leave and join, and the communication quality of UAV-to-UAV links varies over time; 2) The computing resources available at a worker node are also time variant, due to completion of old tasks or receipt of new tasks; 3) The input data $\boldsymbol{x}$ may contain sensitive information and directly sending the data to worker nodes may raise privacy concerns. The desired distributed computing scheme should thus 1) be resilient to network topology and resource changes, 2) be efficient in computing the task, and 3) protect data privacy to certain extent.

III. Review of Existing Solutions

In this section, we review two state-of-the-art distributed convolution strategies.

A. Uncoded Convolution Strategy

In the uncoded convolution strategy introduced in [10], the master node first partitions both vectors $\boldsymbol{a}$ and $\boldsymbol{x}$ evenly into a set of sub-vectors of length $s = \sqrt{\frac{N_1 N_2}{P}}$, i.e., $\{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_{\frac{N_1}{s}}\}$ and $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{\frac{N_2}{s}}\}$, where P is the total number of worker nodes. It then sends each pair of sub-vectors, $\boldsymbol{a}_i$ and $\boldsymbol{x}_j$, to a different worker node for further processing, where $1 \leq i \leq \frac{N_1}{s}$ and $1 \leq j \leq \frac{N_2}{s}$. Each worker node computes $\boldsymbol{a}_i * \boldsymbol{x}_j$ and returns the result back to the master node. After receiving results from all worker nodes, the master node finally aggregates the results with proper shifts to obtain the value of $\boldsymbol{a} * \boldsymbol{x}$.

As this strategy requires the results from all worker nodes to obtain the final value, any delay will significantly degrade its performance and any node/link failure will cause the whole task to fail. In addition, this strategy simply decomposes the workload evenly, and thus cannot address the node heterogeneity and dynamic features of UAV-based NAC. Moreover, it directly sends the input data to the worker nodes and hence may cause information leakage.

B. Traditional Coded Convolution Strategy

To improve the resilience of the uncoded strategy to the straggler effects, a coded strategy was developed in [10]. The key idea is to introduce redundancy into the computation by using the coding theory. In particular, similar to the uncoded strategy, the coded strategy first partitions both vectors $\boldsymbol{a}$ and $\boldsymbol{x}$ into small sub-vectors of equal length s. The difference is that the sub-vector length s can be any value larger than $\sqrt{\frac{N_1 N_2}{P}}$, and the $\frac{N_1}{s}$ sub-vectors of $\boldsymbol{a}$ are encoded into $\frac{Ps}{N_2}$ sub-vectors with each having a length of s, by using a $(\frac{Ps}{N_2}, \frac{N_1}{s})$ MDS code. The following equation shows how a Vandermonde matrix, denoted as $V \in \mathbb{R}^{\frac{N_1}{s} \times \frac{Ps}{N_2}}$, can be used to encode

the set of $\frac{N_1}{s}$ sub-vectors, $\{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_{\frac{N_1}{s}}\}$, into a larger set of $\frac{Ps}{N_2}$ sub-vectors, $\{\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_2, ..., \hat{\boldsymbol{a}}_{\frac{Ps}{N_2}}\}$:

$$\begin{bmatrix} \hat{\boldsymbol{a}}_1 \\ \hat{\boldsymbol{a}}_2 \\ \vdots \\ \hat{\boldsymbol{a}}_{\frac{Ps}{N_2}} \end{bmatrix} = V \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_{\frac{N_1}{s}} \end{bmatrix} = \begin{bmatrix} 1 & g_1 & g_1^2 & \dots & g_1^{\frac{N_1}{s}-1} \\ 1 & g_2 & g_2^2 & \dots & g_2^{\frac{N_1}{s}-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & g_{\frac{Ps}{N_2}} & g_{\frac{Ps}{N_2}}^2 & \dots & g_{\frac{Ps}{N_2}}^{\frac{N_1}{s}-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_{\frac{N_1}{s}} \end{bmatrix}$$

