2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Reinforcement Learning-Guided Channel Selection Across Time for Multivariate Time Series Classification

Leonardos Pantiskas
Vrije Universiteit Amsterdam
0000-0002-4898-5334

Kees Verstoep
Vrije Universiteit Amsterdam
0000-0001-6402-2928

Mark Hoogendoorn
Vrije Universiteit Amsterdam
0000-0003-3356-3574

Henri Bal
Vrije Universiteit Amsterdam
0000-0001-9827-4461

***Abstract*—The promising results of machine learning in time series classification, along with the rise in sensor data-driven use cases, have led to the increasing deployment of models in IoT environments, on edge devices. Since these devices are typically resource constrained, they cannot always execute large and complex models, so they often offload (part of) their tasks to remotely located models. This synergy however introduces the need to transfer a large amount of sensor data to the cloud, which can be detrimental to bandwidth cost and inference speed of the application, and energy utilization of the device. Although techniques such as early classification can limit the data that has to be transferred, there are still unexplored opportunities when it comes to input filtering. A recent versatile early-exit framework, extending early classification and adapting it to multivariate time series, has investigated this potential. In this work, we propose a variation of this method, creating a more flexible, reinforcement learning-enabled framework that can adapt the input variables (channels) considered for classification across time, aiming for maximizing accuracy while minimizing the input data necessary. Extensive testing on synthetic data and real datasets shows that our method can, in multiple cases, achieve better accuracy for a similar percentage of input filtering, both compared to the baseline framework, as well as to the conventional early classification approach.**

***Index Terms*—multivariate, time series, classification, reinforcement learning, input sampling, edge intelligence**

I. Introduction

Nowadays, machine and deep learning methods have demonstrated increasingly recognized achievements in applications such as image classification and time series analysis. Together with the widespread adoption of sensor-enabled systems in Internet of Things environments, this has led to an emergence of use cases where these models are deployed on smaller, portable platforms that are in the immediate vicinity of the data-generating sources. This paradigm is known as *edge intelligence* [1], [2]. A characteristic of these edge systems is that they are often more resource-limited compared to traditional computational platforms. Therefore, in the edge intelligence realm, there is often a shift in the importance of metrics that were previously not considered, such as the energy consumption of devices and sensors, or the computations required for a task, while metrics such as classification accuracy acquire a proportionally adjusted, lower importance [1].

Due to the usually limited resources of the edge devices, some tasks often require cloud-edge collaboration [3]. In this setup, a remote, more complex model supports the local, usually smaller, edge model, or there is even complete offloading of the data to the cloud, potentially after some prepossessing on the edge device. Although this synergy can improve the classification result, it introduces some constraints to the edge intelligence application, such as the bandwidth required to transfer data from the edge environment to the cloud, and additional latency for the classification.

Since this issue is evident with the image and video data inputs, due to the larger volume of data for even a single sample, some works in this field have already tried to achieve data efficiency, by filtering the input and offloading to the cloud only part of it [4]–[6]. For time series data, a traditional approach that can be translated to data efficiency in the same context is early classification [7], where a classifier makes a prediction only after observing a few early timesteps. The main goal of early classification, as the name suggests, has been to decrease the classification time, rather than limit the data transfer. However, as the IoT use cases employ increasingly more data-generating sources, the data volume issue is manifesting with time series data as well [8].

In a recent work [9], we introduced the CHARLEE framework, placed in an edge-cloud synergy context with the clear goal of achieving data efficiency by extending the early classification paradigm specifically to the multivariate time series data, i.e. data with multiple variables (channels) as input. In this work, we aim to take this effort further, address the limitations of CHARLEE, and create a framework that can filter the input data across time and channels in a more versatile manner before they are transferred to the cloud.

Thus, we propose RELEVANT, for **Re**inforcement **L**earning-**E**nabled **V**ariable **A**daptation i**N** **T**ime, a framework adapted to the multivariate time series modality specifically, inspired by the concepts of video analytics data efficiency methods. Our framework operates on a per-sample basis, flexibly excluding and including channels across time in order to maximize the classification accuracy while minimizing the amount of input data used, obviating the need for offloading

This work has been conducted as part of the Just in Time Maintenance project funded by the European Fund for Regional Development.

it from edge to cloud.

