Adacket: ADAptive Convolution KErnel Transform for Multivariate Time Series Classification

Junru Zhang[1], Lang Feng[1], Haowen Zhang[2], Yuhan Wu[1], Yabo Dong[1,*]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[2]College of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China

Tue 19 Sep 23

The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases

Background

Multivariate time series classification (MTSC) has wide applications across domains with diverse signal sources.

- Human activity recognition
- Health monitoring
- Remote sensing
- …

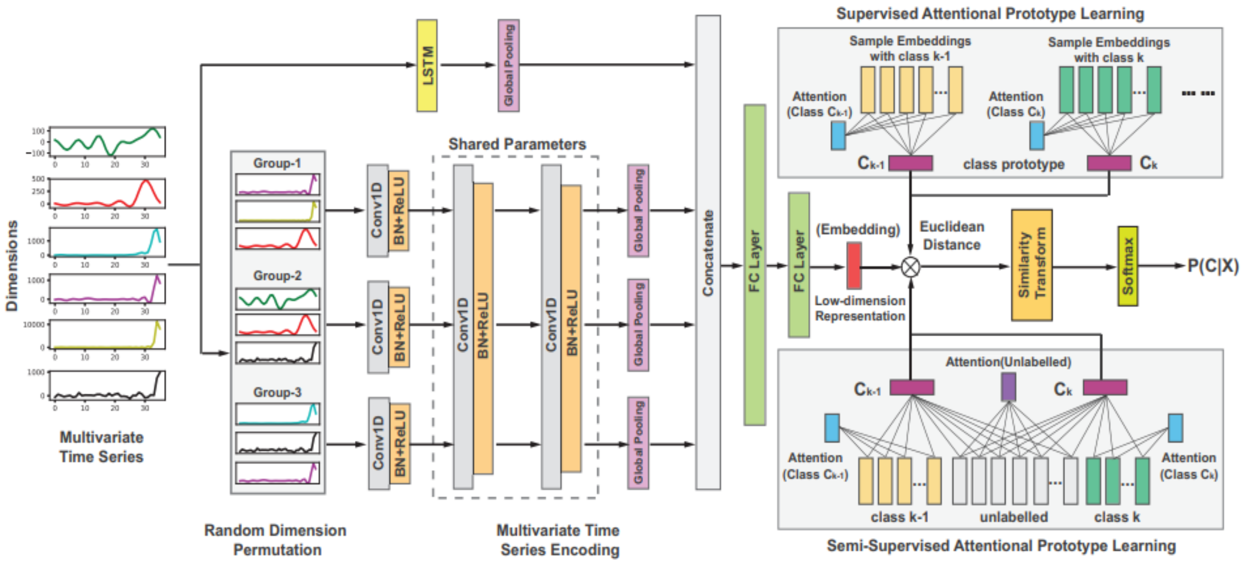
1D convolutional kernels show superiority in MTSC tasks.

- 1D-CNNs (e.g., FCN, ResNet, **InceptionTime**…)
- **ROCKET**
- …

Background

Convolution-based methods

- Produce time series of various temporal scale
- Existing methods for time series classification rely on the empirical design of massive convolutional kernels to achieve high accuracy.

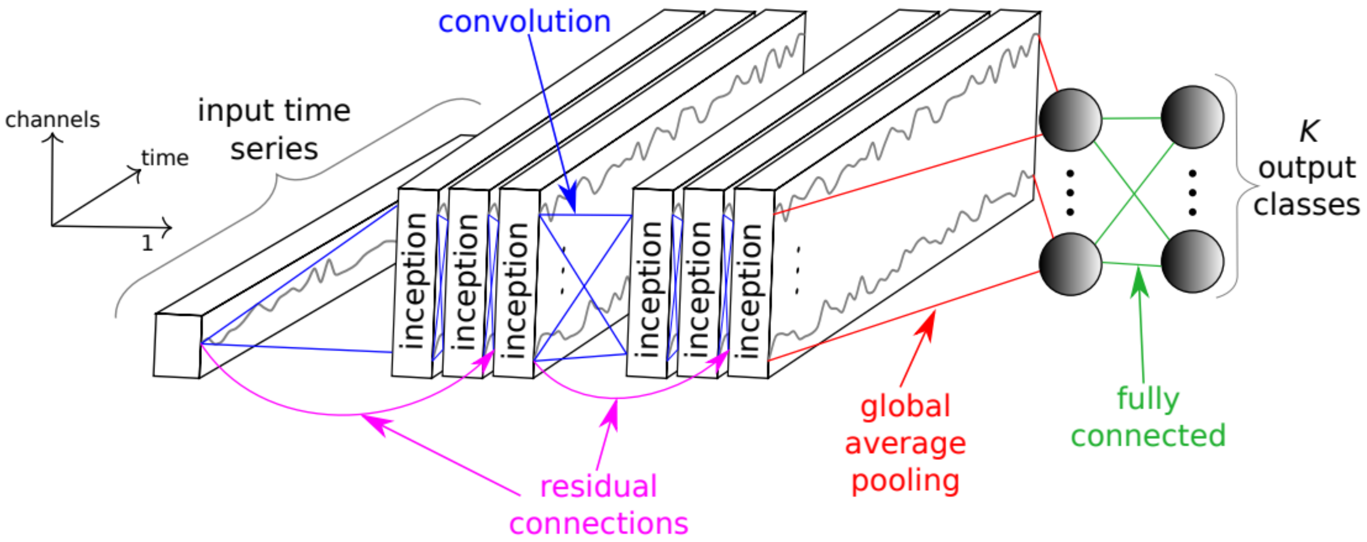


TapNet Architecture (Image source: Zhang X et al. 2020)

InceptionTime

The Inception module uses multiple convolutional kernels of *different size.*

The network architecture is complex and computationally demanding.



InceptionTime Architecture (Image source:
Hassan Ismail Fawaz et al. 2020)