With the encoded sub-vectors $\{\hat{\boldsymbol{a}}_i\}_{i=1}^{\frac{Ps}{N_2}}$, the master node then sends each pair $(\hat{\boldsymbol{a}}_i, \boldsymbol{x}_j)$ to a different worker node, where $1 \leq i \leq \frac{Ps}{N_2}$ and $1 \leq j \leq \frac{N_2}{s}$. The worker nodes then convolve the received two sub-vectors and return the result back to the master node after the task is completed. The master node can decode $\{\boldsymbol{a}_i * \boldsymbol{x}_j | 1 \leq i \leq \frac{N_1}{s}\}$ to reconstruct $\boldsymbol{a} * \boldsymbol{x}_j$ after receiving any $\frac{N_1}{s}$ of the set $\{\hat{\boldsymbol{a}}_i * \boldsymbol{x}_j | 1 \leq i \leq \frac{Ps}{N_2}\}$ by using the following equation:

$$\begin{bmatrix} \boldsymbol{a}_1 * \boldsymbol{x}_j \\ \boldsymbol{a}_2 * \boldsymbol{x}_j \\ \vdots \\ \boldsymbol{a}_{\frac{N_1}{s}} * \boldsymbol{x}_j \end{bmatrix} = V_j^{-1} \begin{bmatrix} \hat{\boldsymbol{a}}_{i_1} * \boldsymbol{x}_j \\ \hat{\boldsymbol{a}}_{i_2} * \boldsymbol{x}_j \\ \vdots \\ \hat{\boldsymbol{a}}_{i_{\frac{N_1}{s}}} * \boldsymbol{x}_j \end{bmatrix} = \begin{bmatrix} 1 & \dots & g_{i_1}^{\frac{N_1}{s}-1} \\ 1 & \dots & g_{i_2}^{\frac{N_2}{s}-1} \\ \vdots & \ddots & \vdots \\ 1 & \dots & g_{i_{\frac{N_1}{s}}}^{\frac{N_1}{s}-1} \end{bmatrix}^{-1} \begin{bmatrix} \hat{\boldsymbol{a}}_{i_1} * \boldsymbol{x}_j \\ \hat{\boldsymbol{a}}_{i_2} * \boldsymbol{x}_j \\ \vdots \\ \hat{\boldsymbol{a}}_{i_{\frac{N_1}{s}}} * \boldsymbol{x}_j \end{bmatrix}$$

where $\{i_k\}_{k=1}^{\frac{N_1}{s}}$ represent any $\frac{N_1}{s}$ distinct indices of $\{1, 2, ..., \frac{Ps}{N_2}\}$, and V_j is a sub-matrix of V. Finally, the master node can reconstruct $\boldsymbol{a} * \boldsymbol{x}$ after obtaining $\{\boldsymbol{a} * \boldsymbol{x}_j\}_{j=1}^{\frac{N_2}{s}}$.

It should be noted that although this strategy can effectively reduce the straggler effect, it has the following limitations: 1) It cannot address the node heterogeneity and dynamic features of UAV-based NAC; 2) It is only resilient to up to $P - \frac{N_1 N_2}{s^2}$ node failures; 3) In order to achieve high resilience, the introduced computation redundancy, indicated by $s - \sqrt{\frac{N_1 N_2}{P}} > 0$, should be large; 4) It also directly sends the input data to the worker nodes and thus may cause information leakage.

IV. Dynamic Coded Convolution Strategy with Privacy Awareness

In this section, we introduce a privacy-aware dynamic coded convolution strategy that addresses the unique features of UAV-based NAC systems. How to decompose and encode the task is first explained, followed by the description of how to allocate and distribute the decomposed subtasks.

