Uncertainty-DTW for Time Series and Sequences (Supplementary Material)

Lei Wang*,†,§ and Piotr Koniusz*,§,†

†Australian National University §Data61/CSIRO
§firstname.lastname@data61.csiro.au

Below we provide additional analyses, protocols and details of our work.

A Datasets and their statistics

Table 6: UCR archive (the latest version from 2018) which we use for time series analysis. The information is grouped based on the type of time series.

Dataset type	Avg. #train	Avg. #test	Total #classes	Avg. length
Device	1261	1135	44	895
ECG	708	1755	95	326
EOG	362	362	24	1250
EPG	40	249	6	601
Hemodynamics	104	208	156	2000
HRM	18	186	18	201
Image	595	1157	334	360
Motion	347	1057	99	517
Power	180	180	2	144
Sensor	420	1286	177	410
Simulated	203	1021	32	267
Spectro	179	147	24	553
Spectrum	305	388	17	1836
Traffic	607	1391	12	24
Trajectory	208	130	78	360

The UCR time series archive [56]. UCR, introduced in 2002, is an important resource in the time series analysis community with at least 1,000 published papers making use of at least 1 dataset from this archive. We use 128 datasets from the latest version of UCR from 2018, encompassing a wide variety of fields and lengths. Table 6 details the statistics of this archive by grouping the whole dataset into different types.
Few-shot action recognition datasets. Table 7 contains statistics of datasets used in our experiments. Smaller datasets listed below are used for more evaluations of supervised and unsupervised few-shot action recognition:

Table 7: Popular benchmark datasets which we use for few-shot action recognition.

Datasets	Year	Classes	Subjects	#views	#clips	Sensor	#joints
MSR Action 3D	2010	20	10	1	567	Kinect v1	20
3D Action Pairs	2013	12	10	1	360	Kinect v1	20
UWA 3D Activity	2014	30	10	1	701	Kinect v1	15
NTU RGB+D	2016	60	40	80	56,880	Kinect v2	25
NTU RGB+D 120	2019	120	106	155	114,480	Kinect v2	25
Kinetics-skeleton	2018	400	-	-	$\sim$ 300,000	-	18

– *MSR Action 3D* [57] is an older AR dataset captured with the Kinect depth camera. It contains 20 human sport-related activities such as *jogging*, *golf swing* and *side boxing*.
– *3D Action Pairs* [59] contains 6 selected pairs of actions that have very similar motion trajectories, *e.g.*, *put on a hat* and *take off a hat*, *pick up a box* and *put down a box*, *etc*.
– *UWA 3D Activity* [61] has 30 actions performed by 10 people of various height at different speeds in cluttered scenes.

As MSR Action 3D, 3D Action Pairs, and UWA 3D Activity have not been used in FSAR, we created 10 training/testing splits, by choosing half of class concepts for training, and half for testing per split per dataset. Training splits were further subdivided for crossvalidation. Section C.1 details the class concepts per split for small datasets.

B Table of notations

Table 8 (next page) shows the notations used in this paper with their short descriptions.

Table 8: Notations and their descriptions.

Notation	Description
$\boldsymbol{\Psi}$	Query feature maps
$\boldsymbol{\Psi}'$	Support feature maps
$\boldsymbol{\Pi}$	Path matrix
$\boldsymbol{D}(\cdot,\cdot)$	Pair-wise distances
$d^2_*(\cdot,\cdot)$	Distance functions and * can be base (squared Euclidean), DTW, sDTW or uDTW
γ	The relaxation parameter of sDTW/uDTW
τ	The number of temporal blocks for query
τ'	The number of temporal blocks for support
$\boldsymbol{\Sigma}$	Pair-wise variances between all possible pairs of two sequences
$\boldsymbol{\Sigma}^\dagger$	Element-wise inverse of $\boldsymbol{\Sigma}$
$f(\cdot;\cdot)$	Encoder function
$\mathcal{P}$	The set of parameters to learn
β	Regularization parameter
σ	Uncertainty parameter
$\mathbf{X}$	Query frames per block
$\mathbf{X}'$	Support frames per block
K	The size of dictionary
K'	The subset size for K' nearest anchors
$\mathbf{x}$	Time series for training
$\mathbf{x}'$	Time series for testing
$\boldsymbol{\mu}_c$	Class prototype for class c
$\Omega(\cdot)$	Regularization penalty
$\boldsymbol{\alpha}$	Coding vector
λ_{DL}	Learning rate for dictionary learning
λ_{EN}	Learning rate for encoder
$\boldsymbol{M}$	Dictionary anchors
B	The number of training episodes
N	The number of classes
Z	The number of samples from each class
J	The number of human body joints
d	Feature dimension after MLP
d'	Feature dimension (output of EN)
δ	The similarity label
N_c	The number of samples for class c

C Evaluation Protocols

Below, we detail our new/additional evaluation protocols used in the experiments on few-shot action recognition.

C.1 Few-shot AR protocols (the small-scale datasets)

As we use several class-wise splits for small datasets, these splits will be simply released in our code. Below, we explain the selection process that we used.

FSAR (MSR Action 3D) . As this dataset contains 20 action classes, we randomly choose 10 action classes for training and the rest 10 for testing. We repeat this sampling process 10 times to form in total 10 train/test splits. For each split, we have 5-way and 10-way experimental settings. The overall performance on this dataset is computed by averaging the performance over the 10 splits.