An illustrative example highlighting the value of our framework can be observed in the following use case within an Industry 4.0 setting [8]: An industrial hydraulic pump can be fitted with sensors measuring variables such as pressure, water flow, temperature, and so forth, in order to monitor it for faults. In a given time series sample, which may correspond to data from a work cycle of the pump, it may become apparent within a few time steps that a fault exists. However, the precise nature of the fault may become evident in a later part of the cycle, and with information based only on the pressure and temperature variables, with the intermediate timesteps not being useful for classification. In this scenario, a conventional early classification approach, treating the input channels in a unified manner, would be unable to terminate the inference at an early stage without sacrificing accuracy, thus processing the data from all channels unnecessarily, potentially sending them to the cloud, until the later part of the sample. On the other hand, CHARLEE would be able to abandon some channels at the beginning, but it would not be able to skip the intermediate steps of the pressure and temperature variables, similarly processing all their data until the end of the sample. In contrast, RELEVANT would allow both early termination of uninformative channels and skipping of the uninformative steps in the middle of the useful variables, only focusing on their part that is necessary for classification.

Our framework differs from other works that focus on specific parts of the input, utilizing e.g. attention mechanisms [10] or shapelet methods [11], in the aspect that these approaches do not translate to data efficiency. Although some parts of the input sample may be more valuable and attended to for the classification task, the whole sample still needs to be processed, which would entail transferring it to the cloud in an edge-cloud synergy context. In contrast to that, we have designed our framework with this explicit scenario in mind, constructing it to completely skip the unnecessary input parts, based on its gathered context. By placing it on an edge device, it can achieve data efficiency both in terms of bandwidth and energy savings in the sensors, since a crucial factor in sensor networks is the energy consumed to collect and transmit the data [12], [13]. Revisiting the example we presented above, the edge device running RELEVANT could instruct all but the necessary sensors (pressure and temperature) to stop gathering and/or sending data, conserving data collection, radio transmission, and data processing energy, as well as bandwidth.

Our contributions with RELEVANT are:

- We introduce the concept of spatio-temporal input filtering for multivariate time series classification, inspired by similar approaches in video data research.
- We formulate this concept as a Partially Observable Markov Decision Process and adapt a reinforcement learning approach presented in a previous work [9] to address it.
- We verify the behavior of our framework on a synthetic dataset and 26 real datasets. We demonstrate its capability and we also compare it against a different reinforcement learning-based framework [9], as well as a simplistic, time-only input filtering solution, showcasing its ability to surpass these alternatives in multiple cases.

II. Related Work

A. Spatial- and Temporal-wise Input Filtering

In the extensive survey on dynamic neural networks of Han et al. [10] we note that there are multiple approaches for video and text data that focus on a subset of the input data, using a variety of techniques, either in the spatial dimension (for image data), temporal dimension, or both. Regarding the spatial dimension, which applies to image data, some approaches perform computation either in selected pixel-level (e.g. [14]) or region-level (e.g. [15]) locations of the image, thus achieving computational efficiency by skipping a part of the input. When it comes to the temporal dimension, there are works that can learn to skip specific words when processing text data, such as LSTM-Jump [16], which reads the first few tokens and then, based on a hidden state, ”jumps” ahead, skipping words. In video data, this concept translates to skipping frames when processing a video, such as the architecture in [17] that decides which frames to observe and when to emit a prediction.

In both modalities there are works that incorporate early stopping, classifying the data after observing a small early part of the tokens or frames. Early stopping has also been widely researched for time series data, with numerous approaches and applications of early classification [7].

In the video processing field, some works try to combine spatial and temporal input selection, with a characteristic example being the AdaFocus+ method [18]. This method simultaneously skips uninformative frames, while also focusing on specific patches of the frames that are not skipped.

RELEVANT transfers the above concepts of combined spatial and temporal input selection, together with early stopping, to the data format of multivariate time series classification. We adapt the concepts presented in some related works, such as sequential decision-making, partially-observable Markov Decision Process formulation, and end-to-end training [16], [17], [19] to the unique needs and format of this task.

B. Reinforcement Learning-based Early Stopping

There are multiple early classification solutions that can be regarded as input filtering, due to not observing the full time series sample before stopping and emitting a prediction [7]. Some works that specifically utilize reinforcement learning methods are [20] and [21], where the authors encode the goals of earliness (i.e. input filtering) and accuracy in the reward function of a deep Q-network [22] agent. The Stop&Hop framework introduced in [23] uses two policies, a Stopping policy for deciding whether to stop the input processing and classify the sample and a Hopping policy that decides whether to ”jump” to future values of the sample. Although all the above solutions could translate to input filtering, they do not explicitly address the variables dimension of the multivariate time series format.

As we discussed, the CHARLEE framework [9] extends the early classification paradigm in both dimensions of multivariate time series, early exiting both across the time and channel dimensions. This is made possible with the introduction of specific constraints, i.e. ordering of the channels in terms of usefulness for the classification and monotonic decrease of the number of channels across time. Although CHARLEE achieves its intended flexibility, it still suffers from some limitations of the early classification approach. For example, if a channel has useful patterns in its initial and final timesteps, but not in the intermediate ones, CHARLEE is forced to either keep it included until the last timestep, thus reducing the achieved input savings, or it can abandon it early on, but with a negative impact on accuracy.

