ProtoTSNet: Interpretable Time Series Classification With Prototypical Parts

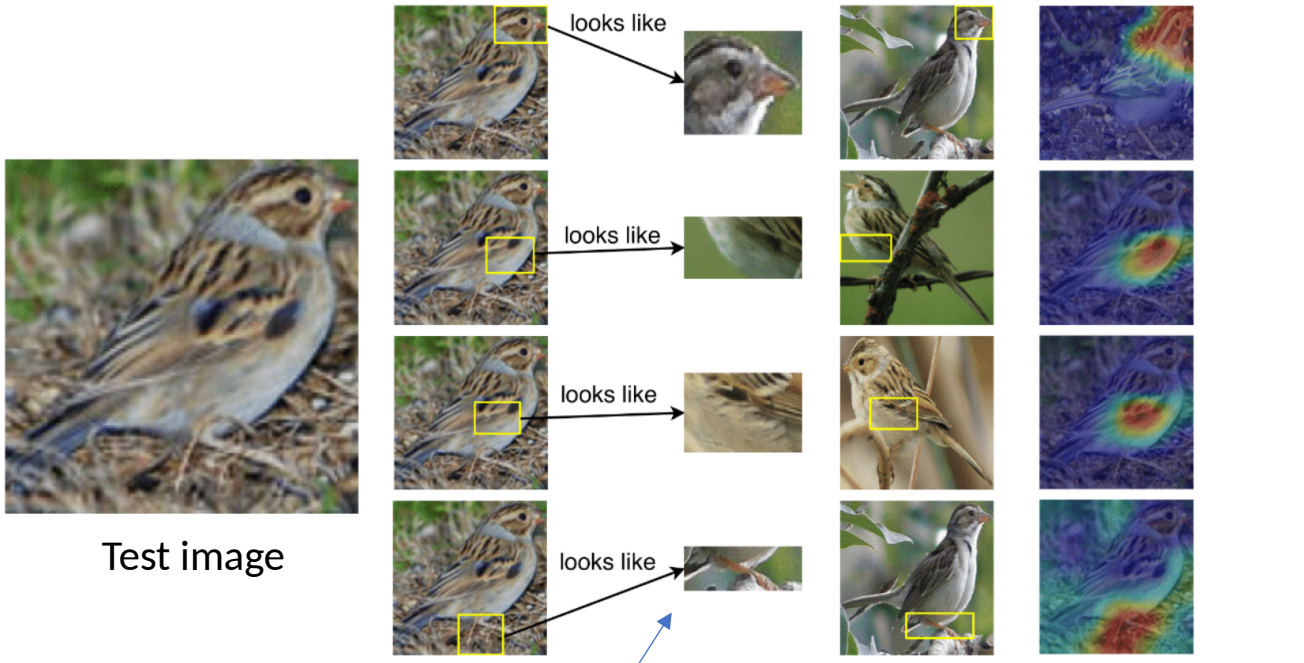
Bartłomiej Małkus

Doctoral School of Exact and Natural Sciences
Technical Computer Science

GEIST
Research
Group

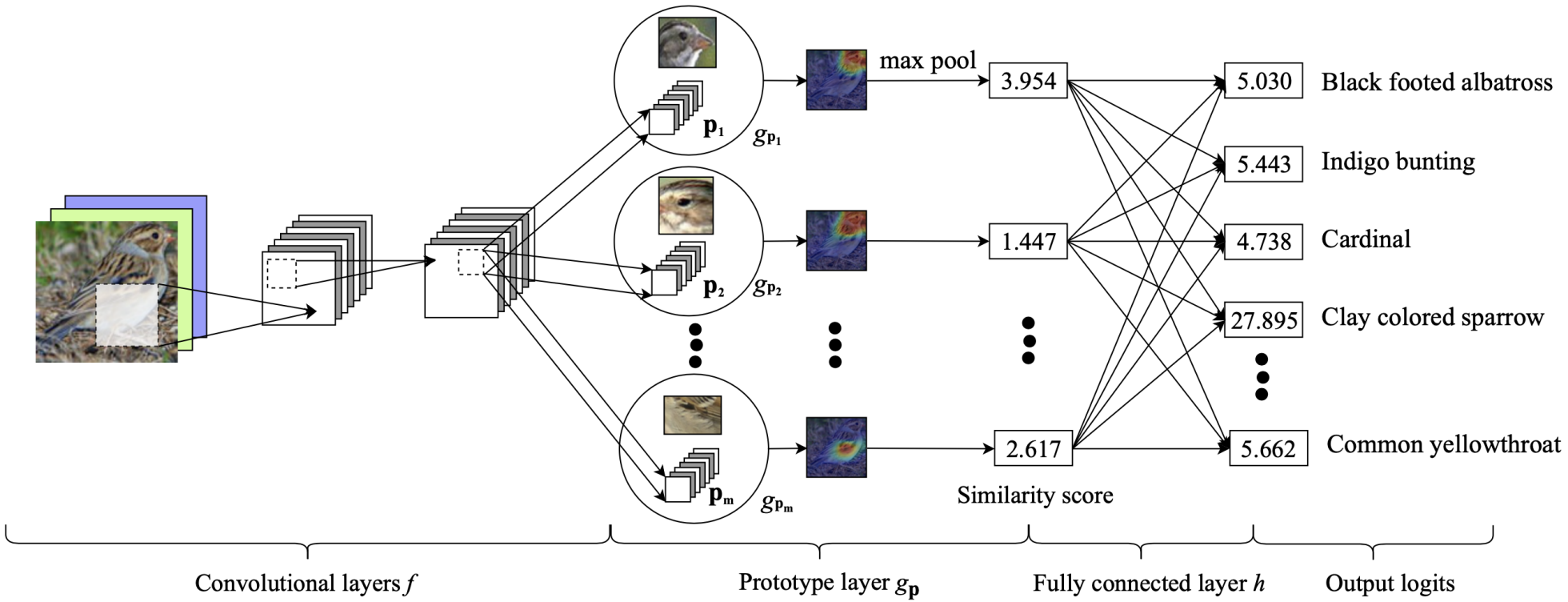
Principle of operation of ProtoPNet

The high-level idea of ProtoPNet is that the network learns some specific parts of images from the training set (prototypical parts), and it makes decisions based on the similarity of new input images to these existing prototypes.



This Looks Like That: Deep Learning for Interpretable Image Recognition, Chen et al. (2021)

Principle of operation of ProtoPNet



This Looks Like That: Deep Learning for Interpretable Image Recognition, Chen et al. (2021)

Challenges

Architecture specific:

- different epochs kinds, run in cycles
- multiple hyperparameters – regularization, protypes size, number of epochs of each kind

Challenges

Architecture specific:

- different epochs kinds, run in cycles
- multiple hyperparameters – regularization, protypes size, number of epochs of each kind

Time series specific:

- features may be not equally important
- for images, we can utilize pretrained encoders, like ResNet, DenseNet, VGG there are no such encoders for time series data
- prototype visualization is more challenging

Challenges

Architecture specific:

- different epochs kinds, run in cycles
- multiple hyperparameters – regularization, protypes size, number of epochs of each kind

Time series specific:

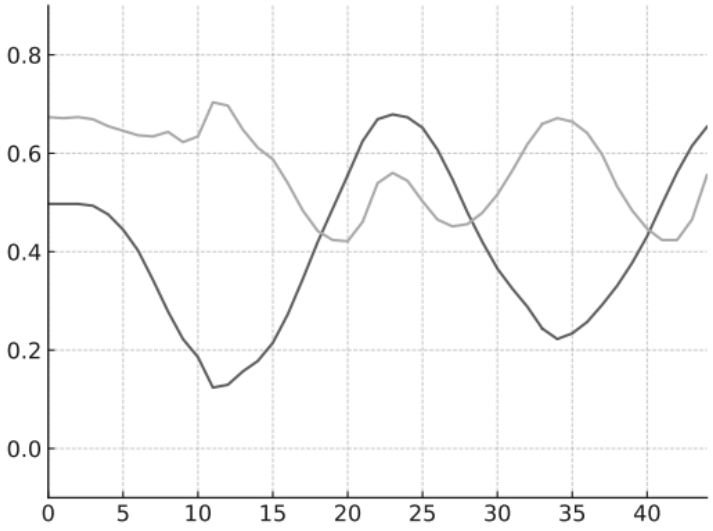
- features may be not equally important
- for images, we can utilize pretrained encoders, like ImageNet, DenseNet, VGG there are no such encoders for time series data
- prototype visualization is more challenging

…

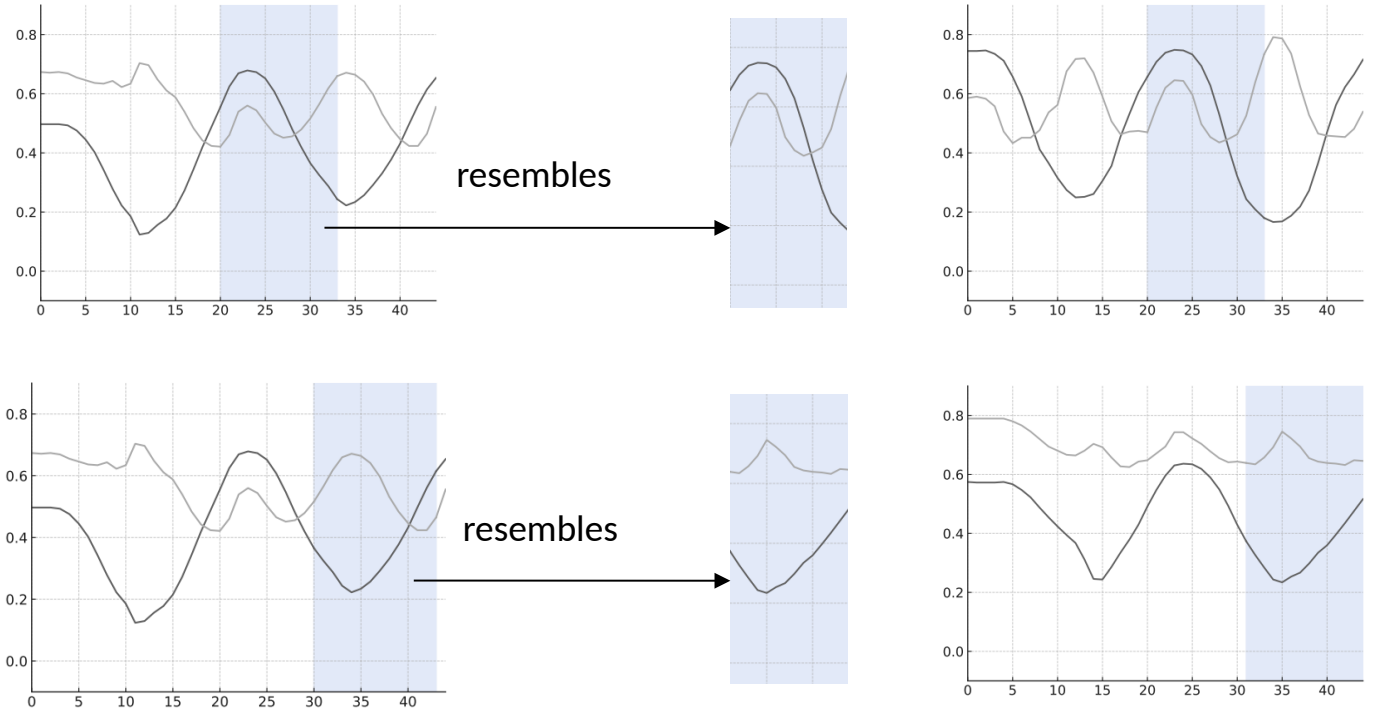
Existing works

- word extraction and bag-of-patters approaches
  - mostly non-explainable
  - weak scalability
- prototypical approaches
  - some are non-explainable
  - prototypes cover whole input sequences
- lots of methods are limited to univariate data
- multivariate explainable approaches provide post-hoc explainability