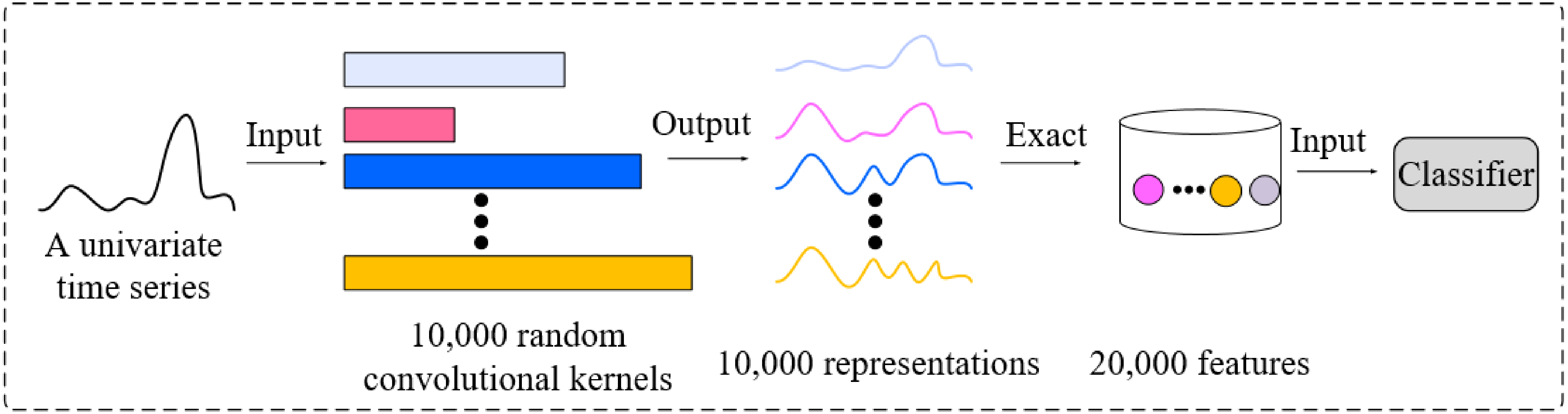
ROCKET *Fast and accurate*

Initially designed for univariate data

Generate 10,000 *random* 1D convolution kernels to transform time series

Without training the kernels

Extract 20,000 features from each transformed sequence



ROCKET (A. Dempster et al. 2020) Architecture

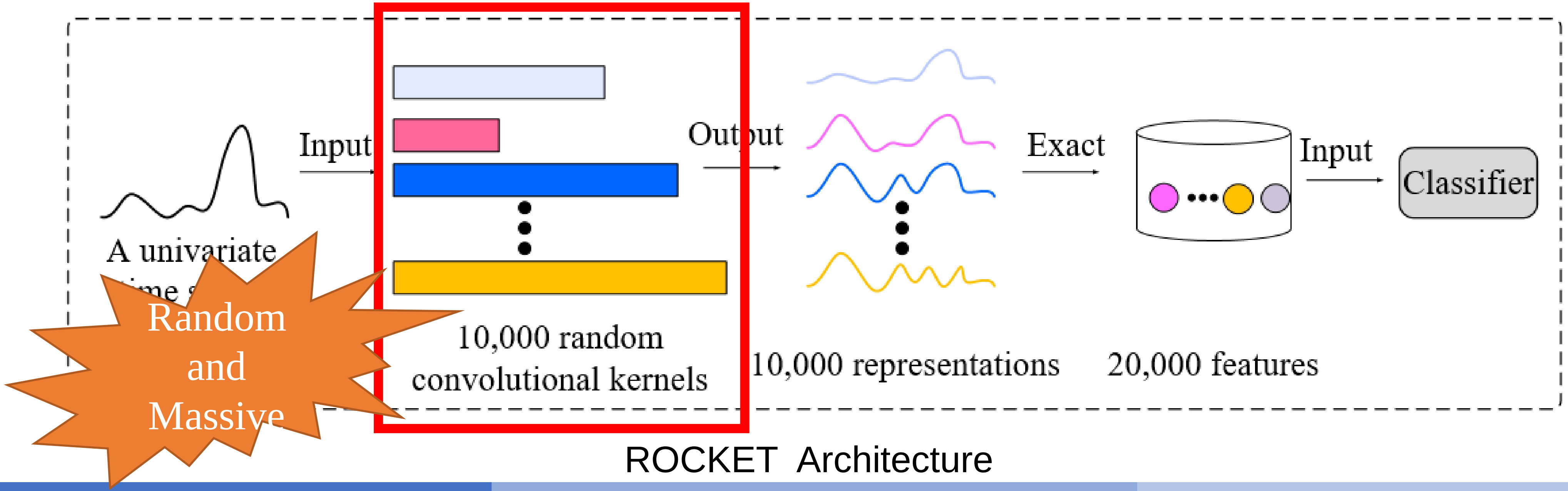
ROCKET *Fast and accurate*

Initially designed for univariate data

Generate 10,000 *random* 1D convolution kernels to transform time series

Without training the kernels

Extract 20,000 features from each transformed sequence

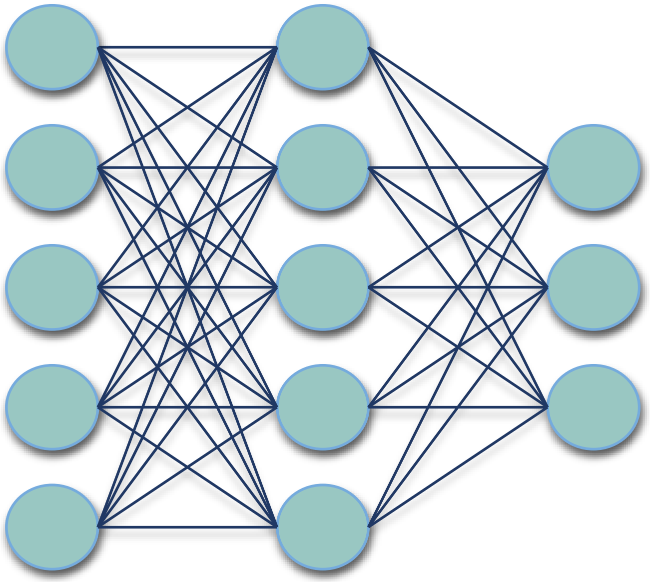


ROCKET Architecture

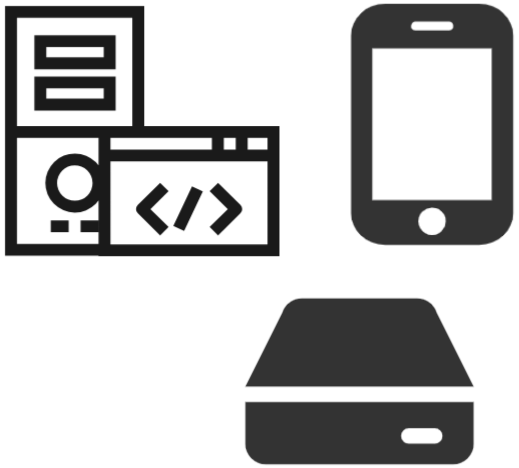
Background

Problem with exisiting convolution-based methods

As the complexity and number of time series data increase, they becomes resource-intensive to learn and store the parameters.