With RELEVANT, we aim to redefine the multidimensional early classification paradigm and shift it to even more versatile multidimensional input filtering. Drawing parallels between the dimensions of video data (time,image patches) and the multivariate time series (time, variables), we identify an opportunity for developing a solution for the latter, similar to AdaFocus+ mentioned above. We want our solution to be able to skip uninformative time slices and exclude/re-include variables across time flexibly during inference, without necessarily abandoning channels in an irreversible manner, or keeping them needlessly until the point where they have useful information. Thus, we remove the CHARLEE constraints, we re-design the reinforcement learning reward and we change the variable selection mechanism to serve the implementation of this new paradigm.

III. Proposed framework

A. Time Slicing and Channel Grouping

In order to achieve the multidimensional early classification paradigm, we have introduced a number of heuristics in [9]. Two of those heuristics are processing the input in groups of timesteps, called time slices, and grouping the channels of the input. These heuristics help to 1) reduce the computational burden of the framework (since it gets executed after every several timesteps instead of every timestep) and 2) reduce the search space of the reinforcement learning agent, since the decision to early exit or not can be taken for multiple channels of a single group together. We find these heuristics valid and useful for the RELEVANT framework as well, so we keep them in our design.

B. POMDP Formulation and Framework Structure

Following the CHARLEE framework design in [9], as well as related work [19], [21], [23], [24], we express the task of adapting the variables across time as a Partially Observable Markov Decision Process (POMDP), defined by a tuple $\{\mathbf{S,A,T,R,}\gamma\}$, where **S** is the state space, **A** is the action space, **T** is the transition model, **R** is the reward function and γ is the discount factor of rewards. In Fig. 1, we can see the structure and pipeline of the RELEVANT framework. The framework functions as follows: First, it receives a group of timesteps, termed a time slice, from the raw input. When the processing begins, this slice always includes all channels. The hidden state encoder, constructs or updates the states $\in \mathbf{S}$, utilizing a 1-d convolutional network and features extracted from it. The hidden state also includes positional information and the history of the policy decisions. Both for the creation of the hidden state, as well as the data propagated to the final classifier, a "filtered" channel means that its raw values are replaced with a mask value (in our case 0).

Based on this state, the policy either decides to stop the processing or select a subset of channels (including none) for the next time slice, an action designated $\alpha = \alpha_{stop} \cup \alpha_{adapt}, \in \mathbf{A}$. The adaptation (selection) policy is implemented as a Bernoulli distribution for each channel (or group of channels) parametrized by a fully-connected network (FCN), resulting in the action vector α_{adapt}. The result of a channel inclusion/exclusion depends on the stochastic outcome of this distribution. The stopping policy is implemented as an FCN with a sigmoid output.

Our transition model **T**, akin to related work [21], is as follows: The action α either leads to a new state, $s' \in \mathbf{S}$, created by the hidden state encoder from the appropriately filtered next slice, or to a terminal state, either because the input is over or the policy performs a stopping action α_{stop}. When the processing episode terminates, all filtered time slices are sent to the final model for classification. Similar to [9], we make RELEVANT classifier-agnostic, and we test it with InceptionTime [25] and ResNet [26], as characteristic examples of classification models.

Based on the classification result and the percentage of input saved, a reward R_{final} is generated for all the policy actions throughout the episode, so $\mathbf{R}(s, \alpha) = R_{final}$ for all states s in the episode. We want to encode the amount of input saved in the reward [9], [20], [21], [27], but contrary to CHARLEE, we want to push the default network behavior towards saving as much input as possible without sacrificing accuracy, if possible. Thus, we formulate the reward as:

$$R_{final} = \begin{cases} 1 + R_{savings}, & \text{if correct classification} \\ -2, & \text{otherwise} \end{cases} \quad (1)$$

In Eq. 1, $R_{savings}$ is a number in the range $[0, 1]$, resulting from the linear mapping of the input savings achieved in the range from minimum savings (no savings) to maximum (only utilization of first time slice). We set the discount factor γ of the rewards to 1, since, for our problem formulation, we want all filtering actions across time to contribute equally to the result.