Principle of operation of ProtoTSNet



Test instance

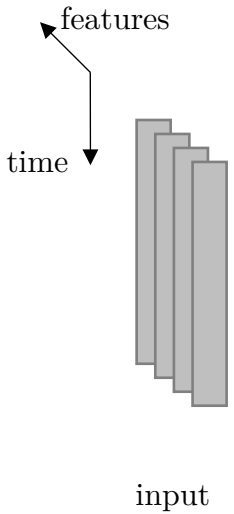


Prototypical parts

Prototype source instances

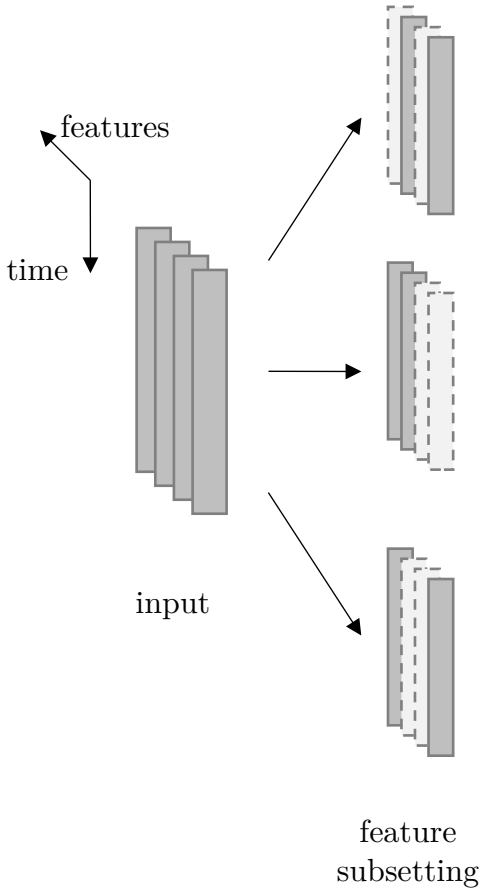
Our solution

The centerpiece of our solution is modified convolutional encoder



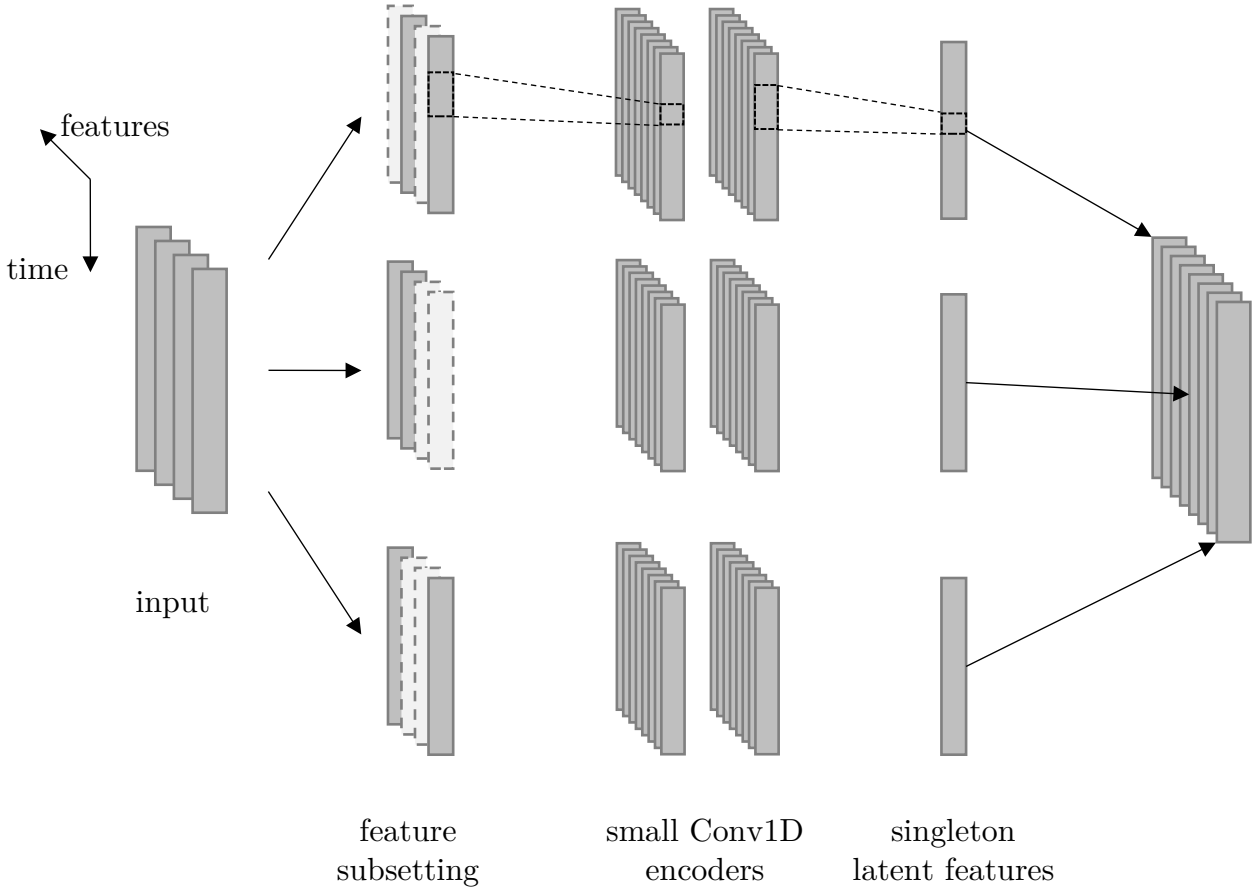
Our solution

We introduce random masks for input features (we're subsetting them)



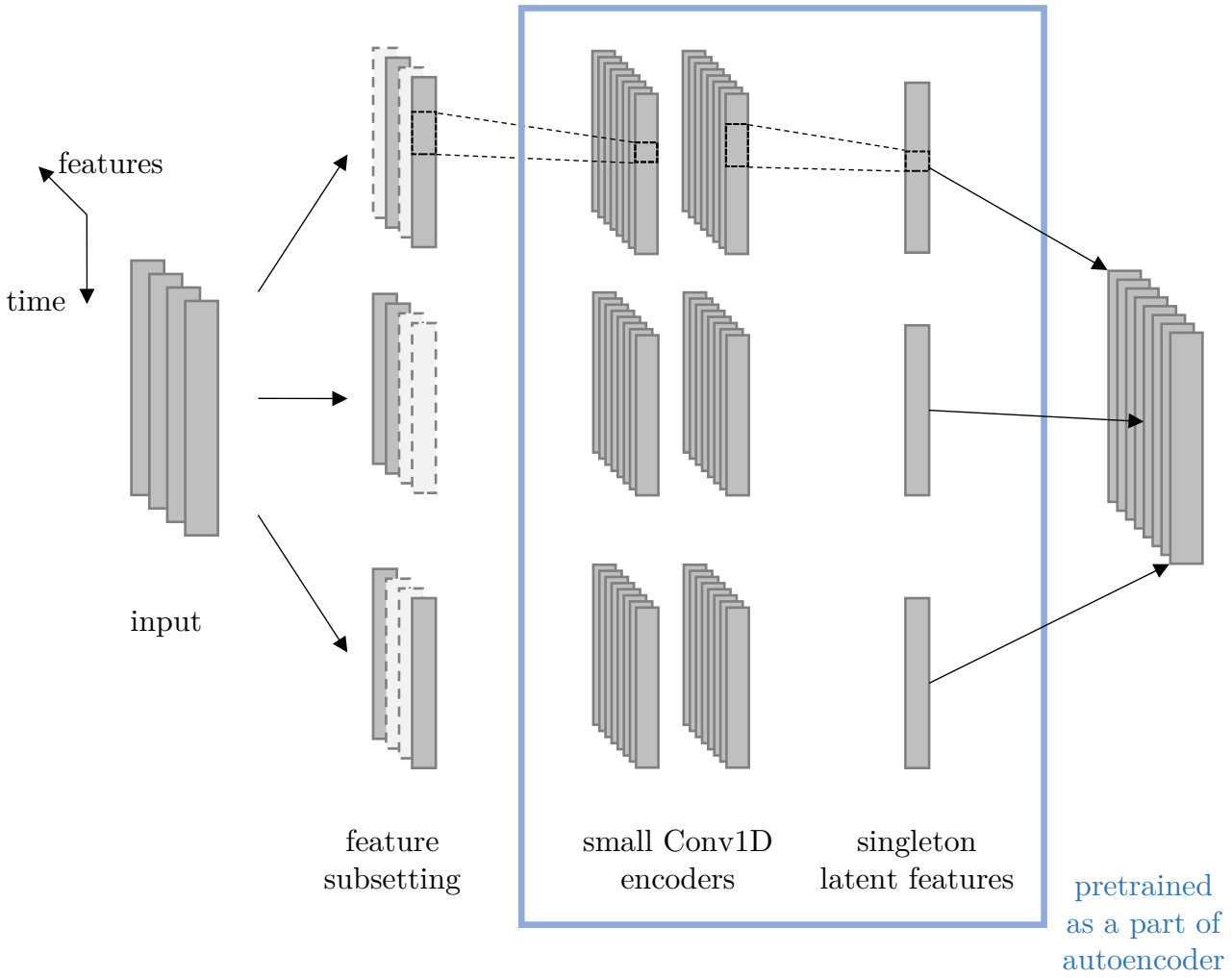
Our solution

We pass each subset to a small convolutional encoder – each encoder is producing single latent feature