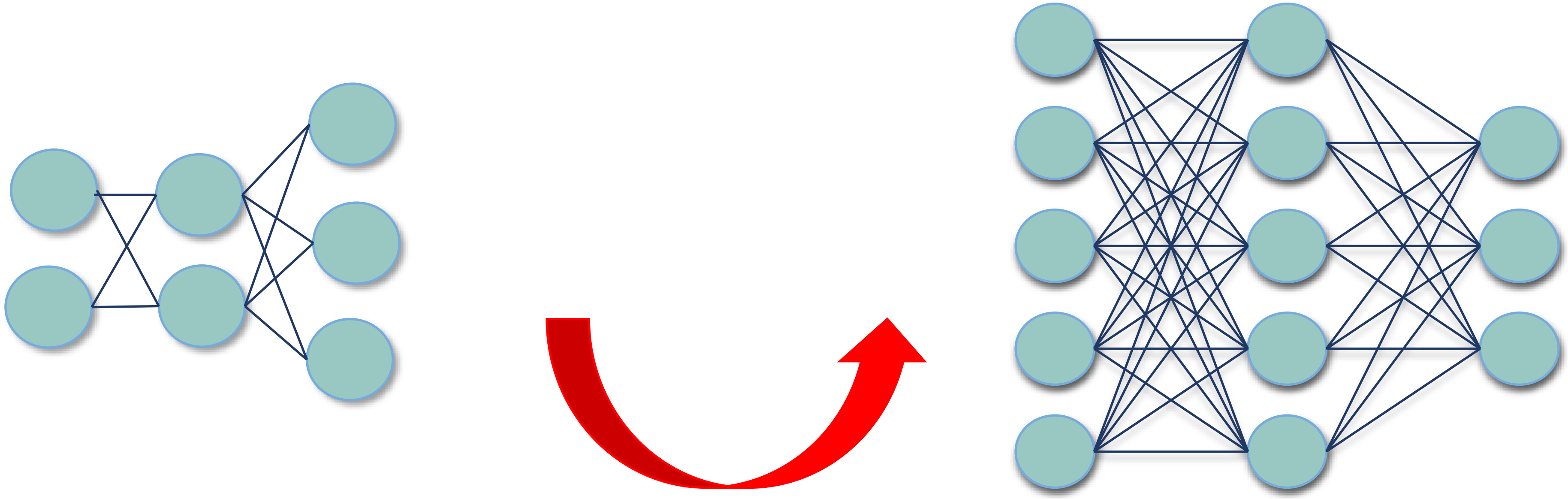


Challenging to deploy



InceptionTime

- ✕ Curse of Dimensionality
- ✕ Computational Bottleneck



Higher-dimensional

multivariate time series

ROCKET

Memory Cost: $N \times 20{,}000 \times 8\ bytes$

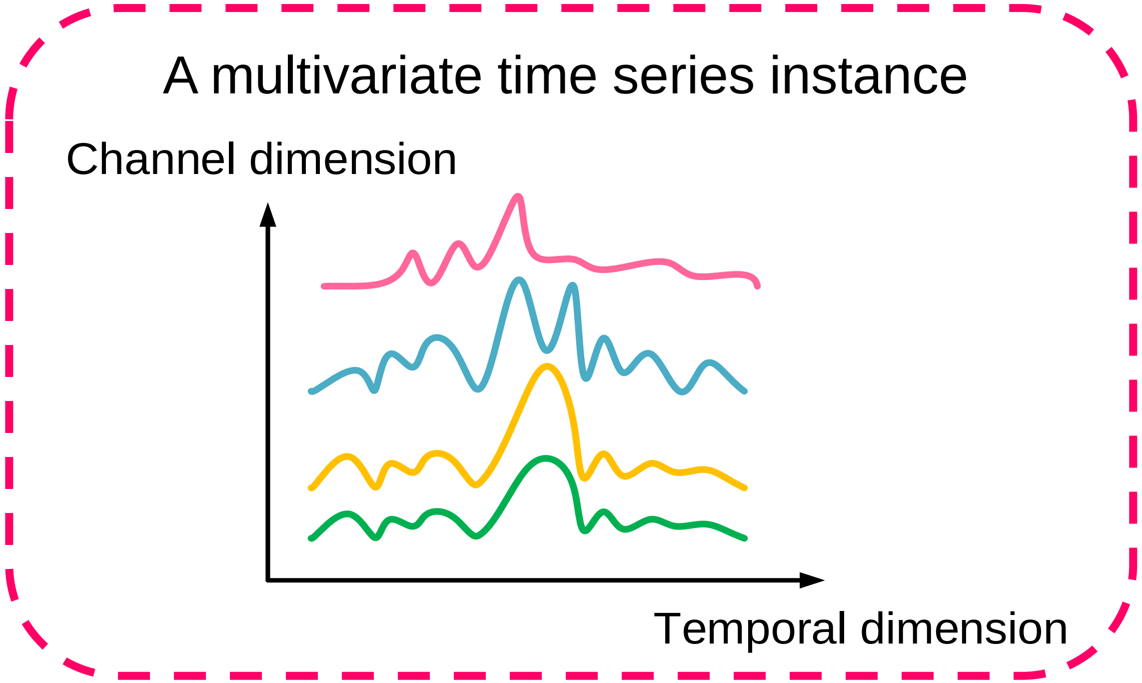
✕ Explosive growth in the number of instances *N*

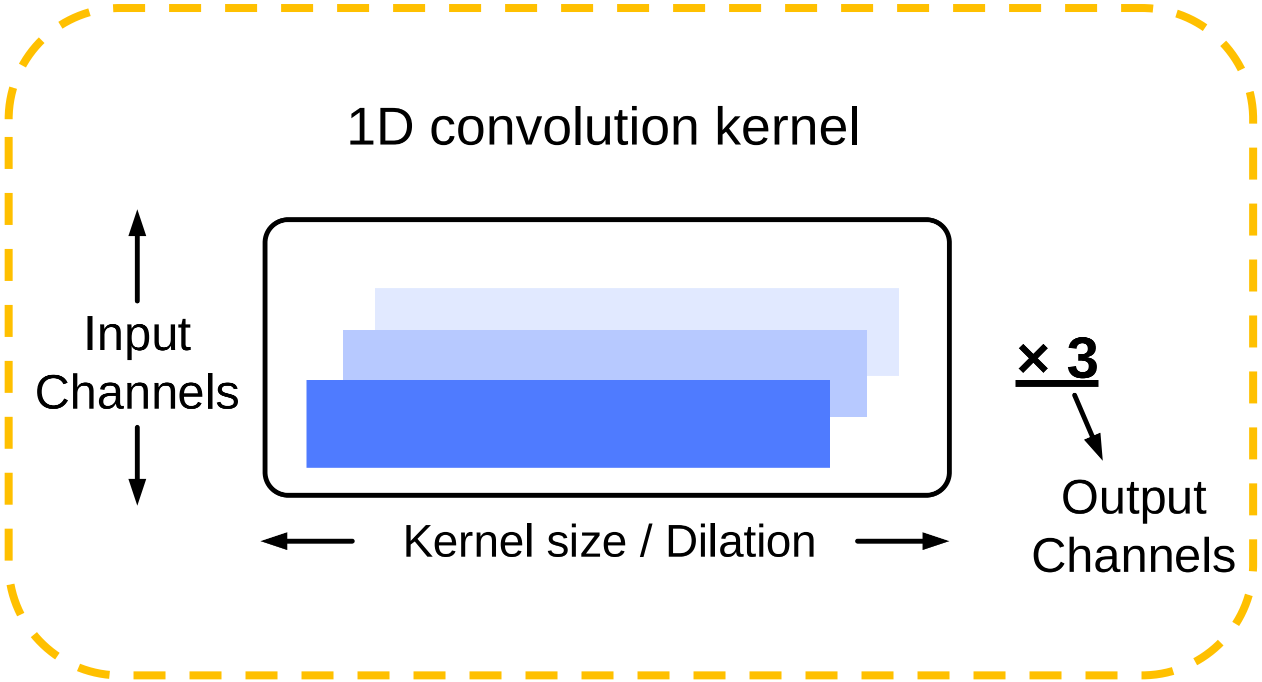
Linear Scaling of Memory Costs with Increasing Instance Size