C. Training and Inference Process

We train all parts of the framework by minimizing the sum of their respective losses. We utilize the typical **cross-entropy loss** L_{acc} for the classifier model, as well as an auxiliary loss L_{aux} only for the incorrectly classified samples. This L_{aux} is the **cross-entropy loss** between the sample label and the unfiltered raw input data and is meant to help the classifier "see" the full data, even when the policy has excluded multiple

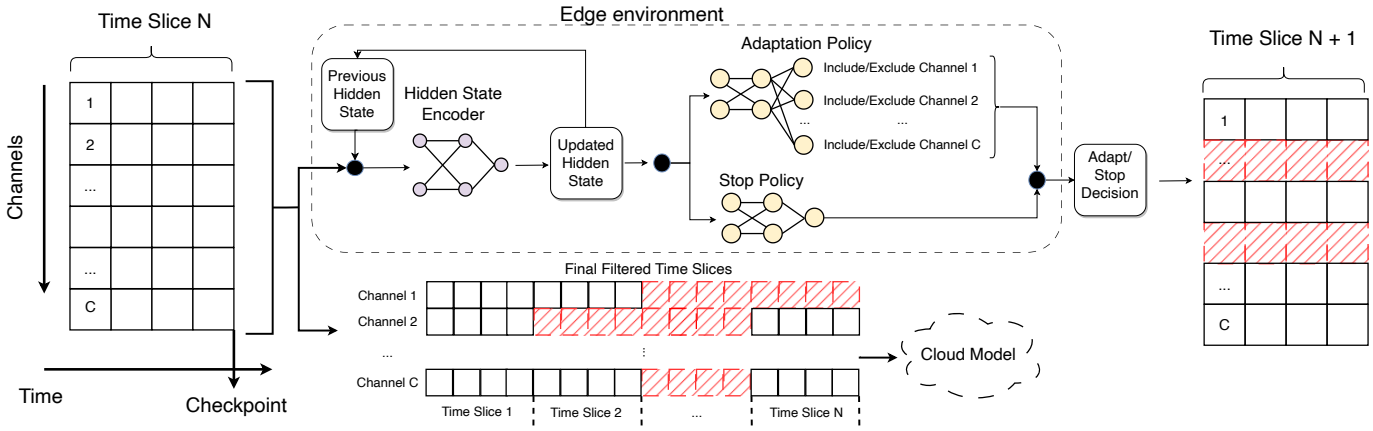

Fig. 1. Structure of RELEVANT framework (adapted from [9])

channels, so that future classification is better informed. The adaptation policy includes non-differentiable sampling from the Bernoulli distributions, so we use the REINFORCE algorithm [28] to estimate the gradients to update its parameters, with the loss being [29]:

$$L_{adapt} = -\mathbb{E}\Big[\sum_{n=0}^{N-1}\sum_{c=0}^{C-1} log\pi_{\theta_a}(A_{c_n}|S_n)\Big(R - b(S_n)\Big)\Big] \quad (2)$$

In Eq. 2, N corresponds to the number of checkpoints and C to the number of channels (or channel groups). A_{c_n} is the channel inclusion/exclusion action for channel **c** at checkpoint **n** by the policy π_{θ_a}. R is the final reward and S_n is the state created by the hidden state encoder network at checkpoint **n**. From the reward R we subtract a baseline value term $b(S_n)$, in order to reduce the variance of the gradient estimation. This value is estimated by a baseline network that has the same input and structure as the adaptation policy and is trained by minimizing the **mean squared error loss** $L_{baseline}$ between the value and the reward.

The stop policy is trained differently. We calculate a reward R_{stop}, similar to R, for each checkpoint, based on the classification result and the input savings up to it. If this R_{stop} is higher at a checkpoint than all the future ones, it means the framework should stop at that, since it could not obtain better results by continuing. The stop policy can then be trained by minimizing the **binary cross-entropy loss** L_{stop} between this binary criterion (stop or not) and the action of the policy. By using this method, we can train the stop policy without actually stopping episodes during training. This benefits the channel adaptation policy by avoiding the issue of the framework not observing the later parts of the time series due to excessive stopping [21], [23]. Thus, the total loss of the framework is:

$$L_{total} = L_{acc} + L_{aux} + L_{adapt} + L_{baseline} + L_{stop} \quad (3)$$

IV. Experiments

In order to explore the performance and behavior of RELEVANT, we run extensive experiments: Firstly, we compare its classification accuracy to that of CHARLEE, on both synthetic and real data, for similar input percentage utilization.

Although RELEVANT is designed to maintain accuracy while filtering the input, different dataset patterns may not be suitable for that, and the framework may converge to a result that sacrifices some of the accuracy in favor of input savings. In addition, its novel approach to spatiotemporal input filtering and its design details, such as the classifier-agnostic endpoint, make it impractical to meaningfully compare with other seemingly associated methods, such as early classification ones. Thus, we use the same comparison approach as multiple related works [9], [21], [30] and for the real datasets we compare the accuracy of RELEVANT against deep learning baselines trained on a fixed percentage of the input. For example, if RELEVANT reaches average input savings of 30% across runs, a point of reference would be the deep learning classifier trained at the dataset truncated at 70% of its timestep length. We refer to this method as Time-Only Input Filtering (TOIF).