FSAR (3D Action Pairs) . This dataset has in total 6 action pairs (12 action classes), each pair of action has very similar motion trajectories, *e.g.*, *pick up a box* and *put down a box*. We randomly choose 3 action pairs to form a training set (6 action classes) and the half action pairs for the test set, and in total there are $\binom{n}{k}=\binom{6}{3}=20$ different combinations of train/test splits. As our train/test splits are based on action pairs, we are able to test whether the algorithm is able to classify unseen action pairs that share similar motion trajectories. We use 5-way protocol on this dataset to evaluate the performance of FSAR, averaged over all 20 splits.

FSAR (UWA 3D Activity) . This dataset has 30 action classes. We randomly choose 15 action classes for training and the rest half action classes for testing. We form in total 10 train/test splits, and we use 5-way and 10-way protocols on this dataset, averaged over all 10 splits.

C.2 One-shot protocol on NTU-60

Following NTU-120 [58], we introduce the one-shot AR setting on NTU-60. We split the whole dataset into two parts: auxiliary set (on NTU-120 the training set is called as auxiliary set, so we follow such a terminology) and one-shot evaluation set.

Auxiliary set contains 50 classes, and all samples of these classes can be used for learning and validation. Evaluation set consists of 10 novel classes, and one sample from each novel class is picked as the exemplar (terminology introduced by authors of NTU-120), while all the remaining samples of these classes are used to test the recognition performance.

Evaluation set contains 10 novel classes, namely A1, A7, A13, A19, A25, A31, A37, A43, A49, A55.

The following 10 samples are the exemplars:
(01)S001C003P008R001A001, (02)S001C003P008R001A007,
(03)S001C003P008R001A013, (04)S001C003P008R001A019,
(05)S001C003P008R001A025, (06)S001C003P008R001A031,
(07)S001C003P008R001A037, (08)S001C003P008R001A043,
(09)S001C003P008R001A049, (10)S001C003P008R001A055.

Auxiliary set contains 50 classes (the remaining 50 classes of NTU-60 excluding the 10 classes in evaluation set).

D Effectiveness of SigmaNet

In this section, we introduce several variants of how $\boldsymbol{\Sigma}$ is computed to verify the effectiveness of our proposed SigmaNet.

Firstly, we investigate whether SigmaNet is needed in its current form (as in taking features to produce the uncertainty variable), or if $\boldsymbol{\Sigma}$ could be learnt as the so-called free variable. To this end, we create a vector of parameters of size $\tau_{(0)}{\cdot}\tau_{(0)}$ which we register as one of parameters of the network (we backpropagate w.r.t. this parameter among others). We set $\tau_{(0)}$ to be the average integer of numbers of blocks over sequences. We then reshape this vector into $\tau_{(0)} \times \tau_{(0)}$ matrix and initialize with 0 ± 0.1 uniform noise. We then apply a 2D bilinear interpolation to the matrix to obtain $\boldsymbol{\Sigma}$ of desired size $\tau \times \tau'$, where τ and τ' are the number of temporal blocks for query and support samples, respectively. The $\tau \times \tau'$ matrix is then passed into the sigmoid function to produce the $\boldsymbol{\Sigma}$ matrix.

For classification of time series, we create a vector of parameters of size $t_{(0)}$ which we register as one of parameters of the network (we backpropagate w.r.t. this parameter among others). We set $\tau_{(0)}$ to be the average integer of numbers of time steps of input time series. We initialize that vector with 0 ± 0.1 uniform noise, and we then use a 1D bilinear interpolation to interpolate the vector into desired length τ. The interpolated vector is passed into the sigmoid function to generate $\boldsymbol{\sigma}_{\mathbf{x}}$ for the input sequence $\mathbf{x}$ of length τ. For sequence $\mathbf{x}'$ (exhaustive search via nearest neighbor) or $\boldsymbol{\mu}_c$ (via nearest centroid), we use exactly the same process to generate $\boldsymbol{\sigma}_{\mathbf{x}'}$ or $\boldsymbol{\sigma}_{\boldsymbol{\mu}_c}$ but of course they have their own vector of length $\tau_{(0)}$ that we minimize over. We obtain $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_{\mathbf{x}}^2 \mathbf{1}^\top + \mathbf{1} \boldsymbol{\sigma}_{\mathbf{x}'}^{\top\,2}$ (or $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_{\mathbf{x}}^2 \mathbf{1}^\top + \mathbf{1} \boldsymbol{\sigma}_{\boldsymbol{\mu}_c}^{\top\,2}$ if we use the nearest centroid), where squaring is performed in the element-wise manner.

Table 9: Comparisons of two different ways of generating $\boldsymbol{\Sigma}$ for few-shot action recognition. Evaluations on the NTU-60 dataset.

#classes	10	20	30	40	50
uDTW ($\boldsymbol{\Sigma}$ via the free variable)	54.1	56.5	61.0	64.1	68.0
uDTW ($\boldsymbol{\Sigma}$ via SigmaNet)	56.9	61.2	64.8	68.3	72.4

In conclusion, the above steps facilitate the direct minimization w.r.t. the variable tied with $\boldsymbol{\Sigma}$ instead of learning $\boldsymbol{\Sigma}$ through our SigmaNet whose input are encoded features *etc*. Tables 9 and 10 show that using SigmaNet is a much better choice than trying to infer the uncertainty by directly minimizing the free variable. The result is expected as SigmaNet learns to associate feature patterns of sequences with their uncertainty patterns. Minimizing w.r.t. the free variables cannot learn per se.