Background

Investigate various *hyperparameters* of the convolutional kernel

- Channel dimension (Input and output channels)
- Temporal dimension (Kernel size and dilation)

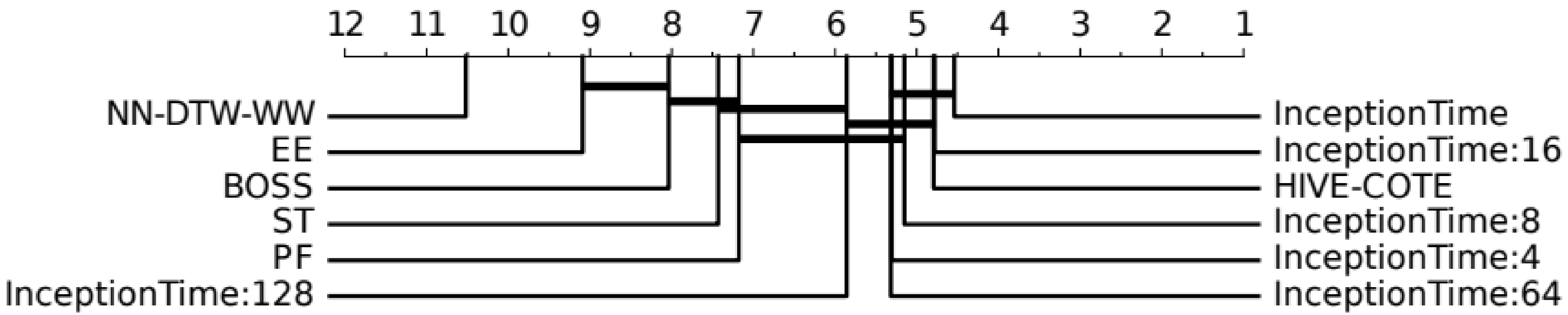




Motivation

Limitations in Hyperparameter Optimization Approaches

✕ The expensive overhead: Computational costs of trial and error in complex hyperparameter tuning



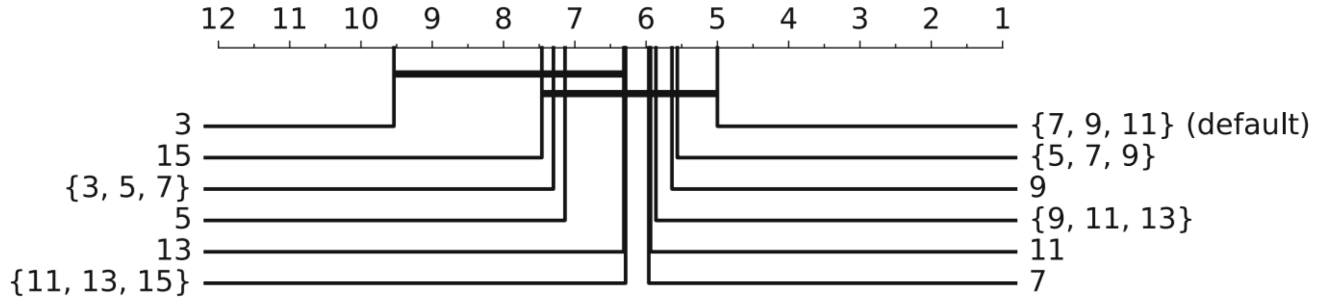
InceptionTime's search for the optimal configuration through trying various **output channels** settings (Image source: Hassan Ismail Fawaz et al. 2020)

Motivation

Limitations in Hyperparameter Optimization Approaches

✕ The expensive overhead: Computational costs of trial and error in complex hyperparameter tuning

✕ Restricted hyperparameter investigation: Neglecting comprehensive evaluation of its impact on model performance



ROCKET's search for the optimal configuration through trying various **kernel size** settings (Image source: A. Dempster et al. 2020)

Motivation

Limitations in Hyperparameter Optimization Approaches

✕ The expensive overhead: Computational costs of trial and error in complex hyperparameter tuning

✕ Restricted hyperparameter investigation: Neglecting comprehensive evaluation of its impact on model performance

Goal: Create resource-efficient convolution kernels!

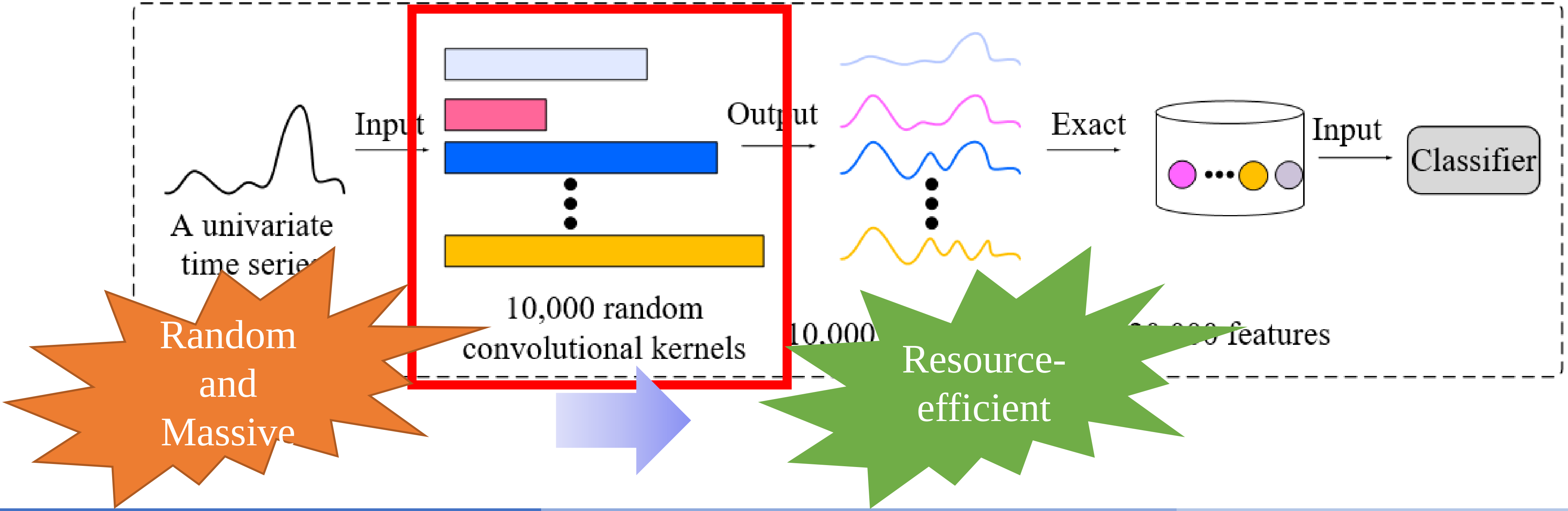
✓ Automatically explore the comprehensive design space of hyperparameters, rather than relying on random convolution kernels.