A. Datasets

1) Synthetic dataset: We utilize the synthetic dataset introduced in [9], which has been constructed so that parts of the input can be skipped without affecting classification accuracy. It is comprised of 4 channels and 96 timesteps, with signal information split across 4 slices (so segments of 24 timesteps). Below we describe the parts of the dataset that are necessary for the correct classification of the pairs of 8 classes:

- Classes 1,2: This pair of classes can be correctly classified utilizing the information of all channels only in the 1st slice, so no processing is necessary beyond that point.
- Classes 3,4: This pair of classes can already be distinguished from all others from the 1st slice. However, the 4th slice of the 4th channel is necessary to differentiate between the two classes of the pair.
- Classes 5,6: This pair of classes can be distinguished from all others based on the information in the 1st slice and on the information of any single channel in the 2nd slice.

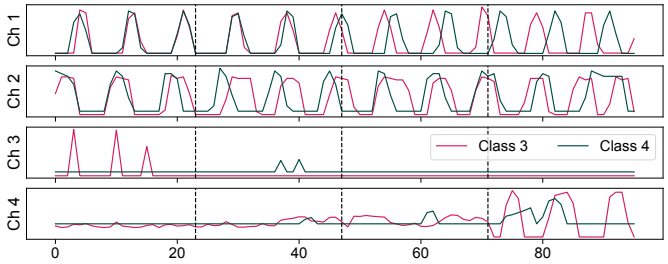

Fig. 2. Samples of classes 3 and 4

Moreover, the 4th slices of the 3rd and 4th channels are necessary to differentiate between the classes in the pair.

- Classes 7,8: Similar to the pair above, this pair of classes can be distinguished from all others based on the information in the 1st slice and on the information of any channel in the 2nd slice. In addition, all channels in the 4th slice are necessary to differentiate the two classes.

In Fig. 2 we can see a sample of the pair of classes 3 and 4, and the segmentation points that have been used for the dataset construction, to better understand the synthetic data logic. This example is characteristic for the applicability of RELEVANT, since it is demonstrated that the intermediate slices do not have useful information and could be completely skipped, while only the 4th channel is useful in the 4th slice for the classification task.

2) Real datasets: We utilize the same set of 26 datasets as [9], comprised of a subset of the UEA collection [31] and the CWRU [32] and MAFAULDA [33] datasets, so that we can easily compare the performance of RELEVANT with the CHARLEE framework. We also process the datasets in the same manner: For each dataset, we train the base deep learning model on 5,10,15,...,95% of its length and get its test set accuracy. If at any percentage this accuracy is similar to that of the DL model trained on the full length, we truncate the dataset to that timestep percentage. In this way, we try to ensure that the input savings are mainly attributed to our framework, rather than trivial early classification due to the dataset properties. With this large collection of datasets, we can more safely ascertain the performance of RELEVANT, and establish the use cases where it is successful and the ones where more exploration or fine-tuning is necessary.

B. Experimental Setup

All experiments were run on the DAS-6 infrastructure [34], on nodes with 24-core AMD EPYC-2 (Rome) 7402P CPUs, NVIDIA A6000 GPUs, and 128 GB of RAM. We implement the framework using PyTorch [35], and we use the implementations of the tsai package [36] for InceptionTime and ResNet. We repeat all experiments 5 times with different random seeds, and we present the averages of the metrics. We use 20% of each training set as a validation set, and we use the weights that result in the best validation score for testing.

We have employed a uniform number of checkpoints (4) across all datasets, and the number of channel groups has been set to the minimum value between the number of channels in each dataset and 10. Although these fixed hyperparameters may not yield optimal results for all datasets, they aim to provide a general evaluation of the performance of RELEVANT. For practitioners, hyperparameter fine-tuning based on specific datasets is likely to improve the accuracy-input savings result. The code for the experiments and the detailed metrics are publicly available at https://github.com/lpphd/RELEVANT to facilitate reproducibility and future work on the topic.