Table 10: Comparisons of two different ways of generating $\boldsymbol{\Sigma}$ for classification of time series. Evaluations on the UCR archive. K denotes the number of nearest neighbors used by the K nearest neighbors based classification.

	Nearest neighbor			**Nearest centroid**
	$K=1$	$K=3$	$K=5$	
uDTW ($\boldsymbol{\Sigma}$ via the free variable)	77.0	77.3	78.0	70.9
uDTW ($\boldsymbol{\Sigma}$ via SigmaNet)	80.0	81.2	83.3	72.2

E Additional Visualizations of Forecasting the Evolution of Time Series

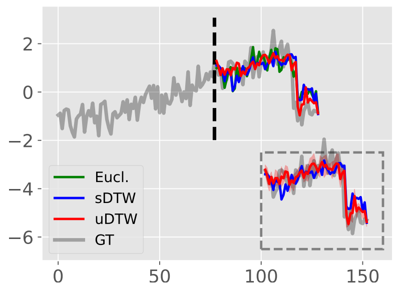
We provide additional visualizations of forecasting the evolution of time series in Figure 7. We notice that our uDTW produces predictions that are better aligned with the ground truth (see Fig. 7a). Moreover, our uDTW generates better shape of the predictions compared to sDTW, and the predictions from sDTW have more perturbations/fluctuations (see Fig. 7b). Quantitative results for the whole UCR archive can be found in the main paper.

F Additional Evaluations for Few-shot Action Recognition

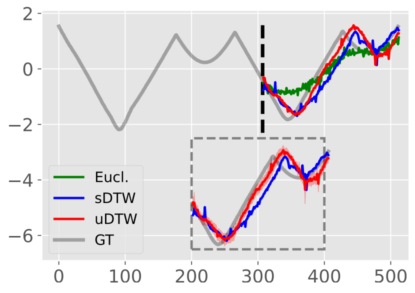
Table 11: uDTW derived under the Normal, Laplacian and Cauchy distributions. Evaluations of few-shot action recognition on small-scale datasets.

	Supervised			**Unsupervised**		
	MSR	3D Action Pairs	UWA 3D	MSR	3D Action Pairs	UWA 3D
TAP (HM) [62]	67.40	77.22	37.13	-	-	-
TAP (Lifted) [62]	65.20	78.33	34.80	-	-	-
TAP (Bino.) [62]	66.67	78.33	36.55	-	-	-
sDTW [55]	70.59	81.67	44.74	62.63	48.33	39.47
uDTW (Laplace)	72.24	82.89	45.64	**66.00**	**55.00**	41.22
uDTW (Cauchy)	70.88	**84.44**	45.03	65.12	50.32	40.50
uDTW (Normal)	**72.66**	83.33	**47.66**	65.00	52.22	**41.74**

We also evaluate our proposed uDTW versus sDTW on smaller datasets for both supervised and unsupervised settings. As uDTW was derived in Section 1.2 under modeling the MLE of the product of the Normal distributions, we investigate modeling each path $\boldsymbol{\Pi}_i$ by replacing the Normal distribution with the Laplace or Cauchy distributions. By applying MLE principles in analogy to Section 1.2, we arrive at $\beta\Omega_{\boldsymbol{\Pi}_i} + d^2_{\boldsymbol{\Pi}_i}$ for



(a) CBF



(b) ShapesAll

Fig. 7: Additional visualizations for forecasting the evolution of time series. Given the first part of a time series, we train the pipeline from Fig. 3b to predict the remaining part of the time series. We compare the use of the Euclidean, sDTW or uDTW distances within the pipeline. We use CBF and ShapesAll in UCR archive, and display the prediction obtained for the given test sample with either of these 3 distances, and the ground truth (GT). Oftentimes, we observe that uDTW helps predict the sudden changes well. (a) Our uDTW aligns well with the ground truth compared to sDTW. (b) Our uDTW generates better shape of prediction compared to sDTW (for example note the red curve following much closer the rising gray slope). Quantitative results of MSE and 'shape' metrics for the whole UCR archive are given in the main paper.

Table 12: uDTW derived under the Normal, Laplacian and Cauchy distributions. Evaluations of few-shot action recognition on the large-scale NTU-60 dataset.

#classes	10	20	30	40	50
			Supervised		
sDTW(baseline) [55]	53.7	56.2	60.0	63.9	67.8
uDTW(Cauchy)	56.1	61.1	62.9	68.3	69.9
uDTW(Laplace)	55.3	59.2	63.3	67.7	70.3
uDTW(Normal)	56.9	61.2	64.8	68.3	72.4
			Unsupervised		
sDTW(baseline) [55]	35.6	45.2	53.3	56.7	61.7
uDTW(Cauchy)	36.7	47.9	54.9	57.3	63.3
uDTW(Laplace)	36.2	48.2	54.3	57.8	63.1
uDTW(Normal)	37.0	48.3	55.3	58.0	63.3

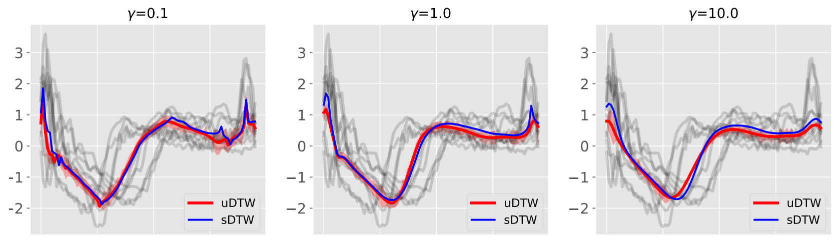
i. Laplace: $\sum_{(m,n)\in\boldsymbol{\Pi}_i} \beta\log(\sigma_{mn}) + \frac{\|\boldsymbol{\psi}_m-\boldsymbol{\psi}'_n\|_1}{\sigma_{mn}}$;

ii. Cauchy: $\sum_{(m,n)\in\boldsymbol{\Pi}_i} \beta\log(\sigma_{mn}) + \log\left(1+\frac{\|\boldsymbol{\psi}_m-\boldsymbol{\psi}'_n\|_2^2}{\sigma_{mn}^2}\right)$.