A. Task Decomposing and Encoding

Instead of partitioning both vectors, we split only the input vector $\boldsymbol{x}$ evenly into $\frac{N_2}{b}$ sub-vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{\frac{N_2}{b}}\}$, where b is the length of each sub-vector and can be any integer between 1 and N_2. Then instead of encoding $\boldsymbol{a}$, we encode the input sub-vectors into a larger set $\{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, ..., \hat{\boldsymbol{x}}_{\frac{N_2}{b}+k}\}$ by applying a $(\frac{N_2}{b} + k, \frac{N_2}{b})$ MDS code, where $k \in \mathbb{Z}^+$ specifies the computation redundancy. This will not only enhance the system resilience to uncertain stragglers, but also protect the data privacy to certain extent as the original input data is not sent. These sub-vectors $\{\hat{\boldsymbol{x}}_i\}_{i=1}^{\frac{N_2}{b}+k}$ are then pushed into a stack, denoted as $\boldsymbol{S}$, at the master node. Whenever a worker node becomes available, we pop a sub-vector $\hat{\boldsymbol{x}}_i$ from the top of stack $\boldsymbol{S}$ and send it to this worker node to compute $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$, where vector $\boldsymbol{a}$ is pre-stored in all worker nodes. Once the master node receives $\frac{N_2}{b}$ convolution results from the worker nodes, it can decode $\{\boldsymbol{a} * \boldsymbol{x}_i | 1 \leq i \leq \frac{N_2}{b}\}$, using the similar decoding procedure described in Section III-B, and thereby reconstructing $\boldsymbol{a} * \boldsymbol{x}$.

Unlike in the traditional coded convolution strategy, where the amount of computation redundancy is fixed after specifying the length s of the sub-vectors, we here base on the network condition to dynamically introduce redundancy when needed. In particular, we first set k as a small value, e.g., 1, so that the initial stack $\boldsymbol{S}$ only contains encoded input sub-vectors merely adequate enough for obtaining the final result. During task execution, whenever the stack $\boldsymbol{S}$ becomes empty (or below a certain threshold) and the master node still hasn't received sufficient results for computing the final value, the master node pushes a new $\hat{\boldsymbol{x}}_i$, generated by encoding $\{\boldsymbol{x}_i\}_{i=1}^{\frac{N_2}{b}}$, into the stack. Note that we can pre-store an encoding matrix V that is large enough at the master node, and take the first $\frac{N_2}{b} + k$ rows to initialize the stack $\boldsymbol{S}$ and take a new row whenever needed to generate new $\hat{\boldsymbol{x}}_i$ during task execution. With this scheme, we can minimize the amount of introduced computation redundancy and maximize the system resilience to uncertain stragglers simultaneously.

B. Task Allocation

To determine which worker node the master node should send the next sub-vector $\hat{\boldsymbol{x}}_i$ (popped from the stack $\boldsymbol{S}$) to and when to send this sub-vector, we borrow the idea introduced in [11]. The key idea is to send a sub-vector $\hat{\boldsymbol{x}}_i$ to each worker node at the beginning. The master node then determines the best time to send the next sub-vector to a worker node based on the estimation of the time required for this worker node to complete its current task as well as send back the result.

In particular, let $\mathcal{T}_{j,i}$ be the time interval between sending two consecutive sub-vectors, $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_{i+1}$, to the worker node j from the master node. As illustrated in Fig. 1, in order to maximally reduce the computation delay, the desired $\mathcal{T}_{j,i}$ will minimize the idle time at the worker node j while not overloading it. That is, ideally, the worker node j should receive $\hat{\boldsymbol{x}}_{i+1}$ immediately after it completes the previous task, i.e., computing $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$. To determine $\mathcal{T}_{j,i}$, the key is thus to estimate the time required for the master node to compute $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$, denoted as $T_{j,i}^{comp}$. Here, we apply the method introduced in [11] to estimate the expected time required for the worker node j to compute $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$. In particular, the expected computation time $\mathbb{E}[T_{j,i}^{comp}]$ can be estimated by following equations:

$$\mathbb{E}[T_{j,i}^{comp}] \approx \frac{t_{j,i}^{c} - t_{j,i}^{u}}{c_j} \tag{1}$$

$$t_{j,i}^{c} \approx t_{j,i}^{r} - \frac{B_r}{B_r + B_x} RTT_j \tag{2}$$

$$t_{j,i}^{u} \approx t_{j,i-1}^{u} + \max(0, RTT_j - t_{j,i-1}^{r} - t_{j,i}^{s}) \tag{3}$$