TABLE I
F1 ACHIEVED BY RELEVANT, CHARLEE AND TIME-ONLY INPUT FILTERING FOR SIMILAR INPUT SAVINGS

	RELEVANT		CHARLEE		TOIF	
	F1	Average Savings	F1	Average Savings	F1	Average Savings
IncTime	0.895	0.506	0.797	0.519	0.618	0.519
ResNet	0.918	0.502	0.780	0.534	0.632	0.534

V. RESULTS

A. Synthetic Dataset

In Table I we can examine the F1 score and input savings achieved by RELEVANT, as well as the F1 achieved by CHARLEE and the Time-Only Input Filtering method for approximately similar input savings. These results confirm that the design of RELEVANT and its added flexibility compared to CHARLEE is beneficial to the classification task, as RELEVANT can select to skip channels for the intermediate slices and include them again in the final slice, when they contain useful information. In contrast to that, CHARLEE performs irrevocable early exiting for channels, so it either has to keep the useful channels throughout the time slices, lowering the input savings, or it can abandon them but with a degradation in accuracy. Finally, as expected, both solutions perform better than Time-Only Input Filtering, which offers the least flexibility, following the conventional early classification paradigm, and misses the information contained in the second half of the dataset.

Although the average F1 achieved by RELEVANT is $\sim$0.9 for both models, in some runs the framework manages to achieve almost perfect accuracy ($\sim$0.97) and very high input savings ($\sim$0.56). We can study the patterns learned by the framework in those runs for all test samples and we note that 4 patterns stand out, appearing much more often than the others. We can express these patterns as a list, with each element representing the set of channels kept at the corresponding time slice. The first pattern is [{1,2,3,4},{},{},{}] and the second is [{1,2,3,4},{},{},{4}]. These patterns perfectly match the first two pairs of classes of the synthetic dataset. RELEVANT immediately stops processing the input for classes 1,2, creating the first pattern, similar to early classification. For classes 3 and 4, it decides to skip the intermediate slices and focus on the 4th slice of the 4th channel. The next two patterns are [{1,2,3,4},{2},{},{1,2,3,4}] and [{1,2,3,4},{2},{2},{1,2,3,4}]. The former corresponds to

classes 5 and 6, and we observe that the only deviation from the ground truth is the inclusion of all channels in the last slice, while only 2 out of 4 are required. The pattern for classes 7 and 8 is even more precise, with the only deviation being the inclusion of the 2nd channel in the 3rd slice, where it is not necessary. Although not perfect, these patterns also capture the underlying synthetic design with high precision.

B. Real Datasets

In Table II there is an overview of the number of datasets where RELEVANT is performing better than Time-Only Input Filtering, and the ones where Time-Only Input Filtering is the preferred choice. We consider a method to be the better option if its F1 score is more than 0.01 (1 percentage point) higher than the alternative. In Table III we have the detailed F1 scores, with the bold font indicating the preferred approach for the specific dataset and model. These results have been sorted from best to worst on the InceptionTime column.

We observe that in multiple datasets, the channel adaptation across time that RELEVANT offers achieves better F1 compared to the established time-only early classification paradigm for equal input savings, while in other cases, the latter is preferred. This is expected, since our universally-fixed hyperparameters and the unique data formats make it unlikely that RELEVANT will be preferable to TOIF in all cases, especially across such an extended collection of datasets. However, the number of positive cases demonstrates the potential of RELEVANT to real-world applications.

Similarly to the synthetic dataset, the average F1 and input savings for a dataset can sometimes be considerably worse than individual runs. For instance, for the CharacterTrajectories (CHAR) dataset, RELEVANT reaches an average F1 score of 0.886, for input savings of 54.6%. However, in one of the runs it has achieved F1 score of 0.93, with input savings of 58.3%, which would make it preferable to the TOIF approach. Although we have used the average metrics for the sake of fairness, this observation indicates that a practitioner following the common practice of model selection based on cross-validation can achieve even better results with RELEVANT.

In Table IV we see the comparison of RELEVANT with CHARLEE. Since CHARLEE entails the savings factor hyper-parameter, which can affect the final F1 - input savings result, for each dataset we use the CHARLEE result which closely matches the input saving achieved by RELEVANT. If this is not possible, we choose the CHARLEE result with similar F1 to that of RELEVANT, and we compare their savings. In the results table, we highlight the cases where one solution is performing distinctly better than the other, with a difference of more than 0.05 either in F1 or input savings. We observe that the additional flexibility of RELEVANT is indeed beneficial to the F1 - input utilization pair for multiple datasets. Although we present only the InceptionTime results, for the sake of space, the ResNet results point to similar conclusions. We can also observe some cases where CHARLEE is performing better. This can be a result of sub-optimal convergence of RELEVANT due to the higher search space, which could be addressed with different policy gradient methods. Moreover, CHARLEE takes into account the utility of each channel for the classification, which may lead to better results for some datasets, even with the reduced adaptability.