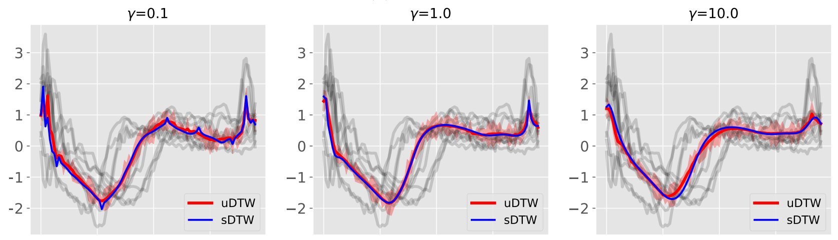
Table 11 shows that uDTW achieves better performance than sDTW, and the Laplace distribution is performing particularly well on the unsupervised few-shot action recognition. Table 12 shows that uDTW based on the Normal distribution is overall better than other distributions on large-scale datasets such as NTU-60. For this very reason we use uDTW based on the Normal distribution.

G Additional Visualizations on Barycenters

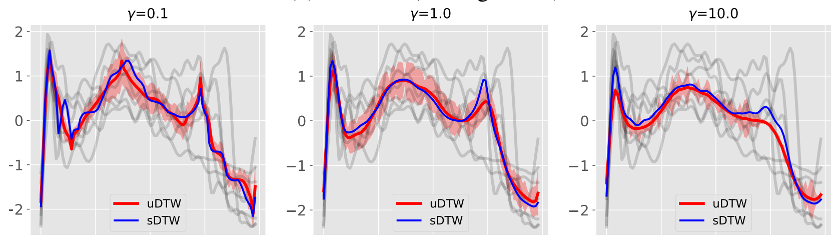
Figure 8 shows more visualizations of barycenters of time series. With our SigmaNet, we obtain much better barycenters with our uDTW compared to sDTW.



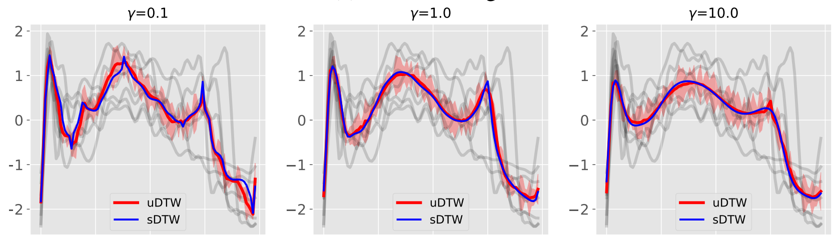
(a) ECG200



(b) ECG200 (our SigmaNet)



(c) Medical Images



(d) Medical Images (our SigmaNet)

Fig. 8: Comparison of barycenters based on our uDTW *vs.* sDTW. We visualize uncertainty around the barycenters in red color for uDTW. Our uDTW with SigmaNet generates reasonable barycenters even when higher γ values are used, *e.g.*, $\gamma = 10.0$. Higher γ value leads to smooth barycenters but introducing higher uncertainty.

H Network Configuration and Training Details

Below we provide the details of network configuration and training process.

H.1 Skeleton Data Preprocessing

Before passing the skeleton sequences into MLP and a simple linear graph network (*e.g.*, S^{2}GC), we first normalize each body joint w.r.t. to the torso joint $\mathbf{v}_{f,c}$:

$$\mathbf{v}'_{f,i} = \mathbf{v}_{f,i} - \mathbf{v}_{f,c}, \tag{23}$$

where f and i are the index of video frame and human body joint respectively. After that, we further normalize each joint coordinate into [-1, 1] range:

$$\hat{\mathbf{v}}_{f,i}[j] = \frac{\mathbf{v}'_{f,i}[j]}{\max([\text{abs}(\mathbf{v}'_{f,i}[j])]_{f\in\mathcal{I}_\tau, i\in\mathcal{I}_J})}, \tag{24}$$

where j is for selection of the x, y and z axes, τ is the number of frames and J is the number of 3D body joints per frame.

For the skeleton sequences that have more than one performing subject, (i) we normalize each skeleton separately, and each skeleton is passed to MLP for learning the temporal dynamics, and (ii) for the output features per skeleton from MLP, we pass them separately to the graph neural network, *e.g.*, two skeletons from a given video sequence will have two outputs obtained from the graph neural network, and we aggregate the outputs through average pooling before passing to sDTW or uDTW.