Motivation

Goal: Create resource-efficient convolution kernels!

✓ Automatically explore the comprehensive design space of hyperparameters, rather than relying on random convolution kernels.

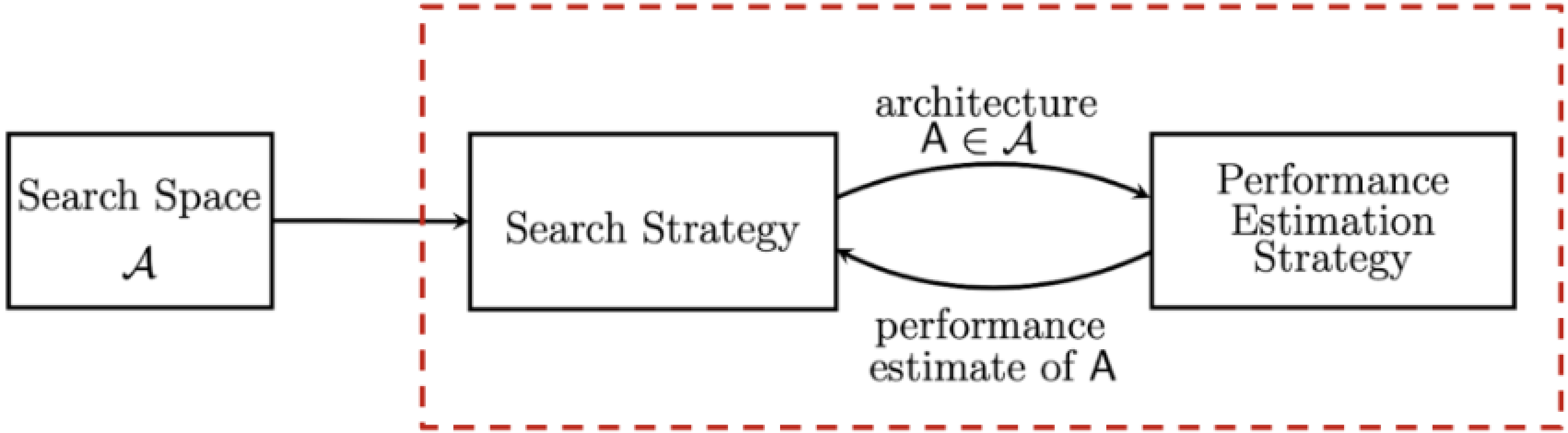
ROCKET Architecture



Motivation

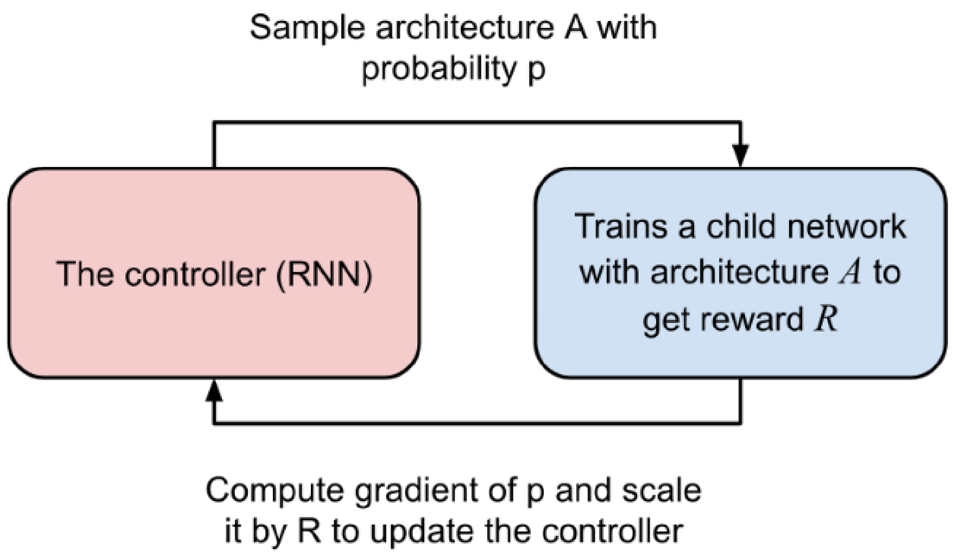
Popular Hyperparameter Optimization Techniques

Neural Architecture Search (NAS) automates the design of neural networks, particularly in CV filed.



Three main components of Neural Architecture Search (NAS) models. (Image source: Elsken, et al. 2019)

NAS RL for Model Automation



A high level overview of NAS, containing a RNN controller and a pipeline for evaluating child models. (Image source: Zoph & Le 2017)

The controller is trained as a *reinforcement learning (RL)* task.

- **Action space**: A list of candidate networks
- **Reward**: Accuracy achieved by a candidate network at convergence
- **Loss**: Controller optimized using RL loss in NAS to maximize expected reward (high accuracy) using the gradient

Motivation

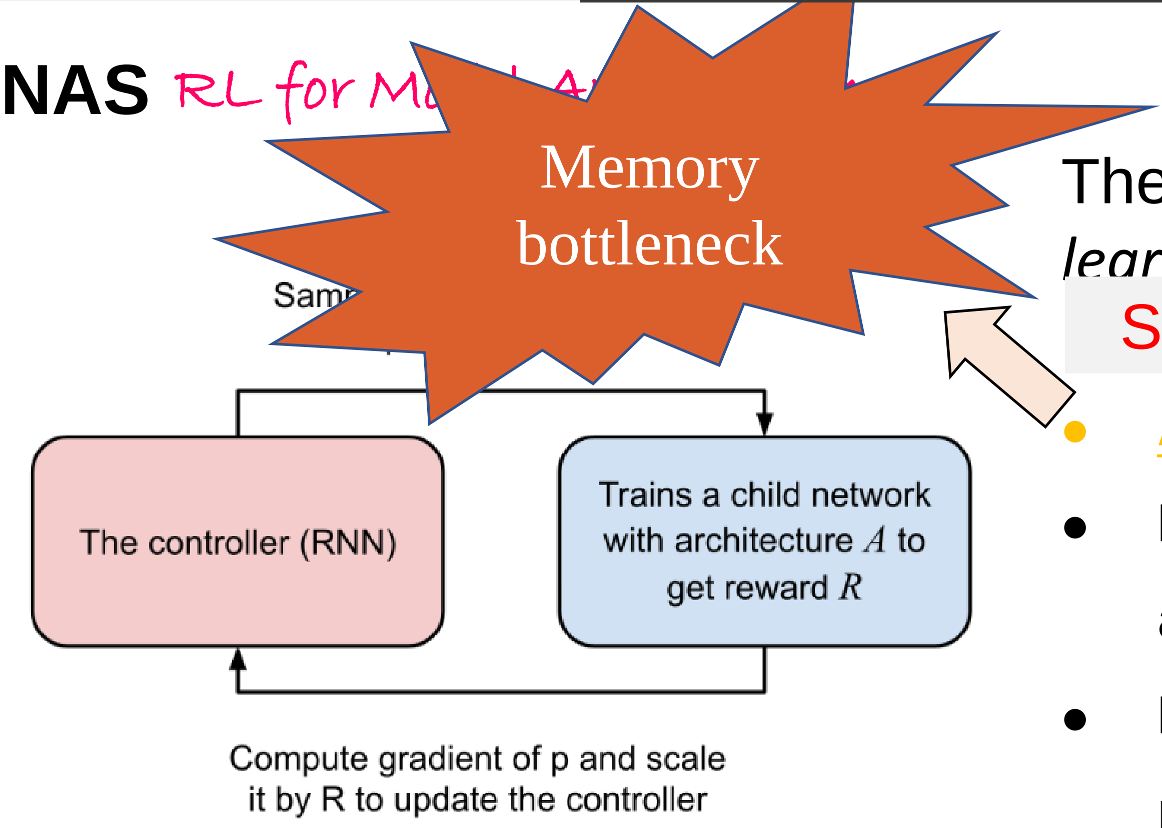
Goal: Create resource-efficient convolutional kernels!