TABLE II
NUMBER OF DATASETS WHERE RELEVANT IS PREFERABLE OVER TOIF WITH EQUAL SAVINGS, OR VICE VERSA

InceptionTime			ResNet		
RELEVANT	TOIF	Ties	RELEVANT	TOIF	Ties
10	7	9	9	9	8

TABLE III
F1 ACHIEVED BY RELEVANT COMPARED TO TOIF APPROACH WITH EQUAL INPUT SAVINGS

Dataset	InceptionTime		ResNet	
	RELEVANT	TOIF	RELEVANT	TOIF
NTPS	**0.773**	0.439	**0.761**	0.404
FD	**0.633**	0.55	**0.606**	0.537
SAD	**0.898**	0.828	**0.905**	0.808
BM	**0.651**	0.607	**0.663**	0.586
PS	**0.26**	0.228	**0.279**	0.198
MIND	**0.398**	0.376	0.363	0.372
HB	**0.696**	0.675	0.692	0.686
DDG	**0.517**	0.501	0.475	0.474
MAF	**0.923**	0.911	**0.947**	0.934
MHAR	**0.933**	0.921	0.884	**0.919**
ER	0.695	0.686	**0.657**	0.597
EP	0.876	0.868	**0.875**	0.861
EMOP	0.676	0.674	0.655	**0.71**
HMD	0.297	0.295	**0.233**	0.211
SRS1	0.77	0.769	0.737	**0.769**
JV	0.912	0.914	0.902	0.908
LSST	0.44	0.445	0.495	0.492
RS	0.766	0.771	0.771	0.768
EW	0.614	0.623	0.563	**0.624**
HW	0.281	**0.317**	0.209	**0.308**
CHAR	0.866	**0.907**	0.927	0.921
CWRU	0.915	**0.967**	0.865	0.864
CR	0.888	**0.941**	0.823	**0.913**
UW	0.76	**0.828**	0.65	**0.681**
AWR	0.808	**0.891**	0.795	**0.857**
LIB	0.742	**0.831**	0.782	**0.842**

VI. CONCLUSION

In this work, we have presented RELEVANT, a framework for reinforcement learning-enabled variable adaptation in time for the task of multivariate time series classification. Our framework processes the input progressively in time slices and after each one, it decides which channels (if any) it should keep for the next time slice, or if it could even stop receiving the input completely. At the end of this procedure, the filtered input is passed to a deep learning model for classification. The goal of the framework is to maximize accuracy while minimizing the amount of input data necessary for the classification. It improves upon the CHARLEE framework [9] by offering more flexibility to the input filtering task and thus greater input savings potential.

TABLE IV
F1 ACHIEVED BY RELEVANT COMPARED TO CHARLEE WITH SIMILAR INPUT SAVINGS, USING INCEPTIONTIME

Dataset	RELEVANT		CHARLEE	
	F1	Average Savings	F1	Average Savings
AWR	0.808	0.595	0.798	**0.716**
BM	**0.651**	0.639	0.531	0.642
CHAR	**0.866**	0.546	0.755	0.547
CR	**0.888**	0.697	0.802	0.705
DDG	**0.517**	0.517	0.483	0.489
EMOP	0.676	0.778	0.666	0.789
ER	**0.695**	0.756	0.573	0.764
EW	0.614	0.441	0.629	**0.575**
EP	0.876	0.668	0.865	0.696
FD	**0.633**	0.552	0.550	0.584
HMD	0.297	0.422	0.256	0.469
HW	0.281	0.489	0.252	0.565
HB	0.696	0.538	0.674	0.583
JV	**0.912**	0.654	0.729	0.699
LSST	0.440	0.448	0.450	**0.620**
LIB	0.742	0.206	0.752	0.196
MIND	0.398	0.769	0.389	0.800
MHAR	0.933	0.718	0.916	0.720
NTPS	0.773	0.525	0.784	0.499
PS	0.260	0.422	**0.233**	0.666
RS	0.766	0.502	0.738	0.548
SRS1	**0.770**	0.595	0.709	0.611
SAD	**0.898**	0.713	0.820	0.737
UW	0.760	0.399	**0.763**	0.336
CWRU	0.915	0.615	0.944	0.586
MAF	0.923	0.800	0.930	0.800

We verify the behavior of our framework on the synthetic dataset introduced in [9] and we test it on several multivariate datasets. We first demonstrate that the higher flexibility of RELEVANT compared to CHARLEE does indeed translate to a better ratio of accuracy over input utilized, for both the synthetic and multiple of the real-world datasets. In addition, we show that RELEVANT can capture the underlying synthetic dataset patterns with high precision. We also compare the accuracy of RELEVANT to deep learning classifiers trained on time-only truncated versions of real-world datasets. The datasets are truncated so that the input utilization is equal to the one achieved by RELEVANT. We show that, although RELEVANT is not the optimal choice in all cases, in multiple datasets it is beneficial to utilize it over time-only input truncation, which would correspond to time-only early classification. Given that we utilize a standard set of hyperparameters over all datasets, there is also potential for improvement after use-case specific hyperparameter tuning.