H.2 Network Configuration

SigmaNet. It is composed of an FC layer and a scaled sigmoid function which translate the learned features of either actions or time series into desired $\boldsymbol{\Sigma}$. The input to FC is of the size of feature dimension (depends on the encoder) and the output is a scalar. SigmaNet with the scaled sigmoid function can be defined as:

$$\sigma(\boldsymbol{\psi}) = \frac{\kappa}{1+\exp(-\text{FC}(\boldsymbol{\psi}))} + \eta, \tag{25}$$

where $\eta > 0$ is the offset and $\kappa \geq 0$ is the maximum magnitude of sigmoid. For an entire sequence with τ blocks, the SigmaNet produces vector $\boldsymbol{\sigma}_{\mathbf{x}}$ for sequence $\mathbf{x}$ and $\boldsymbol{\sigma}_{\mathbf{x}'}$ for sequence $\mathbf{x}'$ (we concatenate per-block scalars to form these vectors), and we typically obtain $\boldsymbol{\Sigma} = \boldsymbol{\sigma}_{\mathbf{x}}^2 \mathbf{1}^\top + \mathbf{1}\boldsymbol{\sigma}_{\mathbf{x}'}^{2\,\top}$.

Forecasting of the evolution of time series. The MLP for this task consists of two FC layers with a tanh layer in between. The input to the first FC layer is t and output size is t', and after the tanh layer, the input to the second FC layer is t' and output $(\tau - t)$ dimensional prediction. We set $t' = 30$ or 50 depending on the length of time series in each dataset.

Few-shot action recognition. Given the temporal block size M (the number of frames in a block) and desired output size d, the configuration of the 3-layer MLP unit is: FC ($3M \rightarrow 6M$), LayerNorm (LN) as in [11], ReLU, FC ($6M \rightarrow 9M$), LN, ReLU, Dropout (for smaller datasets, the dropout rate is 0.5; for large-scale datasets, the dropout rate is 0.1), FC ($9M \rightarrow d$), LN. Note that M is the temporal block size and d is the output feature dimension per body joint. We set $M = 10$ for experiments.

For the encoding network, let us take the query input $\mathbf{X} \in \mathbb{R}^{3\times J\times M}$ for the temporal block of length M as an example, where 3 indicates that Cartesian coordinates (x, y, z) were used, and J is the number of body joints. As alluded to earlier, we obtain $\widehat{\mathbf{X}}^T = \text{MLP}(\mathbf{X}; \mathcal{P}_{MLP}) \in \mathbb{R}^{d\times J}$.

Subsequently, we employ a simple linear graph network, S^{2}GC from Section H.3, and the transformer encoder [11] which consists of alternating layers of Multi-Head Self-Attention (MHSA) and a feed-forward MLP (two FC layers with a GELU non-linearity between them). LayerNorm (LN) is applied before every block, and residual connections after every block. Each block feature matrix $\widehat{\mathbf{X}} \in \mathbb{R}^{J\times d}$ encoded by a simple linear graph network S^{2}GC (without learnable $\boldsymbol{\Theta}$) is then passed to the transformer. Similarly to the standard transformer, we prepend a learnable vector $\mathbf{y}_{\text{token}} \in \mathbb{R}^{1\times d}$ to the sequence of block features $\widehat{\mathbf{X}}$ obtained from S^{2}GC, and we also add the positional embeddings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(1+J)\times d}$ based on the sine and cosine functions (standard in transformers) so that token $\mathbf{y}_{\text{token}}$ and each body joint enjoy their own unique positional encoding. We obtain $\mathbf{Z}_0 \in \mathbb{R}^{(1+J)\times d}$ which is the input in the following backbone:

$$\mathbf{Z}_0 = [\mathbf{y}_{\text{token}}; \text{S}^2\text{GC}(\widehat{\mathbf{X}})] + \mathbf{E}_{\text{pos}}, \tag{26}$$

$$\mathbf{Z}'_k = \text{MHSA}(\text{LN}(\mathbf{Z}_{k-1})) + \mathbf{Z}_{k-1}, \quad k = 1, ..., L_{\text{tr}} \tag{27}$$

$$\mathbf{Z}_k = \text{MLP}(\text{LN}(\mathbf{Z}'_k)) + \mathbf{Z}'_k, \quad k = 1, ..., L_{\text{tr}} \tag{28}$$

$$\mathbf{y}' = \text{LN}\big(\mathbf{Z}^{(0)}_{L_{\text{tr}}}\big) \quad \text{where} \quad \mathbf{y}' \in \mathbb{R}^{1\times d} \tag{29}$$

$$f(\mathbf{X}; \mathcal{P}) = \text{FC}(\mathbf{y}'^T; \mathcal{P}_{FC}) \quad \in \mathbb{R}^{d'}, \tag{30}$$

where $\mathbf{Z}^{(0)}_{L_{\text{tr}}}$ is the first d dimensional row vector extracted from the output matrix $\mathbf{Z}_{L_{\text{tr}}}$ of size $(J+1) \times d$ which corresponds to the last layer L_{tr} of the transformer. Moreover, parameter L_{tr} controls the depth of the transformer, whereas $\mathcal{P} \equiv [\mathcal{P}_{MLP}, \mathcal{P}_{S^2GC}, \mathcal{P}_{Transf}, \mathcal{P}_{FC}]$ is the set of parameters of EN. In case of S^{2}GC, $|\mathcal{P}_{S^2GC}| = 0$ because we do not use their learnable parameters $\boldsymbol{\Theta}$ (*i.e.*, think $\boldsymbol{\Theta}$ is set as the identity matrix in Eq. (31)).

We can define now a support feature map as $\boldsymbol{\Psi}' = [f(\boldsymbol{X}_1; \mathcal{P}), ..., f(\boldsymbol{X}_{\tau'}; \mathcal{P})] \in \mathbb{R}^{d'\times\tau'}$ for τ' temporal blocks, and the query map $\boldsymbol{\Psi}$ accordingly.