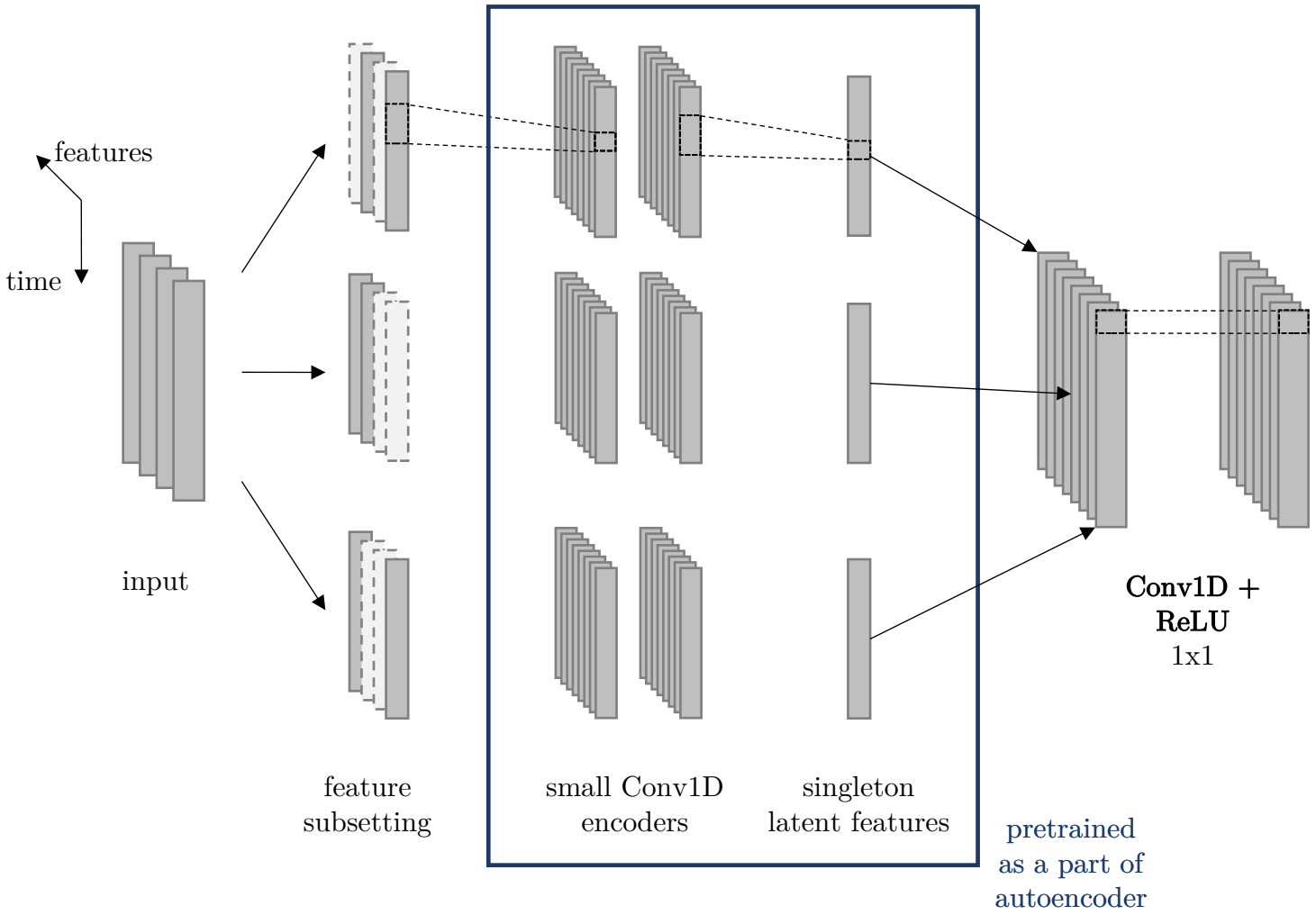
Our solution

Such encoder is pretrained as a part of regular autoencoder prior to the main training process



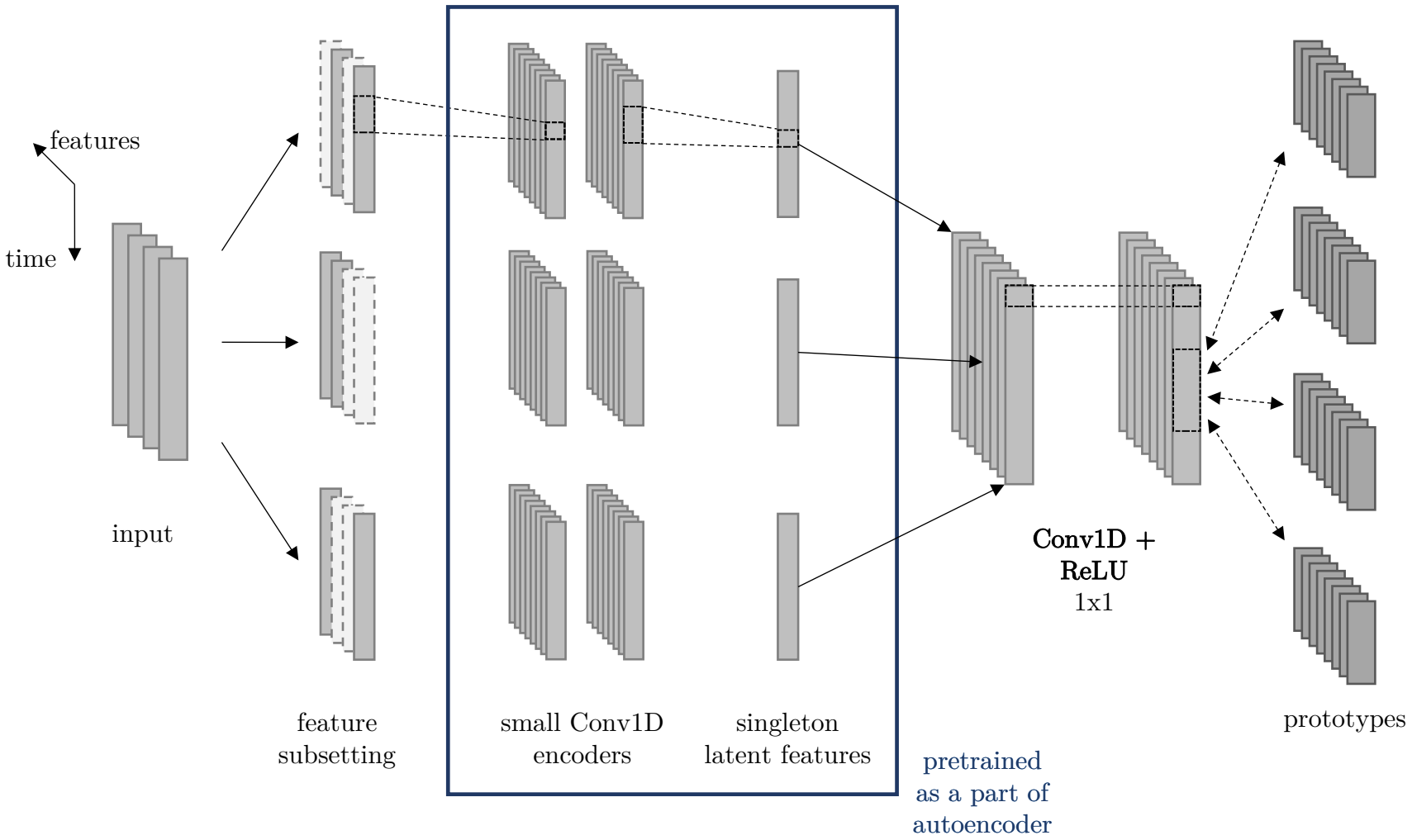
Our solution

We introduce single convolution layer with 1x1 filters – this allows for feature importance calculation