where $t_{j,i}^{c}$ is the time when the worker node j finishes computing $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$, $t_{j,i}^{r}$ is the time when the master node receives the computation result of $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$ from the worker node j, and $t_{j,i}^{s}$ is the time when the master node sends sub-vector $\hat{\boldsymbol{x}}_i$ to the worker node j. $t_{j,i}^{u}$ is the accumulated idle time of worker node j. c_j is the number of results sent back from the worker node j, B_x (bytes) is the size of the vector $\hat{\boldsymbol{x}}_i$ and B_r (bytes) is the size of the result of $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$. Lastly, RTT_j is the round trip time of sending $\hat{\boldsymbol{x}}_i$ to the worker node j and receiving the computed result, which can be estimated at the master node by exchanging Acknowledgement (ACK) packages [13] or based on the timestamps returned by the worker node j.

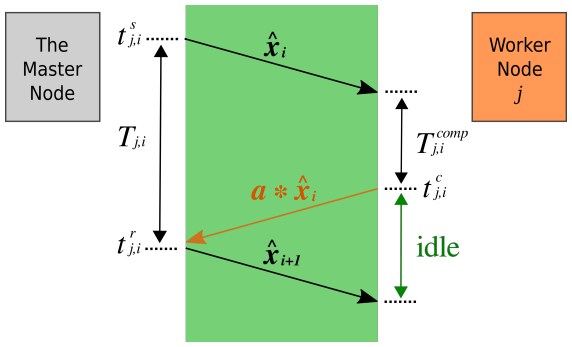

Fig. 1. The communication flow between the master node and the worker node j.

Given $\mathbb{E}[T_{j,i}^{comp}]$, we then determine $\mathcal{T}_{j,i}$ using the following equation:

$$\mathcal{T}_{j,i} = \min(t_{j,i}^{r} - t_{j,i}^{s}, \mathbb{E}[T_{j,i}^{comp}]) \tag{4}$$

Algorithm 1 summarizes the complete procedure of the proposed dynamic coded convolution strategy.

Algorithm 1: Dynamic Coded Distributed Convolution Strategy

Data: $\boldsymbol{a}$, $\boldsymbol{x}$, b, V
Result: $\boldsymbol{a} * \boldsymbol{x}$
$k \leftarrow 1$, $N_1 \leftarrow |\boldsymbol{a}|$, $N_2 \leftarrow |\boldsymbol{x}|$;
Partition $\boldsymbol{x}$ into a set of sub-vectors $\{\boldsymbol{x}_i\}_{i=1}^{\frac{N_2}{b}}$ with each of length b;
Use the first $\frac{N_2}{b} + k$ rows of V to encode $\{\boldsymbol{x}_i\}_{i=1}^{\frac{N_2}{b}}$ into a larger set $\{\hat{\boldsymbol{x}}_i\}_{i=1}^{\frac{N_2}{b}+k}$, and push them into the stack $\boldsymbol{S}$;
$\mathcal{R} \leftarrow$ empty stack for storing received results;
$\mathcal{P} \leftarrow$ list of worker nodes within master node's communication range;
for *each node* j *in* $\mathcal{P}$ **do**
  Send $\hat{\boldsymbol{x}}_i$ popped from $\boldsymbol{S}$ to node j;
  $t_{j,i}^{s} \leftarrow current\_time()$;
end
while $|\mathcal{R}| < \frac{N_2}{b}$ **do**
  for *each node* j *in* $\mathcal{P}$ **do**
    if $|\boldsymbol{S}| \leq 1$ **then**
      $k \leftarrow k + 1$;
      Use the k-th row in V to generate $\hat{\boldsymbol{x}}_{\frac{N_2}{b}+k}$ and then push it into $\boldsymbol{S}$;
    end
    if $current\_time() \geq t_{j,i}^{s} + \mathcal{T}_{j,i}$ **then**
      Send $\hat{\boldsymbol{x}}_{i+1}$ popped from $\boldsymbol{S}$ to node j;
      $t_{j,i+1}^{s} \leftarrow current\_time()$;
    end
    if *receiving result of* $\boldsymbol{a} * \hat{\boldsymbol{x}}_i$ *from node* j **then**
      Push the result into $\mathcal{R}$;
      Update $\mathcal{T}_{j,i}$ using (1)-(4);
    end
  end
end
Reconstructs $\boldsymbol{a} * \boldsymbol{x}$ using results from $\mathcal{R}$;