The hidden size of our transformer (the output size of the first FC layer of the MLP depends on the dataset. For smaller datasets, the depth of the transformer is $L_{\text{tr}} = 6$ with 64 as the hidden size, and the MLP output size is $d = 32$ (note that the MLP which provides $\widehat{\mathbf{X}}$ and the MLP in the transformer must both have the same output size). For NTU-60, the depth of the transformer is $L_{\text{tr}} = 6$, the hidden size is 128 and the MLP output size is $d = 64$. For NTU-120, the depth of the transformer is $L_{\text{tr}} = 6$, the hidden size is 256 and the MLP size is $d = 128$. For Kinetics-skeleton, the depth for the transformer is $L_{\text{tr}} = 12$, hidden size is 512 and the MLP output size is $d = 256$.

The number of Heads for the transformer of smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton is set as 6, 12, 12 and 12, respectively.

The output sizes d' of the final FC layer are 50, 100, 200, and 500 for the smaller datasets, NTU-60, NTU-120 and Kinetics-skeleton, respectively.

H.3 Linear Graph Network (S^{2}GC)

Based on a modified Markov Diffusion Kernel, Simple Spectral Graph Convolution (S^{2}GC) is the summation over l-hops, $l=1, ..., L$. The output of S^{2}GC is given as:

$$\mathbf{\Phi}_{\mathrm{S^2GC}} = \frac{1}{L}\sum_{l=1}^{L}((1-\alpha)\mathbf{S}^l\mathbf{X}+\alpha\mathbf{X})\mathbf{\Theta}, \tag{31}$$

where $L \geq 1$ is the number of linear layers and $\alpha \geq 0$ determines the importance of self-loop of each node (we use their default setting $\alpha=0.05$ and $L=6$). Choice of other graph embeddings are possible, including contrastive models COLES [67] or COSTA [66], adversarial Fisher-Bures GCN [63] or GCNs with rectifier attention [65]. One may also use kernels on 3D body joints as in [64] or even use CNN to encode 3D body joints as COLTRANE [60].

H.4 K-NN classifier with SoftMax

For the K-NN classifier, instead of using K best weights proportional to the inverse of the distance from the query sample $\mathbf{x}^*$ to the closest samples $\mathbf{x}_n$ (as is done in the soft-DTW paper [55]) and expressed by

$$w(\mathbf{x}_n|\mathbf{x}^*) = \frac{1}{d^2\big(\mathbf{x}^*, \mathbf{x}_n\big)}, \tag{32}$$

we weigh the neighbors $\mathbf{x}_n$ of $\mathbf{x}^*$ using

$$w(\mathbf{x}_n|\mathbf{x}^*) = \frac{\exp\big(-\frac{1}{\gamma''}d^2\big(\mathbf{x}^*, \mathbf{x}_n\big)\big)}{\sum_{n'\in\mathcal{N}(\mathbf{x}^*;K)}\exp\big(-\frac{1}{\gamma''}d^2\big(\mathbf{x}^*, \mathbf{x}_{n'}\big)\big)} \tag{33}$$

such that $\mathcal{N}(\mathbf{x}^*; K)$ produces K nearest samples $\mathbf{x}_{n'}$ of $\mathbf{x}^*$ according to distance $d(\cdot,\cdot)$, *e.g.*, the Euclidean distance, sDTW or uDTW. Parameter $\gamma'' > 0$ (in our case, we set $\gamma''=6$) further controls the impact of each sample $\mathbf{x}_n$ on the classifier based on the bell shape of Radial Basis Function in the above equation.

Table 13 shows the comparisons. We notice that the use of SoftMax in the K-NN classifier improves the performance for all the methods when $K=3$ and $K=5$.

H.5 Training Details

For both time series and few-shot action recognition pipelines, the weights are initialized with the normal distribution (zero mean and unit standard deviation). We use 1e-3 for the learning rate, and the weight decay is 1e-6. We use the SGD optimizer.

Table 13: Classification accuracy (mean±std) on UCR archive using nearest neighbor. K denotes the number of nearest neighbors in the K-NN classifier. Highlighted rows are the based on SoftMax from Eq. (33). Non-highlighted rows are based on Eq. (32).

	Nearest neighbor		
	$K=1$	$K=3$	$K=5$
Euclidean	71.2±17.5	69.5±18.0	67.5±17.6
Euclidean (SoftMax)	71.2±17.5	72.3±18.1	73.0±16.7
DTW [54]	74.2±16.6	72.8±16.9	71.4±16.8
DTW [54] (SoftMax)	74.2±16.6	75.0±17.0	75.4±15.8
sDTW [55]	76.2±16.6	74.0±15.6	70.5±17.6
sDTW [55] (SoftMax)	76.2±16.6	77.2±15.9	78.0±16.5
sDTW div. [53]	78.6±16.2	76.5±16.4	74.8±15.8
sDTW div. [53] (SoftMax)	78.6±16.2	79.5±16.7	80.1±16.5
uDTW	80.0±15.0	78.0±15.8	76.2±15.0
uDTW (SoftMax)	80.0±15.0	81.2±17.8	83.3±16.2

For time series, we set the training epochs to 30, 50 and 100 depending on the dataset in the UCR archive (due to many datasets, the epoch settings will be provided in the code directly).

For few-shot action recognition, we set the number of training episodes to 100K for NTU-60, 200K for NTU-120, 500K for 3D Kinetics-skeleton, 10K for small datasets such as UWA 3D Multiview Activity II.