✓ **Automatically** explore the **comprehensive** design space of hyperparameters, rather than relying on random convolution kernels.

Idea: Integrate **RL agents** into convolutional model building for multivariate time series classification tasks

NAS

RL for M



A high level overview of NAS, containing a RNN controller and a pipeline for evaluating child models. (Image source: Zoph & Le 2017)

The controller is trained as a *reinforcement learning (RL)* task.

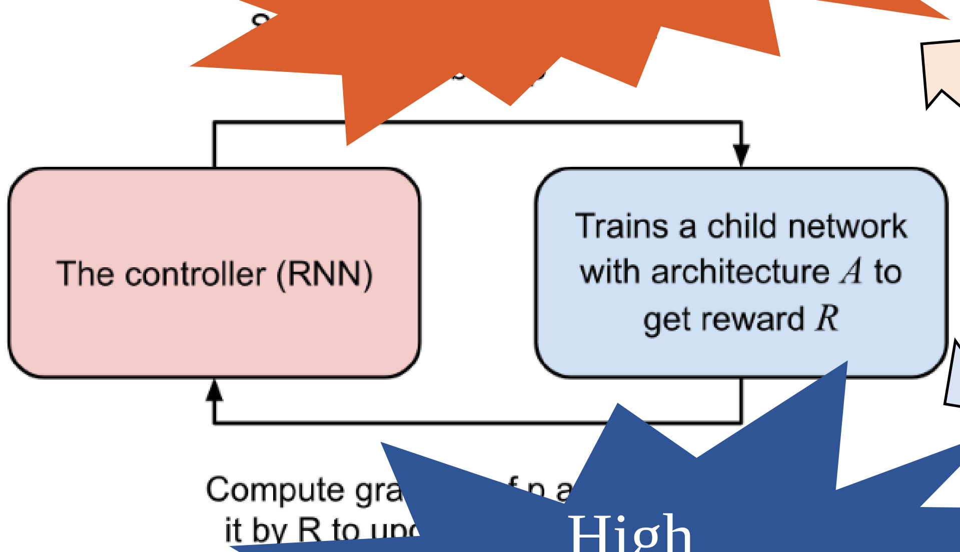
Store

- **Action space**: A list of candidate networks
- **Reward**: Accuracy achieved by a candidate network at convergence
- **Loss**: Controller optimized using RL loss in NAS to maximize expected reward (high accuracy) using the gradient.

NAS

RL for

Memory bottleneck



High computational overhead

A high lev... a RNN controller ... evaluating child models. (Image source: Zoph & Le 2017)

The controller is trained as a *reinforcement learning (RL)* task.

Store

- **Action space**: A list of candidate networks
- **Reward**: Accuracy achieved by a candidate network at convergence

Train

Controller optimized using RL loss in NAS to maximize expected reward (high accuracy) using the gradient.

Motivation

Goal: Create resource-efficient convolutional kernels!

✓ **Automatically** explore the **comprehensive** design space of hyperparameters, rather than relying on random convolution kernels.

Idea: Integrate **RL agents** into convolutional model building for multivariate time series classification tasks

Research question: How to design an **efficient** RL agent ?

Our work

Method: ADAptive Convolutional KErnel Transform (Adacket)

Automatically generate 1D convolutional kernels to transform specific channels of input time series data into discriminative representations.

↓

A sequential decision-making problem using the RL paradigm

↓

At each timestep, the RL agent encodes a specific channel embedding in the MTS data, preparing to generate and utilize specific convolutional kernels for further transformation.

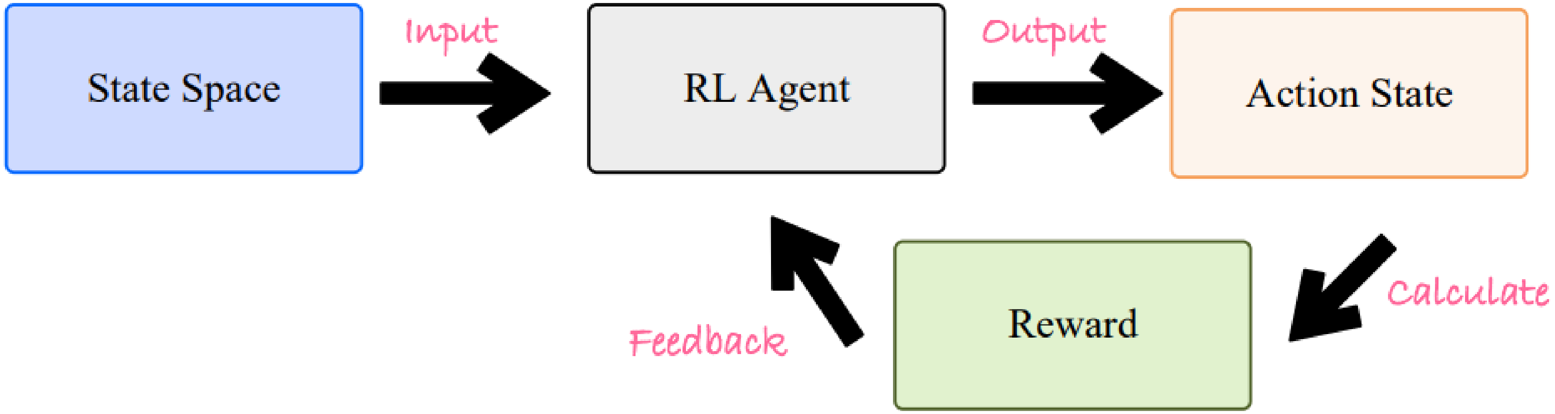
Our work

Method: ADAptive Convolutional KErnel Transform (Adacket)

Automatically generate 1D convolutional kernels to transform specific channels of input time series data into discriminative representations.