V. Simulation Studies

In this section, we conduct simulations to evaluate the performance of the proposed strategy, in comparison with the uncoded convolution and traditional coded convolution schemes. All simulations are performed on a PC with 16GB of RAM and Intel Core i5-4590.

A. System Models

We use the following system models to simulate the movement of UAVs, as well as how they compute and how they communicate with each other.

1) Mobility Model: A simple 2-dimensional (2D) point-mass mobility model is adopted to simulate the movement of each UAV. In particular, let $\boldsymbol{p}_j(t)$ denote the location of UAV j at time t. Then its location at the next time point t' is given by the following equation:

$$\boldsymbol{p}_j(t') = \boldsymbol{p}_j(t) + \boldsymbol{v}_j(t)(t' - t)$$

where $\boldsymbol{v}_j(t)$ is the velocity of UAV j at time t, where $t' \geq t$.

2) Computing Model: To simulate the time required by each UAV j to convolve two vectors, say $\boldsymbol{e}_1 \in \mathbb{R}^{n_1}$ and $\boldsymbol{e}_2 \in \mathbb{R}^{n_1}$, we adopt the following shifted exponential distribution model commonly used in the literature [7]:

$$\Pr[T_j^{comp}(\boldsymbol{e}_1, \boldsymbol{e}_2) \leq t] = 1 - e^{\frac{\mu_j}{c(n_1,n_2)}(t - \alpha_j c(n_1,n_2))}$$

where $T_j^{comp}(\boldsymbol{e}_1, \boldsymbol{e}_2)$ is the time taken by UAV j to compute $\boldsymbol{e}_1 * \boldsymbol{e}_2$. $\alpha_j > 0$ and $\mu_j > 0$ are shift and straggling parameters, respectively, which characterize the computing power of UAV j. $c(n_1, n_2)$ represents the computation load required for computing $\boldsymbol{e}_1 * \boldsymbol{e}_2$. To derive this value, we assume that the Fast Fourier Transform (FFT) is used by each UAV to calculate the convolution of two vectors with arbitrary lengths. Hence, we have [14], [15]:

$$\begin{aligned} c(n_1, n_2) &= \mathcal{O}\big((n_1 + n_2 - 1)(\log(n_1 + n_2 - 1) + 1)\big) \\ &= C(n_1 + n_2)\log(n_1 + n_2) \end{aligned}$$

where C is a constant independent of the lengths of the vectors.

3) Communication Model: We assume that the communication between any two UAV nodes is achieved through a directional antenna, and the antennas are always aligned during the movement [16], [17]. The communication time required for the master node to transmit to (or receive from) a worker node j a dataset containing n numbers at time t can be approximated by the following equation [18]:

$$T_j^{comm}(n, t) = \frac{n \times u}{R_j(t)}$$

where u is the average size of the numbers in the dataset. $R_j(t)$ is the data rate (bits/sec) given by:

$$R_j(t) = B \log_2(1 + \frac{10^{\frac{S_d(t) - 30}{10}}}{N_0})$$

where B (Hz) is the communication bandwidth between the master node and worker node j and N_0 is the noise power (W), both of which are assumed to be constant. $S_d(t) = P_t + 20log_{10}(\lambda) - 20log_{10}(4\pi) - 20log_{10}(d(t)) + G_{l|dBi} + w$ is the signal power (W) that depends on the distance $d(t)$ between the master node and the worker node j at time t. P_t (dBm) is the transmitting power, $G_{l|dBi}$ is the sum of the transmitting and receiving gains, w is the Gaussian noise and λ is the wave length.