I Hyperparameters Evaluation

In this section, we evaluate the impact of key hyperparameters. Remaining hyperparameters are obtained through Hyperopt [52] for hyperparameter search on the validation set.

I.1 Evaluation of $\boldsymbol{\Sigma}$

We compare results given different formulations of $\boldsymbol{\Sigma}$ in Tables 14 and 15. We notice that on smaller datasets, it is hard to determine which variant of $\boldsymbol{\Sigma}$ is better (as these earlier datasets have fewer limited reliable skeletons compared to the new datasets). However, on bigger datasets, $\boldsymbol{\Sigma}=\boldsymbol{\sigma}_{\psi}^2\mathbf{1}^\top+\mathbf{1}\boldsymbol{\sigma}_{\psi'}^{\top\,2}$ performs the best in all cases; thus we choose this formulation of $\boldsymbol{\Sigma}$ for large-scale datasets.

I.2 Evaluation of κ and η of SigmaNet

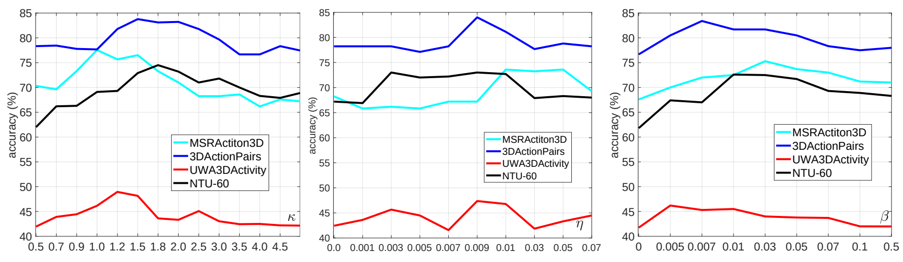
Figures 9a and 9b show the impact of κ and η of the scaled sigmoid function in SigmaNet on both small-scale datasets and the large-scle NTU-60 dataset. We notice that $\kappa=1.5$ performs the best on the three small-scale datasets and $\kappa=1.8$ works the best on

Table 14: Evaluation of different variants of $\boldsymbol{\Sigma}$ computation on small-scale datasets (supervised few-shot action recognition). Operator $\odot$ is the Hadamart product.

	$\boldsymbol{\sigma}_\psi\mathbf{1}^\top\odot\mathbf{1}\boldsymbol{\sigma}_{\psi'}^\top$	$\boldsymbol{\sigma}_\psi^2\mathbf{1}^\top\odot\mathbf{1}\boldsymbol{\sigma}_{\psi'}^{\top\,2}$	$\boldsymbol{\sigma}_\psi\mathbf{1}^\top+\mathbf{1}\boldsymbol{\sigma}_{\psi'}^\top$	$\boldsymbol{\sigma}_\psi^2\mathbf{1}^\top+\mathbf{1}\boldsymbol{\sigma}_{\psi'}^{\top\,2}$
MSR Action 3D	**72.32**	68.51	70.59	69.20
3D Action Pairs	82.78	80.56	82.22	**85.00**
UWA 3D Activity	43.86	**45.91**	**45.91**	45.03

Table 15: Evaluation of different variants of $\boldsymbol{\Sigma}$ computation on the large-scale NTU-60 dataset (supervised few-shot action recognition).

#classes	$\boldsymbol{\sigma}_\psi\mathbf{1}^\top\odot\mathbf{1}\boldsymbol{\sigma}_{\psi'}^\top$	$\boldsymbol{\sigma}_\psi^2\mathbf{1}^\top\odot\mathbf{1}\boldsymbol{\sigma}_{\psi'}^{\top\,2}$	$\boldsymbol{\sigma}_\psi\mathbf{1}^\top+\mathbf{1}\boldsymbol{\sigma}_{\psi'}^\top$	$\boldsymbol{\sigma}_\psi^2\mathbf{1}^\top+\mathbf{1}\boldsymbol{\sigma}_{\psi'}^{\top\,2}$
10	56.6	56.0	55.6	**56.9**
20	60.4	61.0	61.2	**61.2**
30	64.2	64.1	63.5	**64.8**
40	68.1	66.9	67.2	**68.3**
50	72.0	72.3	72.0	**72.4**



(a) (b) (c)

Fig. 9: Evaluation of (a) κ which controls the maximum magnitude and (b) η offset from Eq. (25) in SigmaNet and (c) β from Eq. (17). Note that $\beta=0$ means no regularization term of uDTW in use. We notice that with the regularization term added to the uDTW, the overall performance is improved.

NTU-60. We choose $\kappa=1.8$ in the experiments for the large-scale datasets. Moreover, $\eta\in[0.003, 0.01]$ works better on NTU-60, and on the small-scale datasets, $\eta=0.01$ achieves the best performance; thus we choose $\eta=0.01$ for the experiments.

I.3 Evaluation of β

Figure 9c shows the evaluations of β for both small-scale datasets and NTU-60. Firstly, note that $\beta=0$ means lack of the regularization term of uDTW, which immediately causes the performance deterioration. As shown in the figure, $\beta=0.05$ performs the best on UWA 3D Activity, $\beta=0.03$ achieves the best performance on MSR Action 3D and $\beta=0.007$ works the best on 3D Action Pairs dataset. We use the corresponding best β values for the smaller datasets. On NTU-60, $\beta \in [0.01, 0.05]$ performs the best compared to other β values, thus we choose $\beta = 0.03$ for the experiments on all large-scale datasets.