RL Agent :

- **State Space**
- **Action Space**
- **Reward**

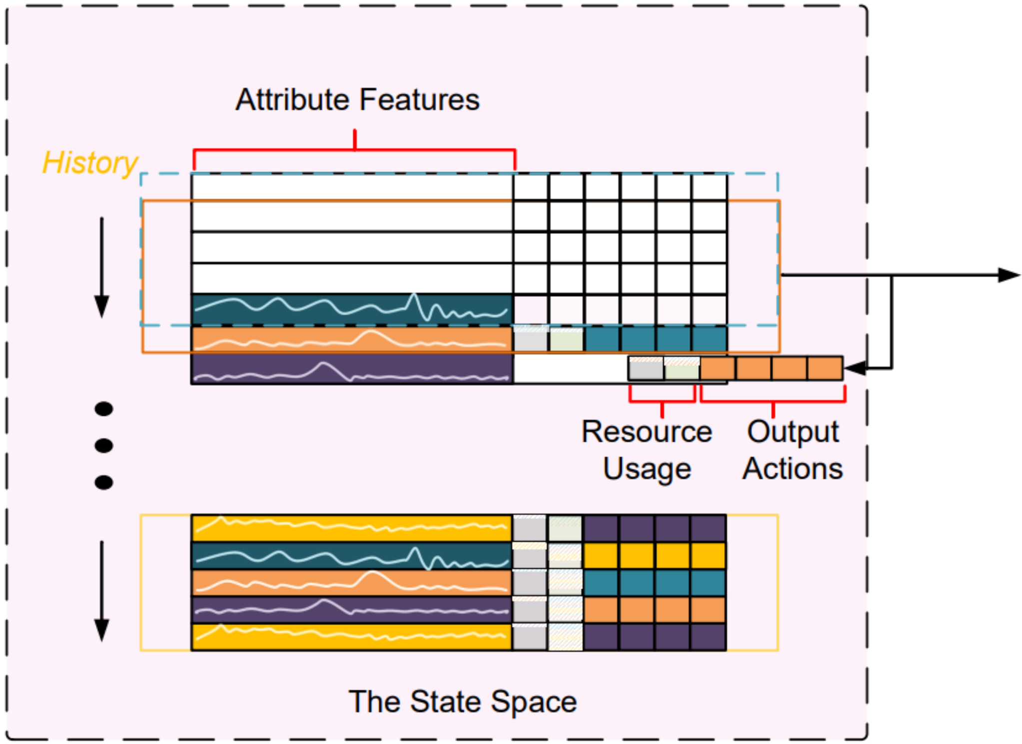


Our work

- **State Space:** The historical observations and one current observation (i.e., channel embedding)

The channel embedding:

- **Attribute features** associated with this channel, such as its index, temporal patterns.
- **Dynamic environment properties,** including resource usage and historical actions.



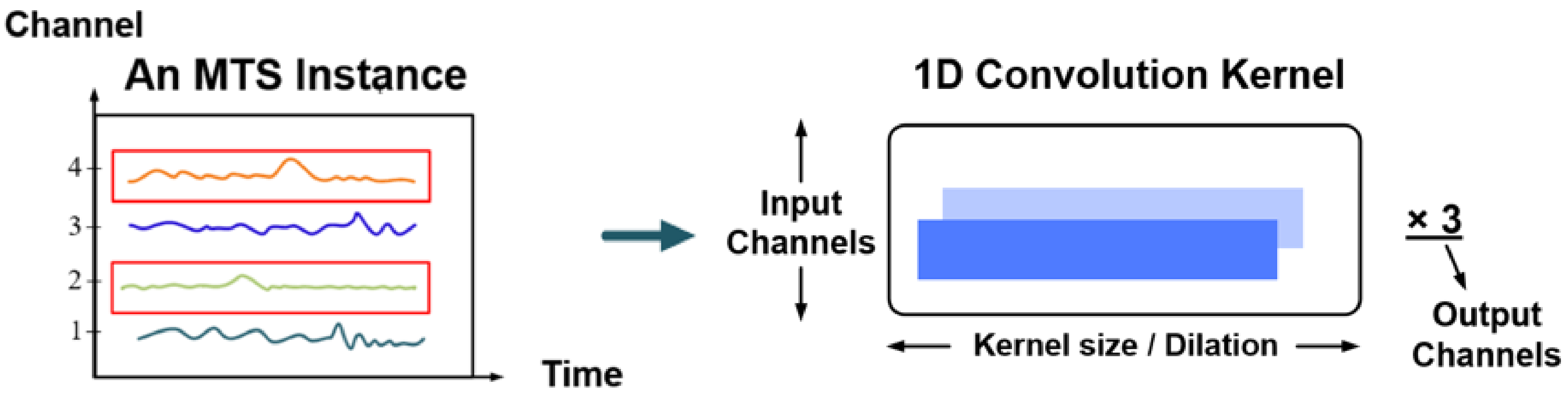
Our work

- **Action Space:** Form a kernel-channel pair to specify the input channels of the MTS data for a set of convolutional kernels.

Channel & Temporal

At each timestep：

Four output actions values map into input and output channels, kernel size, and dilation.



Our work

- **Action Space:** Form a kernel-channel pair to specify the input channels of the MTS data for a set of convolutional kernels.

Channel & Temporal

At each timestep：

Four output actions values map into input and output channels, kernel size, and dilation.

Each action: A continuous value ranging from 0 to 1.

✓ Fine-grained convolutional hyperparameters in the channel and temporal dimensions

✓ No need to store massive convolutional weights

Our work

- **Reward:** A multi-objective metric of candidate convolutional models

$$Reward(M) = \boxed{Score(M)} \times \epsilon + \boxed{\frac{Score(M)}{\log Resource(M)}} \times (1-\epsilon),$$

Model performance **Resource efficiency**

- *M* is a candidate convolutional model.
- *Score(M)* is converted from the contrastive loss function (Z. Yue et al. 2021) .
- Resource(M) is the sum of the parameters of the convolutional model and classifier.
- $\epsilon \in [0, 1]$ is a trade-off parameter.

Our work

- **Reward:** A multi-objective metric of candidate convolutional models

$$Reward(M) = \boxed{Score(M)} \times \epsilon + \boxed{\frac{Score(M)}{\log Resource(M)}} \times (1-\epsilon),$$

Model performance **Resource efficiency**

✓ Efficient performance evaluation: Replace accuracy with contrastive loss helps avoid time-consuming training processes.

✓ Adaptability: The balanced reward function for various resource-constrained scenarios.

Our work

Method: ADAptive Convolutional KErnel Transform (Adacket)

Automatically generate 1D convolutional kernels to transform specific channels of input time series data into discriminative representations.