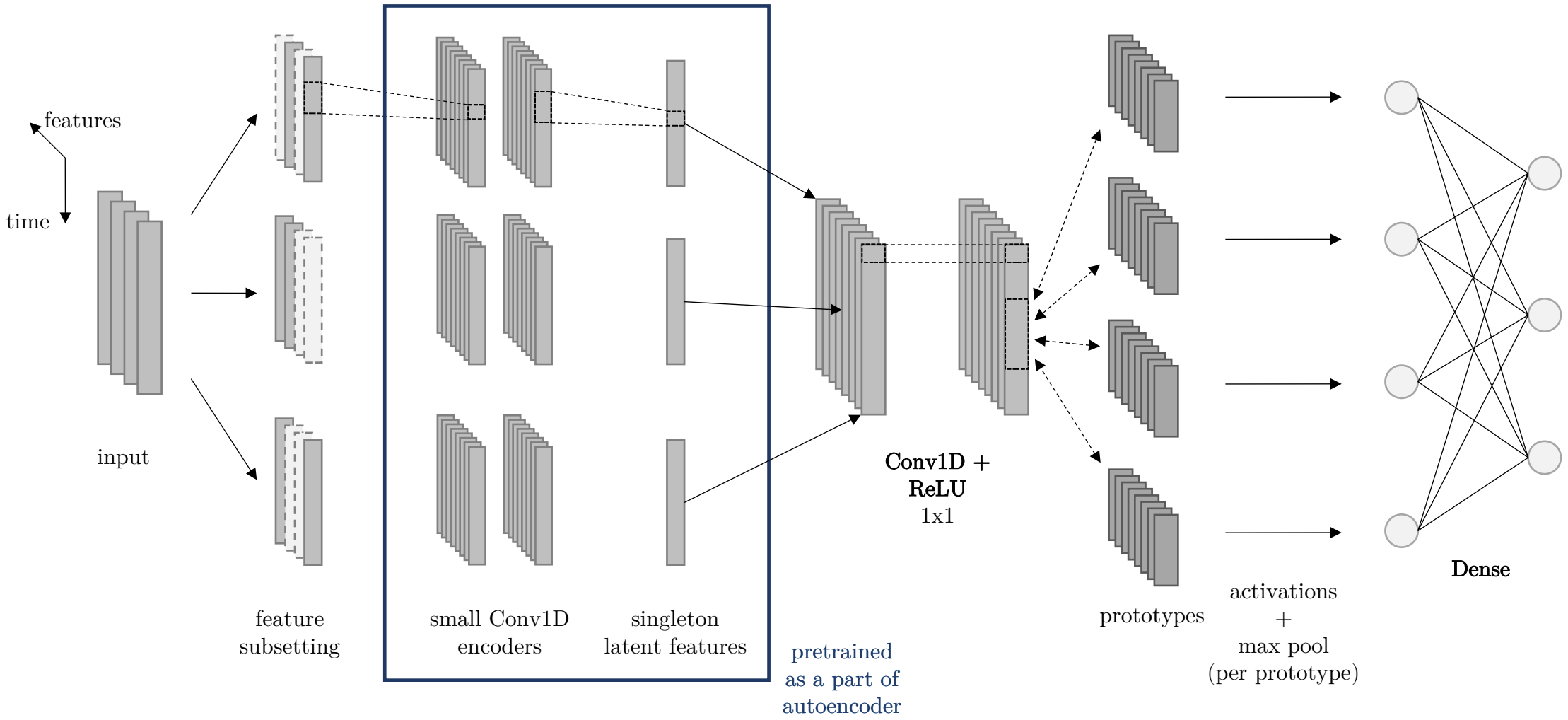
Our solution

Output encoded in such a way is passed to the prototype layer

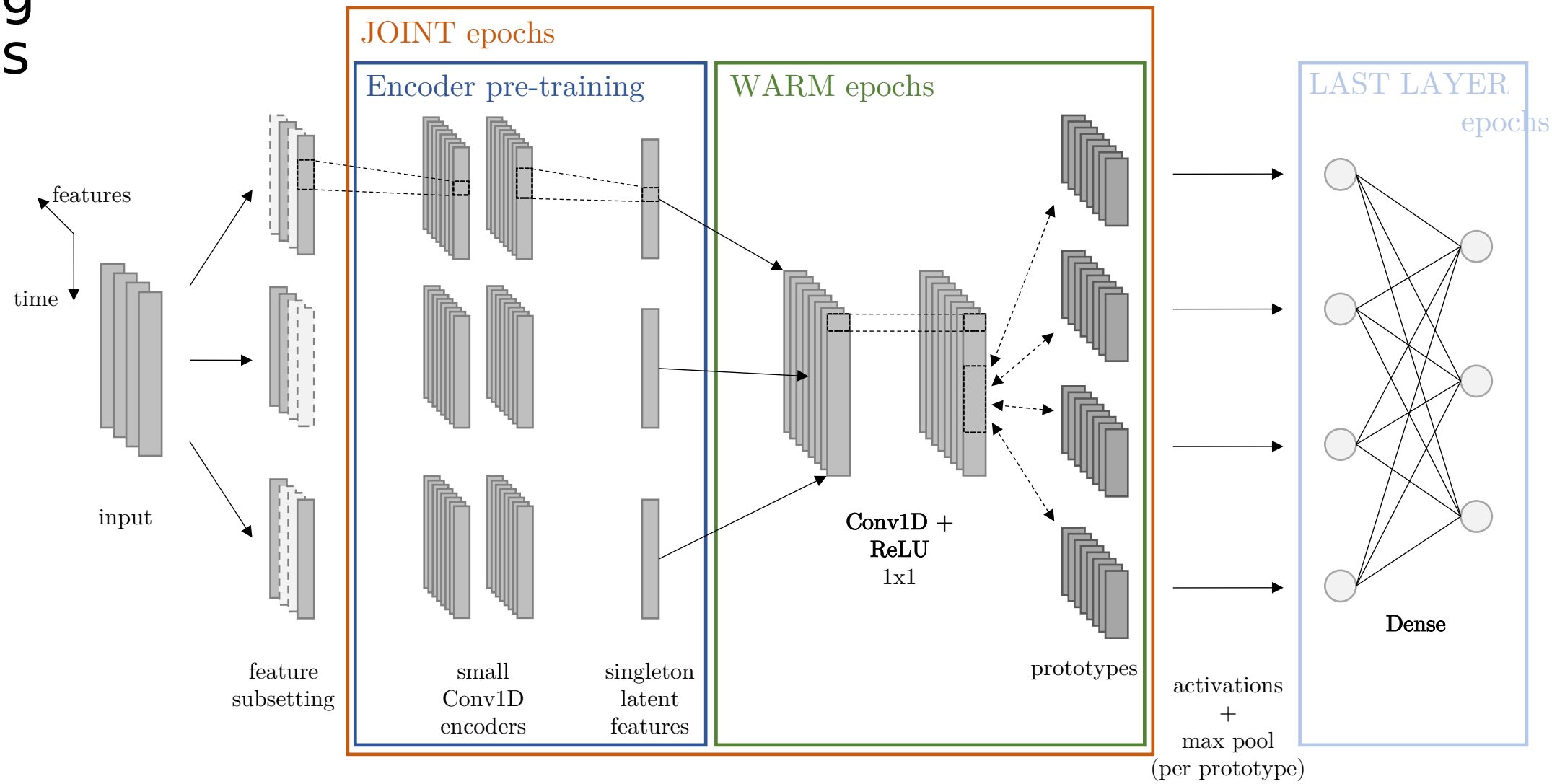


Our solution

In the last step, activations are passed to the dense layer which performs the classification