I.4 Evaluation of warping window width

Table 16 on ECGFiveDays (from UCR) and NTU-60 (50-class, supervised / unsup. settings) shows that uDTW does not break quicker than sDTW (window size is parametrized by r). Very small r may preclude backpropagating through some paths (of large distance). For such paths 'beyond window', learning uncertainty is limited but this is normal. For similar reasons, choosing the right window size is required by other DTW variants too. Also, if r is very large, large uncertainty score may decrease the distance on multitude of paths by downweighting parts of paths (could lead to strange matching) but as the uncertainty is aggregated into the regularization penalty, this penalty prevents uDTW from unreasonable solutions. Lack of regularization penalty (*w/o reg.*) affects the most the unsupervised few-shot learning, while supervised loss can still drive SigmaNet to produce meaningful results.

Table 16: Experimental results on ECGFiveDays (from UCR) and NTU-60 (50-class, supervised / unsup. settings) for different warping window widths.

		$\gamma = 0.001$			$\gamma = 0.01$			$\gamma = 0.1$			$\gamma = 1$		
		r=1.0	r=3.0	r=5.0	r=1.0	r=3.0	r=5.0	r=1.0	r=3.0	r=5.0	r=1.0	r=3.0	r=5.0
ECG FiveDays	sDTW	83.4	82.8	82.0	79.7	76.8	77.8	75.4	69.0	65.3	62.5	61.7	60.2
	uDTW	85.6	91.2	81.0	93.5	82.8	80.6	79.7	73.9	67.3	69.0	65.3	62.5
	uDTW *w/o reg.*	75.4	74.0	69.0	79.7	77.9	76.8	65.3	62.5	61.5	61.2	62.0	60.2
NTU-60 (sup.)	sDTW	65.7	64.7	64.8	65.2	67.8	63.9	60.0	58.9	54.3	54.0	52.2	52.3
	uDTW	71.5	71.0	70.0	72.4	72.4	70.0	68.3	66.7	67.8	65.7	64.8	66.8
	uDTW *w/o reg.*	66.3	65.0	65.5	66.4	68.0	65.2	62.0	59.2	55.0	52.0	52.0	51.2
NTU-60 (unsup.)	sDTW	56.7	53.2	50.0	61.7	61.7	60.0	54.4	52.5	52.1	48.3	45.2	40.9
	uDTW	61.0	61.5	60.7	63.3	63.0	62.5	59.2	59.0	57.3	58.0	57.2	55.7
	uDTW *w/o reg.*	50.1	49.3	47.0	55.3	54.0	51.3	44.1	42.0	40.7	42.3	40.1	35.6

References

52. Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D.: Hyperopt: a python library for model selection and hyperparameter optimization. Computational Science & Discovery **8**(1), 014008 (2015), `http://stacks.iop.org/1749-4699/8/i=1/a=014008`
53. Blondel, M., Mensch, A., Vert, J.P.: Differentiable divergences between time series. In: Banerjee, A., Fukumizu, K. (eds.) Proceedings of The 24th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 130, pp. 3853–3861. PMLR (13–15 Apr 2021)

54. Cuturi, M.: Fast global alignment kernels. In: International Conference on Machine Learning (ICML) (2011)
55. Cuturi, M., Blondel, M.: Soft-dtw: a differentiable loss function for time-series. In: International Con- ference on Machine Learning (ICML) (2017)
56. Dau, H.A., Keogh, E., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G.: The UCR Time Series Classification Archive (October 2018), https://www.cs.ucr.edu/~eamonn/time_series_data_2018/
57. Li, W., Zhang, Z., Liu, Z.: Action Recognition Based on A Bag of 3D Points. In: CVPR. pp. 9–14 (2010)
58. Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L.Y., Kot, A.C.: Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019). https://doi.org/10.1109/TPAMI.2019.2916873
59. Oreifej, O., Liu, Z.: HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In: CVPR. pp. 716–723 (2013)
60. Prabowo, A., Koniusz, P., Shao, W., Salim, F.D.: COLTRANE: convolutional trajectory network for deep map inference. In: Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys 2019, New York, NY, USA, November 13-14, 2019. pp. 21–30. ACM (2019). https://doi.org/10.1145/3360322.3360853
61. Rahmani, H., Mahmood, A., Huynh, D.Q., Mian, A.: HOPC: Histogram of Oriented Principal Components of 3D Pointclouds for Action Recognition. In: ECCV. pp. 742–757 (2014)
62. Su, B., Wen, J.R.: Temporal alignment prediction for supervised representation learning and few-shot sequence classification. In: International Conference on Learning Representations (2022)
63. Sun, K., Koniusz, P., Wang, Z.: Fisher-bures adversary graph convolutional networks. Conference on Uncertainty in Artificial Intelligence **115**, 465–475 (2019)
64. Tas, Y., Koniusz, P.: Cnn-based action recognition and supervised domain adaptation on 3d body skeletons via kernel feature maps. The British Machine Vision Conference (BMVC) (2018)
65. Zhang, Y., Zhu, H., Meng, Z., Koniusz, P., King, I.: Graph-adaptive rectified linear unit for graph neural networks. In: Proceedings of the ACM Web Conference 2022. p. 1331–1339. WWW '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3485447.3512159
66. Zhang, Y., Zhu, H., Song, Z., Koniusz, P., King, I.: Costa: Covariance-preserving feature augmentation for graph contrastive learning. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (2022), https://doi.org/10.1145/3534678.3539425
67. Zhu, H., Sun, K., Koniusz, P.: Contrastive laplacian eigenmaps. Advances in Neural Information Processing Systems **34** (2021)