B. Experiment Setup

We consider the following four computation scenarios:

- **Scenario 1:** $N_1 = 2^{12}$, $N_2 = 2^{11}$, $P = 8$.
- **Scenario 2:** $N_1 = 2^{12}$, $N_2 = 2^{11}$, $P = 4$.
- **Scenario 3:** $N_1 = 20000$, $N_2 = 30000$, $P = 8$.
- **Scenario 4:** $N_1 = 20000$, $N_2 = 30000$, $P = 6$.

In all scenarios, the straggling parameter μ_j in the computation model is randomly sampled from the range $[3 \times 10^6, 6 \times 10^6]$, and the shift parameter α_j is set to $\alpha_j = \frac{1}{\mu_j}$. To simulate the straggler effect, we consider two cases: 1) stragglers caused by long communication latency and/or computation delay; and 2) stragglers caused by system failures or moving out of the master node's communication range. To model the first case, we manually make the run-time of the stragglers to be 15 times the simulated run-time returned from its computation model. To model the second case, we make the stragglers stop returning any results to the master node.

For the configuration of the mobility model, the initial position of each UAV is randomly sampled from the range $[(-1500, -1500), (1500, 1500)]$. The velocity of each UAV is randomly sampled from the range $[(-10, -10), (10, 10)]$ m/s once every second. Lastly, the parameters in the communication model are configured as $B = 10^6$, $N_0 = 10^{-12}$, and $S_d(t) = 6 - 20log_{10}(d(t))$. It is worthy of remark that our method does not require any knowledge of the mobility, computation or communication models.

C. Simulation Results

1) Impact of Parameter b: We first study the impact of the key parameter in our method, i.e., the length of the input sub-vectors b, by evaluating the performance of our method at different values of b. To reduce uncertainty, each experiment in our simulation study is repeated for 25 times and the mean execution times are recorded. As shown in Fig. 2, in all four scenarios, the execution time of our method first decreases as b increases, and

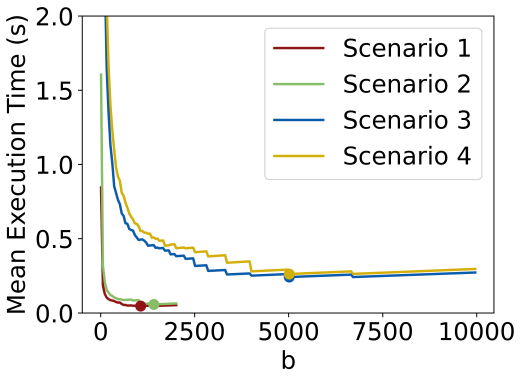

Fig. 2. Impact of parameter b on the performance of the proposed strategy.

then increases after b reaches a certain value. The best b is thus the one that leads to the minimum mean execution time, which varies in different scenarios. Fig. 2 also reveals that with the increase of the problem size (characterized by N_1 and N_2) or the decrease of the amount of computing resources available (indicated by P), the time required for conducting the computation task increases.

In subsequent experiments, we use the best-performing b to configure our method in each scenario, which are marked with big dots in Fig. 2. For the choice of s, length of the sub-vectors in the traditional coded convolution strategy, we follow the selection guideline provided in [10] and set s as the integer from range $[\sqrt{\frac{N_1 N_2}{P}}, \min(N_1, N2)]$ that maximizes $|\epsilon(s)|$, where $\epsilon(s)$ is given by:

$$\epsilon(s) = -\sum_{j=1}^{P} \frac{(\frac{Ps}{N_2} - \frac{N_1}{s} + 1)\mu_j^{\alpha_j}}{P((2Cs)\log(2s))^{\alpha_j}}$$

2) Comparison Studies: We first study the case when stragglers with long communication/computation delay are present. Fig. 3 compares the performance of different strategies when 50% of the worker nodes randomly selected are such stragglers. It shows that our method achieves the highest efficiency in all scenarios and the uncoded convolution strategy is the least efficient.