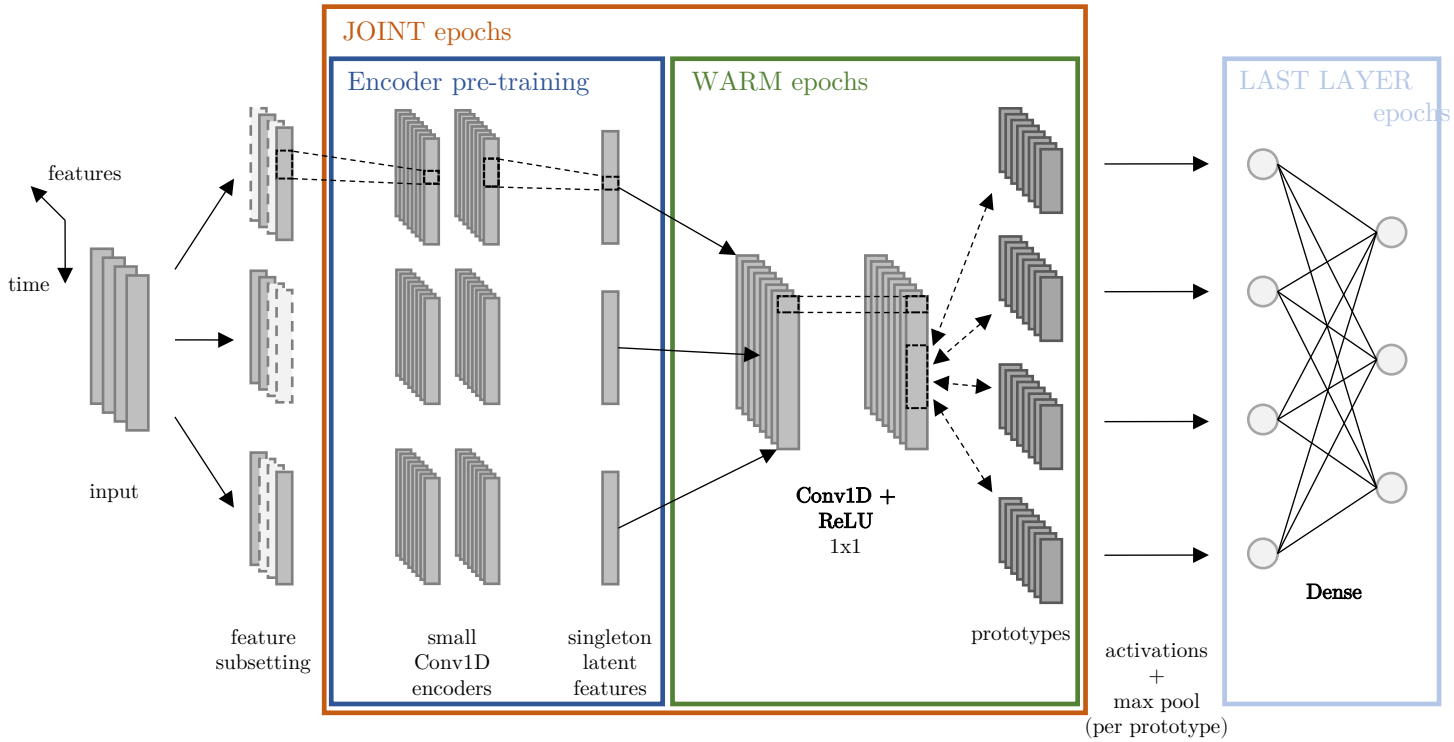


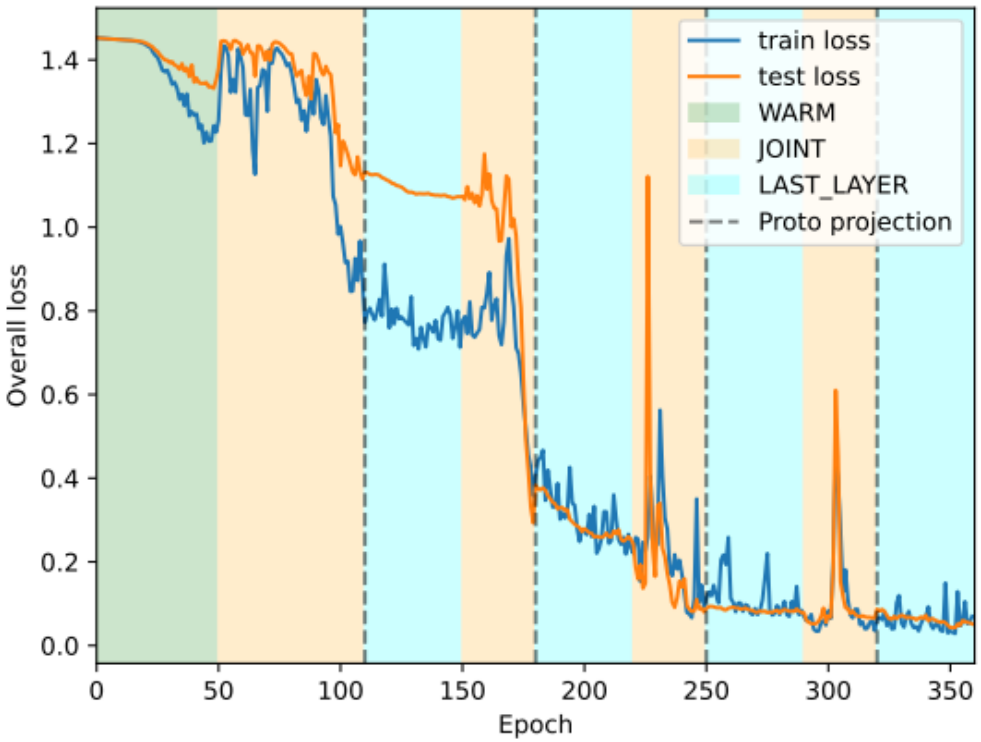
Training process



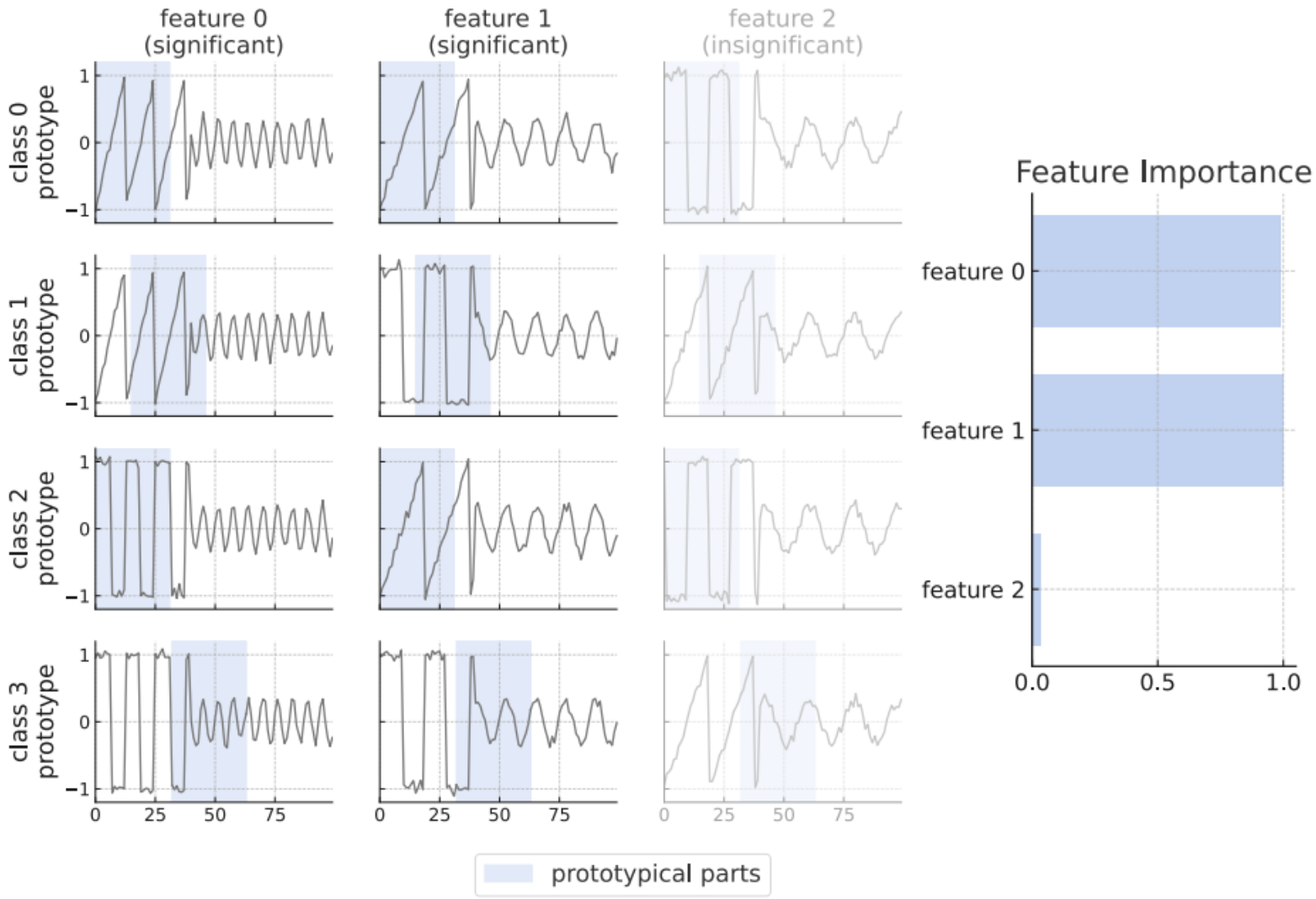
Encoder pre-training → WARM → JOINT → PUSH → LAST LAYER