RL Agent :

- **State Space**
- **Action Space**
- **Reward**

DDPG Algorithm

Actor-critic structure

Adacket

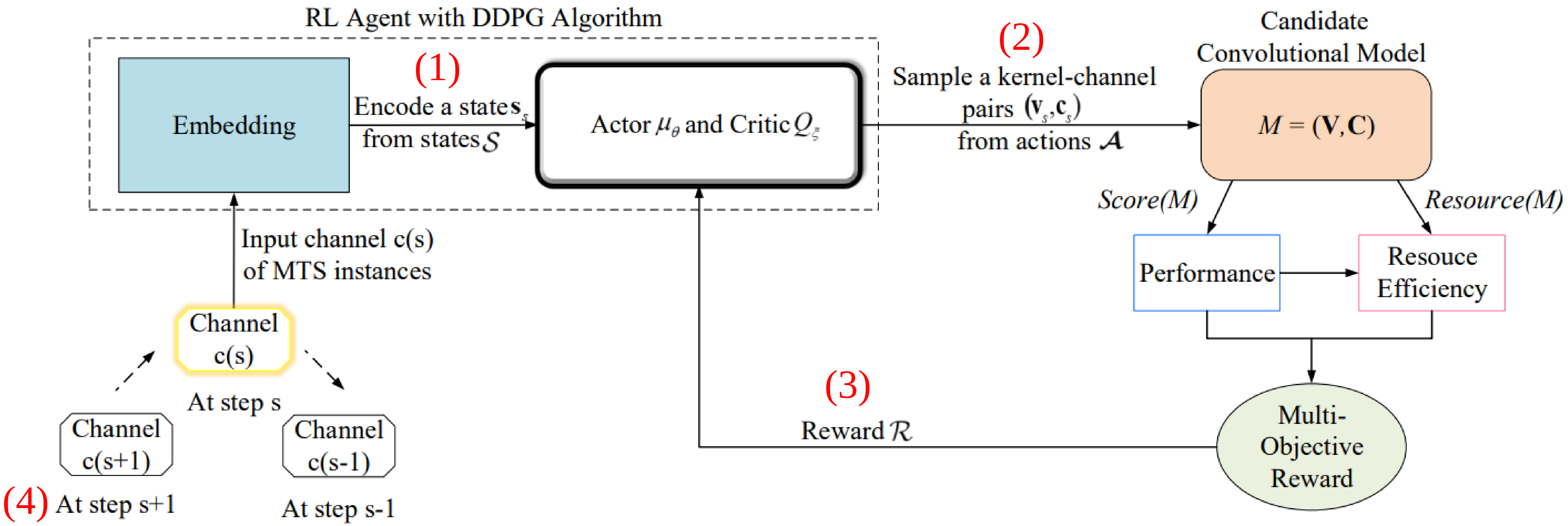


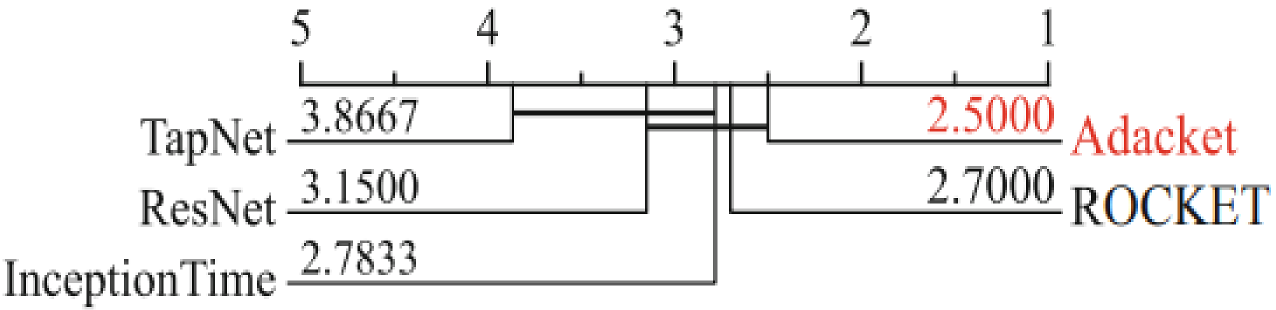
Illustration of Adacket to introduce one kernel-channel pair

Experiments

Datesets: UEA (30 MTSC tasks)

Baselines: TapNet, ResNet, InceptionTime, **ROCKET**

- Accuracy



CD diagrams for comparing different methods on all UEA datasets

Adacket outperforms all baseline methods in average accuracy rank in MTSC tasks.

Experiments

- Computational Efficiency

Method	Train Time	Inference Time
InceptionTime	48.55	1.61
ROCKET	**1.25**	1.58
Adacket	1.61	**0.68**

Train time (in hours) and inference time (in seconds) on all UEA datasets.

Adacket exhibits significantly faster training time compared to InceptionTime, similar to the performance of ROCKET.

Adacket stands out as the fastest method in terms of inference time, showcasing its efficient design.

Experiments

- Resource Efficiency

Dataset	Method	Acc	Params	Mem
SRS	InceptionTime	86.55	4.685	-
	ROCKET	84.69	-	42.88
	Adacket	**89.42**	**0.021**	**3.34**
HB	InceptionTime	73.20	4.810	-
	ROCKET	71.76	-	32.64
	Adacket	**77.07**	**0.012**	**0.58**
DDG	InceptionTime	54.00	7.754	-
	ROCKET	46.13	-	8.00
	Adacket	**58.00**	**0.003**	**0.68**

Comparison of accuracy (%), parameters (MB), and memory cost (MB) of three MTSC datasets with **different number of channels**.

Adacket achieves superior accuracy while utilizing fewer parameters and less memory.

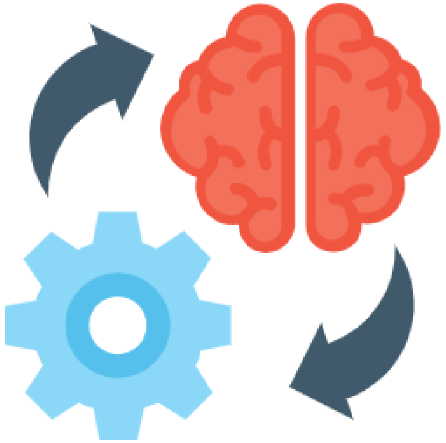
Adacket's adaptability to dataset characteristics, as opposed to fixed structures used by InceptionTime and ROCKET, highlights its flexibility and efficiency.

The DDG dataset, with 1345 channels, has the most channels in the UEA archive.

Contributions

To our best knowledge, Adacket is the first MTSC approach to incorporate RL for convolutional kernels adaptation.

- Introduce a multi-objective convolutional kernel search method that jointly optimizes performance and resource efficiency.
- Novelly model a multi-objective issue as a sequential decision-making problem using the RL paradigm, which enables the automatic design of convolutional kernels.
- Propose a comprehensive search of the convolutional kernel design space through multiple action spaces.
- Adacket exhibits excellent performance on both accuracy and resources.



Questions ?

Presenter: Junru Zhang
junruzhang@zju.edu.cn

Any questions? (ask now or @poster OGR on Tuesday evening)

Discussion and cooperation are welcome.