To better understand the three strategies, we further conduct a stress test by varying the percentages of the stragglers with long delays. Fig. 4 shows the results of the stress test for different strategies in Scenario 4. As we can see, the performance of all three strategies degrade with the increase of the straggler ratio and our method achieves the best performance in all cases. It can also be observed that the uncoded strategy is the most sensitive to stragglers, as indicated by the immediate increase of its execution time when the straggler ratio becomes non-zero. Nevertheless, the execution time of our method does not increase much until the straggler ratio exceeds around 83%, demonstrating its high resilience to uncertain stragglers.

Lastly, we investigate the case when node failures or node leaves can happen. As the master node cannot receive any results from such stragglers, the results received from other worker nodes may not be sufficient enough for the master node to reconstruct the convolution $\boldsymbol{a} * \boldsymbol{x}$, leading to task failures. Therefore, in this study, we measure the task success rate (ratio of successful runs) of each strategy at the presence of such type of stragglers. Fig. 5 shows the success rates of different strategies, where each strategy runs 2000 times in each scenario. In each simulation run, 0 up to P worker nodes can fail. The result demonstrates that our method is highly resilient to node failures. It is worthy noting that our method can successfully complete the task as long as there is a worker node alive, which can be the master node itself. Moreover, even if the master node loses connection with all worker nodes, as long as there is a new node joining later, the task will resume.

VI. Conclusion

This paper introduces an efficient, resilient, and privacy-aware distributed computing strategy for vector convolution tasks in heterogeneous and mobile UAV-based NAC systems. It combines the coding theory with a novel task decomposing and allocation mechanism to achieve a high resilience to uncertain stragglers with the minimal computation redundancy. As input data is encoded, it also provides some protection for data privacy. The simulation results show that the proposed strategy outperforms existing solutions in both efficiency and resilience, especially when a large number of high-latency computing nodes are present or frequent node

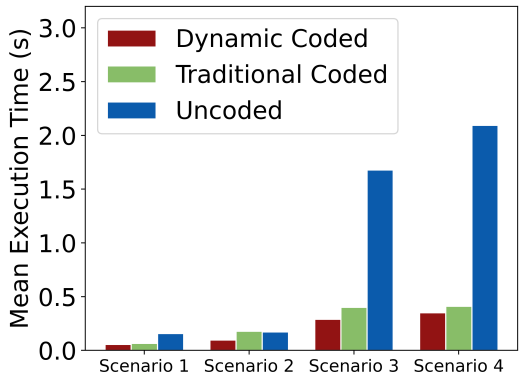

Fig. 3. Comparison of different strategies when 50% of the worker nodes are stragglers with long communication/computation delays.

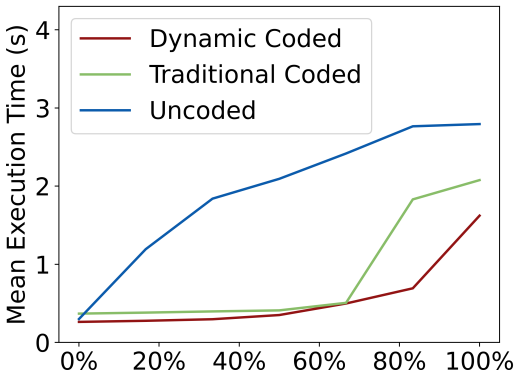

Fig. 4. Comparison of different strategies in Scenario 4 when the straggler ratio increases.

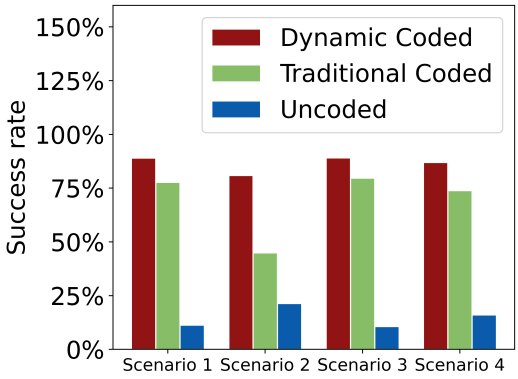

Fig. 5. The task success rate of different strategies when node failures/leaves can happen.

leaves/failures happen. Moreover, our method is adaptive to the dynamic network changes in UAV-based NAC systems and can complete the task as long as there is a worker node alive, which can be the master node itself.