Interesting future research would be to utilize data imputation methods as part of the framework, so the filtered values can be replaced in a more informed manner than a universal mask value. This could improve both classification accuracy and potentially increase input savings. Moreover, a more complex grouping method could be used to cluster channels of similar patterns per time slice together, so that there is more efficient utilization or filtering of these channel groups.

REFERENCES

[1] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[3] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[4] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. Dulloor, "Scaling video analytics on constrained edge nodes," in *Proceedings of Machine Learning and Systems*, 2019.

[5] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 159–173.

[6] V. Nigade, L. Wang, and H. E. Bal, "Clownfish: Edge and cloud symbiosis for video stream analytics," in *5th IEEE/ACM Symposium on Edge Computing, SEC*, 2020, pp. 55–69.

[7] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, "Approaches and Applications of Early Classification of Time Series: A Review," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 47–61, 2020.

[8] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, 2018.

[9] L. Pantiskas, K. Verstoep, M. Hoogendoorn, and H. Bal, "Multivariate time series early classification across channel and time dimensions," *arXiv preprint arXiv:2306.14606*, 2023.

[10] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, 2021.

[11] G. He, Y. Duan, R. Peng, X. Jing, T. Qian, and L. Wang, "Early classification on multivariate time series," *Neurocomputing*, vol. 149, pp. 777–787, 2015.

[12] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri, "Energy management in wireless sensor networks with energy-hungry sensors," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 2, pp. 16–23, 2009.

[13] T. Kannan and H. Hoffmann, "Budget RNNs: Multi-Capacity Neural Networks to Improve In-Sensor Inference Under Energy Budgets," in *27th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, 2021, pp. 143–156.

[14] Z. Xie, Z. Zhang, X. Zhu, G. Huang, and S. Lin, "Spatially adaptive inference with stochastic feature sampling and interpolation," in *ECCV*, 2020.

[15] J.-B. Cordonnier, A. Mahendran, A. Dosovitskiy, D. Weissenborn, J. Uszkoreit, and T. Unterthiner, "Differentiable patch selection for image recognition," in *CVPR*, 2021.

[16] A. W. Yu, H. Lee, and Q. V. Le, "Learning to skim text," in *ACL*, 2017.

[17] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *CVPR*, 2016.

[18] Y. Wang, Z. Chen, H. Jiang, S. Song, Y. Han, and G. Huang, "Adaptive focus for efficient video recognition," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[19] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[20] C. Martinez, G. Perrin, E. Ramasso, and M. Rombaut, "A deep reinforcement learning approach for early classification of time series," in *26th EUSIPCO*. IEEE, 2018, pp. 2030–2034.

[21] C. Martinez, E. Ramasso, G. Perrin, and M. Rombaut, "Adaptive early classification of temporal sequences using deep reinforcement learning," *Knowledge-Based Systems*, vol. 190, p. 105290, 2020.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[23] T. Hartvigsen, W. Gerych, J. Thadajarassiri, X. Kong, and E. Rundensteiner, "Stop&Hop: Early Classification of Irregular Time Series," in *Proceedings of the 31st ACM CIKM*, 2022, pp. 696–705.

[24] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner, "Recurrent halting chain for early multi-label classification," in *Proceedings of the 26th ACM SIGKDD KDD*, 2020, pp. 1382–1392.
[25] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.
[26] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 IJCNN*. IEEE, 2017, pp. 1578–1585.
[27] U. Mori, A. Mendiburu, S. Dasgupta, and J. A. Lozano, "Early classification of time series by simultaneously optimizing the accuracy and earliness," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4569–4578, 2017.
[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Reinforcement learning*, pp. 5–32, 1992.
[29] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
[30] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner, "Adaptive-halting policy network for early classification," in *Proceedings of the 25th ACM SIGKDD KDD*, 2019, p. 101–110.
[31] A. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The UEA multivariate time series classification archive, 2018," *arXiv:1811.00075 [cs, stat]*, 2018.
[32] "Case Western Reserve University Bearing Data Center," https://engineering.case.edu/bearingdatacenter.
[33] "Machinery Fault Database [Online]," http://www02.smt.ufrj.br/˜offshore/mfs/page_01.html.
[34] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term," *Computer*, vol. 49, no. 5, pp. 54–63, 2016.
[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[36] I. Oguiza, "tsai - a state-of-the-art deep learning library for time series and sequential data," Github, 2022. [Online]. Available: https://github.com/timeseriesAI/tsai