Training process





Synthetic dataset evaluation



Real-world tests

Tested on 30 UEA multivariate datasets (*timeseriesclassification.com*) along with 5 other methods

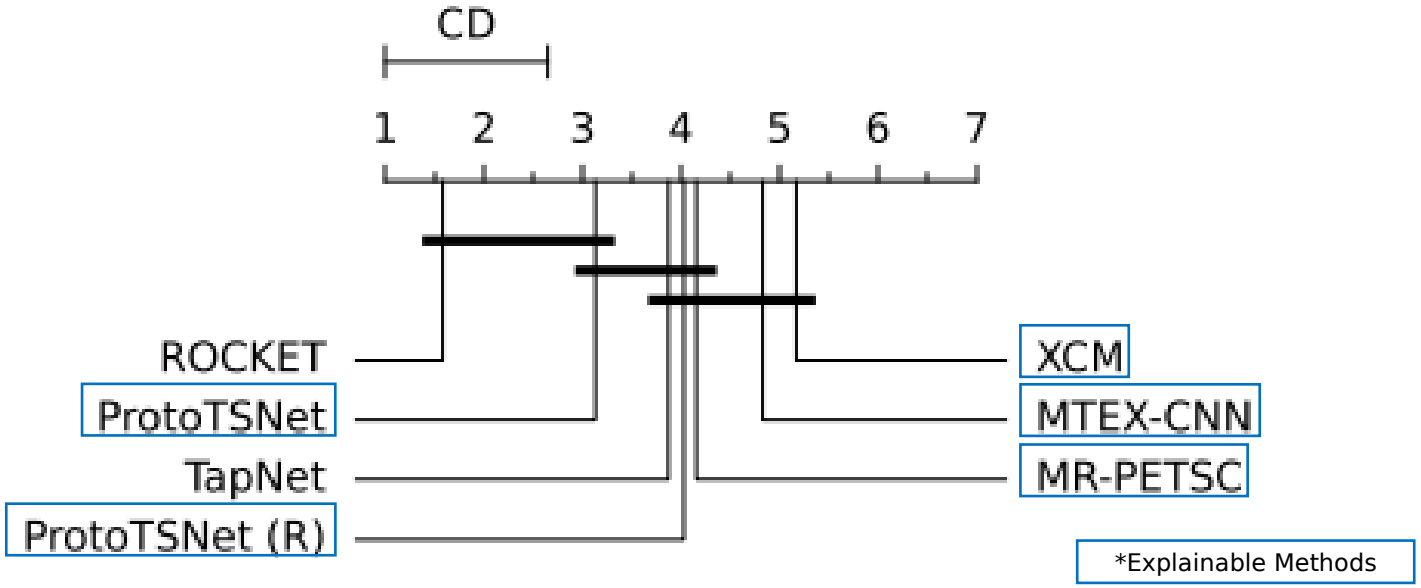
Table of ranks based on accuracy of each method for each dataset
divided into black-box and explainable groups

	Explainable					Black-Box	
	ante hoc			post hoc			
	ProtoTSNet	ProtoTSNet (regular encoder)	MR-PETSC	XCM	MTEX-CNN	TapNet	ROCKET
Avg. Rank	3,13	4,03	4,16	5,17	4,83	3,87	1,58
Wins/Ties	1	0	2	0	2	5	21
Avg. Rank	**1,97**	2,77	2,84	3,63	3,33		
Wins/Ties	**13**	5	8	2	3		

Real-world tests

Tested on 30 UEA multivariate datasets (*timeseriesclassification.com*) along with 5 other methods

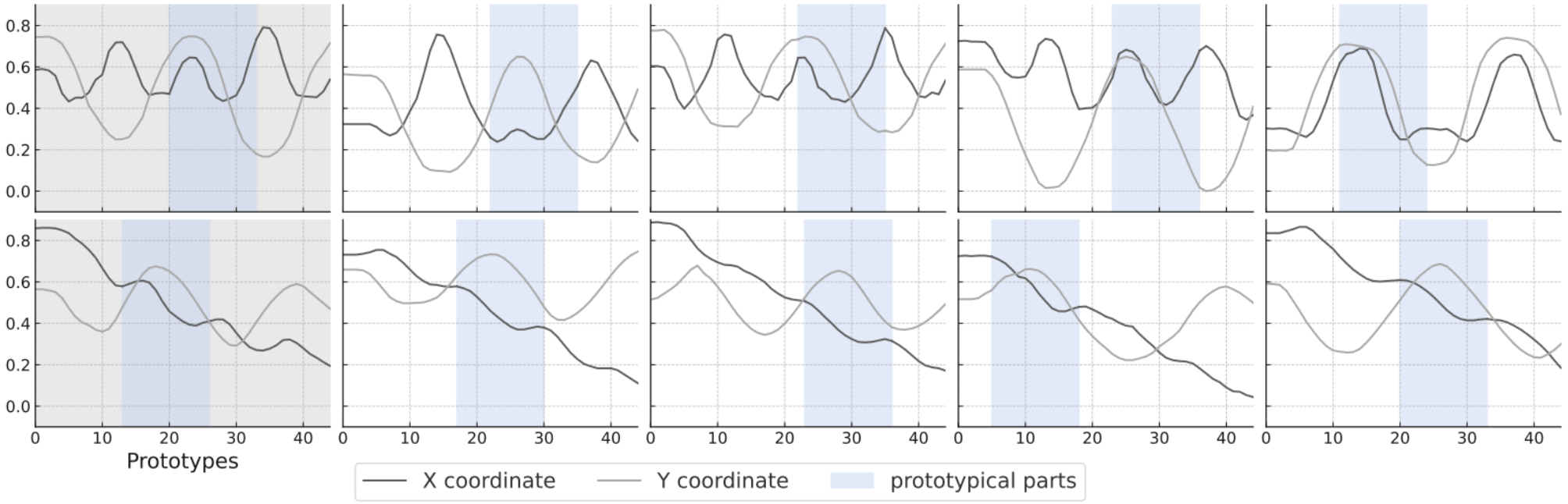
Critical difference diagram of tested methods
(methods connected by bars statistical differences from each other)



Real-world tests – Libras dataset

Hand sign language dataset

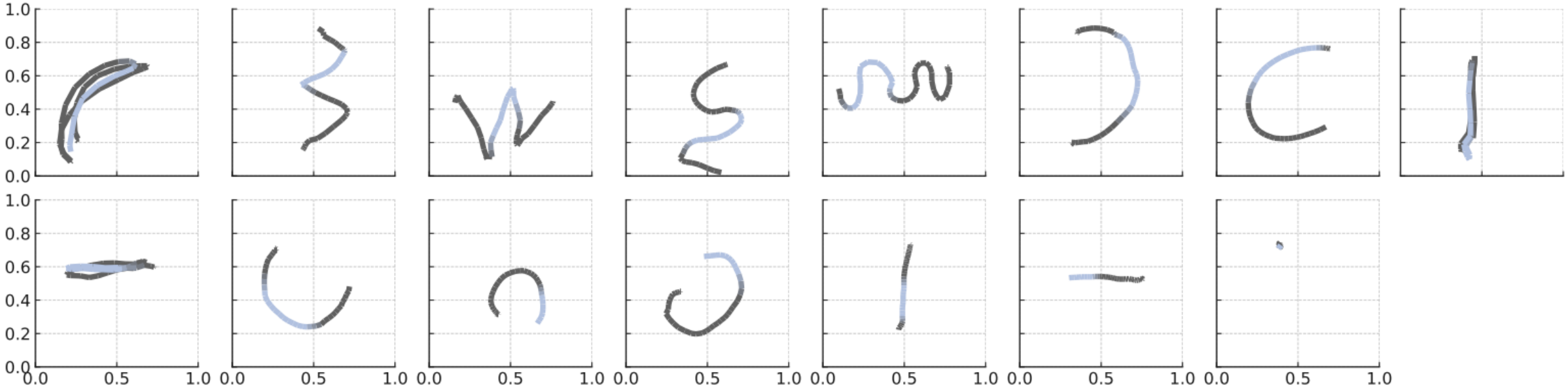
Prototypical parts for example gestures shown, along with instances classified based on similarity to them