In the future, we will design intelligent strategies to automate the configuration for the key parameter b, and extend the proposed strategy to real CNN-based applications. We will also develop hardware testbed for UAV-based NAC and conduct flight tests to evaluate the performance of the proposed strategy.

ACKNOWLEDGMENT

We would like to thank the National Science Foundation (NSF) under Grants CI-1953048 and CAREER-2048266 for the support of this work.

REFERENCES

[1] E. Honkavaara, H. Saari, J. Kaivosoja, I. Pölönen, T. Hakala, P. Litkey, J. Mäkynen, and L. Pesonen, "Processing and assessment of spectrometric, stereoscopic imagery collected using a lightweight uav spectral camera for precision agriculture," *Remote Sensing*, vol. 5, no. 10, pp. 5006–5039, 2013.

[2] K. Choi, I. Lee, J. Hong, T. Oh, and S. W. Shin, "Developing a uav-based rapid mapping system for emergency response," in *Unmanned Systems Technology XI*, vol. 7332. SPIE, 2009, pp. 75–86.

[3] Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard, "Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works," *Visualization in Engineering*, vol. 4, no. 1, pp. 1–8, 2016.

[4] K. Lu, J. Xie, Y. Wan, and S. Fu, "Toward uav-based airborne computing," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 172–179, 2019.

[5] J. S. Ng, W. Y. B. Lim, N. C. Luong, Z. Xiong, A. Asheralieva, D. Niyato, C. Leung, and C. Miao, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Communications Surveys & Tutorials*, 2021.

[6] K. Lee, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Coded computation for multicore setups," in *Proc. of IEEE ISIT 2017*, 2017, pp. 2413–2417.

[7] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "On batch-processing based coded computing for heterogeneous distributed computing systems," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2438–2454, 2021.

[8] ——, "Coding for heterogeneous uav-based networked airborne computing," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[9] A. Reisizadeh, S. Prakash, R. Pedarsani, and S. Avestimehr, "Coded computation over heterogeneous clusters," in *Proc. of IEEE ISIT 2017*, 2017.

[10] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2403–2407.

[11] Y. Keshtkarjahromi, Y. Xing, and H. Seferoglu, "Dynamic heterogeneity-aware coded cooperative computation at the edge," in *Proc. of ICNP 2018*. IEEE, sep 2018, pp. 23–33.

[12] B. Wang, J. Xie, K. Lu, Y. Wan, and S. Fu, "Multi-agent reinforcement learning based coded computation for mobile ad hoc computing," *arXiv preprint arXiv:2104.07539*, 2021.

[13] P. Sessini and A. Mahanti, "Observations on round-trip times of tcp connections," *Simulation Series*, vol. 38, no. 3, p. 347, 2006.

[14] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The fast fourier transform and its applications," *IEEE Transactions on Education*, vol. 12, no. 1, pp. 27–34, 1969.

[15] R. N. Bracewell, *The Fourier transform and its applications, 3rd Ed*. McGraw-Hill New York, 1999.

[16] J. Chen, J. Xie, Y. Gu, S. Li, S. Fu, Y. Wan, and K. Lu, "Long-range and broadband aerial communication using directional antennas (acda): Design and implementation," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 793–10 805, 2017.

[17] S. Li, C. He, M. Liu, Y. Wan, Y. Gu, J. Xie, S. Fu, and K. Lu, "Design and implementation of aerial communication using directional antennas: learning control in unknown communication environments," *IET Control Theory & Applications*, vol. 13, no. 17, pp. 2906–2916, 2019.

[18] M. Liu, Y. Wan, S. Li, F. L. Lewis, and S. Fu, "Learning and uncertainty-exploited directional antenna control for robust long-distance and broad-band aerial communication," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 593–606, 2019.