Corresponding hand signs

Real-world tests – Libras dataset

Prototypes transformed and shown as 2D hand gestures along with prototypical parts (in blue)



DeepProbLog as ProtoTSNet complement

Prolog

```
likes(john, susie).                          /* John likes Susie */

likes(X, susie).                             /* Everyone likes Susie */

likes(john, Y).                              /* John likes everybody */

likes(john, Y), likes(Y, john).              /* John likes everybody and everybody likes John */

likes(john, susie); likes(john, mary).       /* John likes Susie or John likes Mary */
not(likes(john, pizza)).                     /* John does not like pizza */

likes(john, susie) :- likes(john, mary).     /* John likes Susie if John likes Mary. */
friends(X,Y) :- likes(X,Y),likes(Y,X).       /* X and Y are friends if they like each other */
hates(X,Y) :- not(likes(X,Y)).               /* X hates Y if X does not like Y. */
enemies(X,Y) :- not(likes(X,Y)),not(likes(Y,X)). /* X and Y are enemies if they don't like each other */
```

ProbLog

```
person(john).
person(mary).

0.7::burglary.
0.2::earthquake.
0.9::alarm :- burglary, earthquake.
0.8::alarm :- burglary, \+earthquake.
0.1::alarm :- \+burglary, earthquake.

0.8::calls(X) :- alarm, person(X).
0.1::calls(X) :- \+alarm, person(X).

evidence(calls(john),true).
evidence(calls(mary),true).
query(burglary).    % 0.98193926
query(earthquake).  % 0.22685136
```

ProbLog

```
person(john).
person(mary).

0.7::burglary.
0.2::earthquake.
0.9::alarm :- burglary, earthquake.
0.8::alarm :- burglary, \+earthquake.
0.1::alarm :- \+burglary, earthquake.

0.8::calls(X) :- alarm, person(X).
0.1::calls(X) :- \+alarm, person(X).

evidence(calls(john),true).
evidence(calls(mary),true).
query(burglary).    % 0.98193926
query(earthquake).  % 0.22685136
```

DeepProbLog

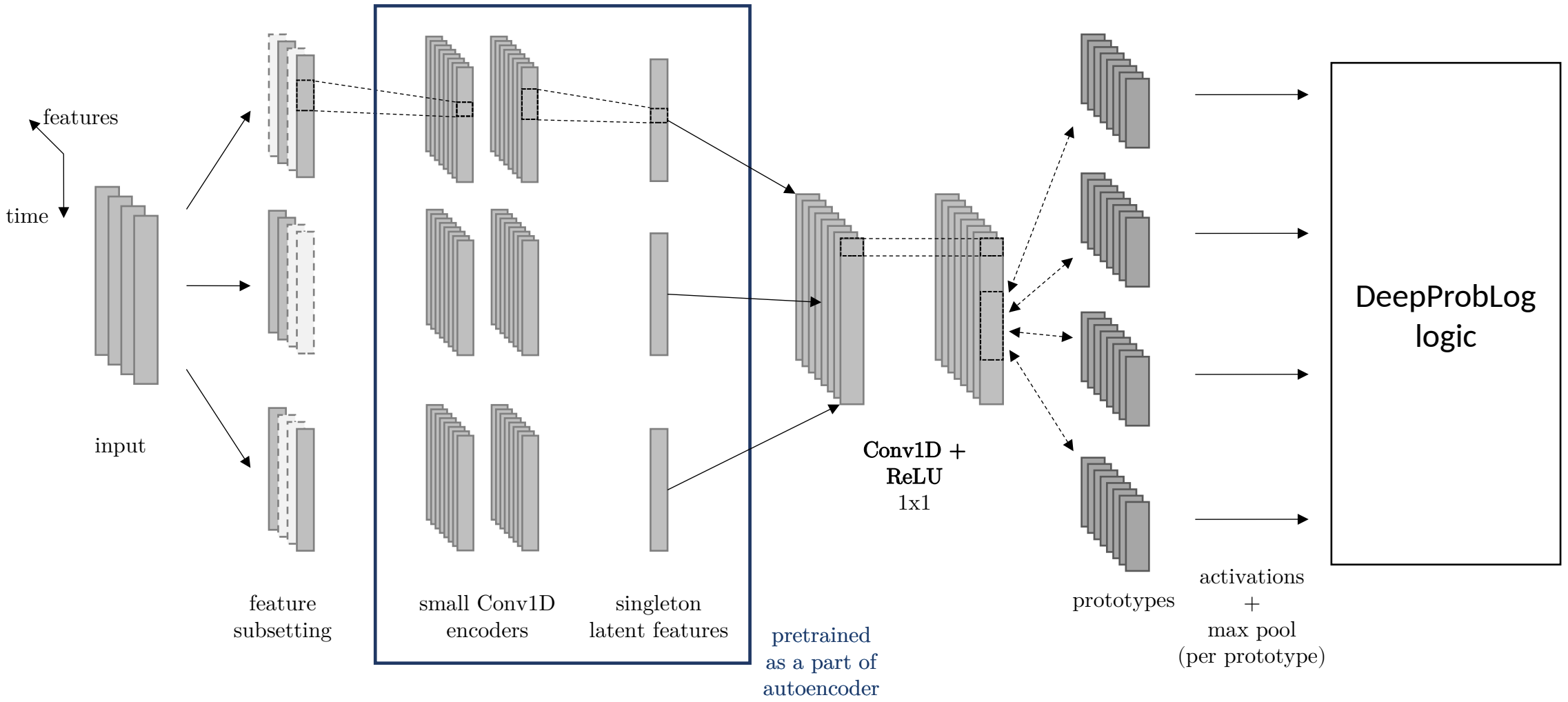
```
nn(mnist_net,[X],Y,[0,1,2,3,4,5,6,7,8,9])::digit(X,Y).
```

$nn(m_{\text{digit}}, \boxed{5}, [0, \ldots, 9]) :: \texttt{digit}(\boxed{5}, 0); \ldots; \texttt{digit}(\boxed{5}, 9).$

```
addition(X,Y,Z) :- digit(X,X2), digit(Y,Y2), Z is X2+Y2.
```

?- addition( ,  , 8)

ProtoTSNet + DeepProbLog



ProtoTSNet + DeepProbLog

```
nn(ptsnet, [TS, P], H, [0])::has_proto(TS, P, H).

is_class(TS, c0) :- class(c0), has_proto(TS, p0).
is_class(TS, c1) :- class(c1), has_proto(TS, p1).

class(c0).
class(c1).

query(is_class(ts0, c0)).
query(is_class(ts1, c1)).
```

```
nn(ptsnet, [TS, P], H, [0, 1])::has_proto(TS, P, H).

t(_)::connected(p0, c0).
t(_)::connected(p1, c1).

is_class(TS, c0) :- class(c0), has_proto(TS, p0), connected(p0, c0).
is_class(TS, c1) :- class(c1), has_proto(TS, p1), connected(p1, c1).

class(c0).
class(c1).

query(is_class(ts0, c0)).
query(is_class(ts1, c1)).
query(is_class(ts2, c1)).
```

Thank you for your attention