# TSMix: time series data augmentation by mixing sources

**Luke Darlow[1], Martin Asenov[1], Artjom Joosen[1], Qiwen Deng[1,2], Jianfeng Wang[3], Adam Barker[1,4]**

1. Edinburgh Research Centre, Central Software Institute, Huawei.
2. School of Informatics, University of Edinburgh
3. Hangzhou Research Centre, Central Software Institute, Huawei
4. School of Computer Science, University of St Andrews

## Introduction

- Data augmentation is a common strategy for improving generalisation capability of ML models such as neural networks.
- Data augmentation **alters existing data to produce realistic perspectives on data that might be expected during inference**.
- Data augmentation is widespread for image data because it is relatively trivial to construct automatic augmentation strategies (e.g., horizontal flipping, colour augmentation, mixup [1], etc.).
- Data **augmentation for time series is challenging because an individual datum is not an independent quantity, but a variable dependent on past data**. Individual time series data points are not necessarily independent and identically distributed (i.i.d.).
- In this paper we **propose and evaluate the idea that an individual time series (univariate or multivariate) can be considered as an i.i.d. sample from a wider set of time series**. We evaluate this idea through TSMix, a simple **method to combine multiple univariate time series to increase training dataset size and thereby improve forecast model generalisation.**

## Use-case: FaaS

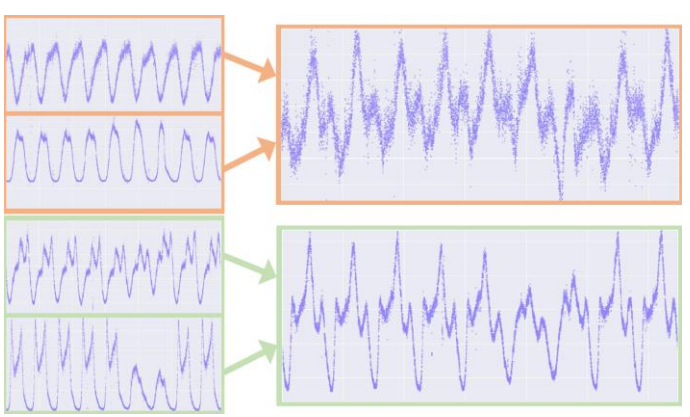
- **TSMix is motivated by and applied to serverless computing**, a cloud computing model where computational resources are transparently allocated by a cloud provider on-demand for the user.
- Function as a Service (FaaS) platforms such as Huawei FunctionGraph, Microsoft Azure Functions and Google Cloud Functions allow customers to run code in the form of single stateless event-driven functions. FaaS platforms execute thousands of concurrent function requests by different users.
- **These individual functions each result in multivariate time series**, allowing the collection of functions to be seen as a large set of independent time series.
- This work considers function requests, which are the incoming function request invocations per-minute as the common univariate time series. **FaaS requests of top functions tend to have daily periodicity and may tend to resolve into similar patterns.** Therefore, it is possible to produce qualitatively similar pseudo-function datasets by simply averaging two or more real time series from different functions.

## Algorithm

- N univariate time series (e.g. time series from different function calls in FaaS framework) are **combined by averaging their standardised versions (i.e., zero mean and unit standard deviation).**
- The final mixed version is **unnormalized by the average of incoming means and standard deviation** to ensure scale invariance with respect to the input series.

**Input:** $X$ = list of $N$ time series, length $L$
**Output:** $y$ = mixed time series, length $L$

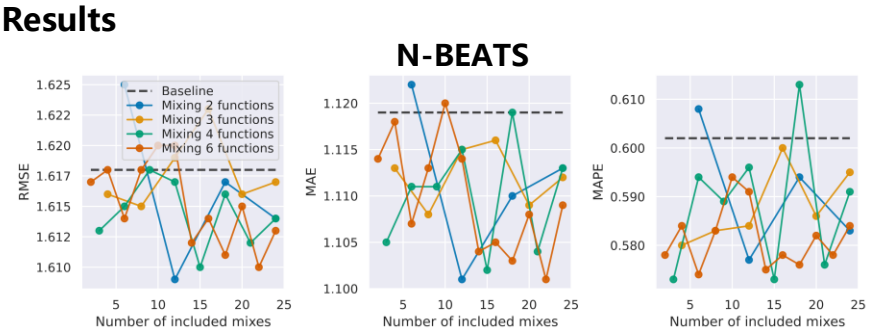
$Y :=$ vector of zeros, length $L$
$\mu := 0$
$\sigma := 0$
**for** $x_n$ *in* $X$ **do**
  $\mu_n = \text{mean}(x)$
  $\sigma_n = \text{std}(x)$
  $y+ = (x_n - \mu_n)/\sigma_n$
  $\mu+ = \mu_n$
  $\sigma+ = \sigma_n$
$\mu = \mu/N$
$\sigma = \sigma/N$
$y = (y/N) \times \sigma + \mu$

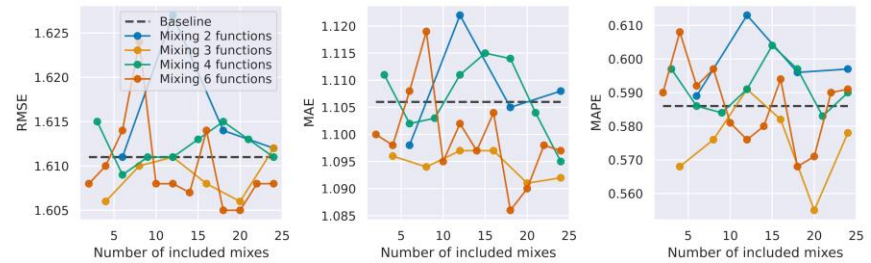


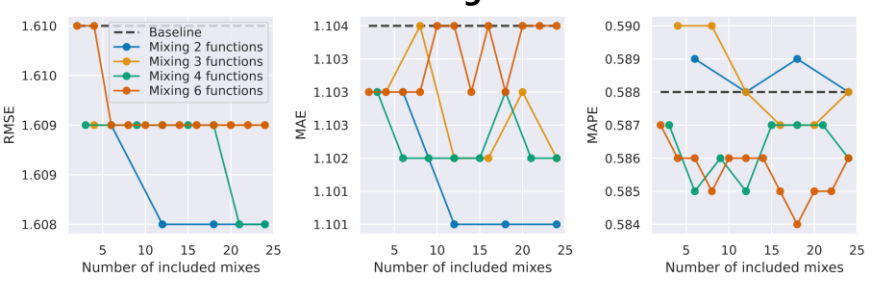
## Experimental setup

- Used top 12 functions from Azure 2019 (by median) [2]
- 14 days per function (9 for training and 5 for testing)
- Predictions over the 5 test days computed autoregressively
- Evaluate 3 models: N-HiTS [3], N-BEATS [4], linear regression
- The following 31 experiments were conducted for each model:
  - Training on original 12 functions only.
  - For N=2 we ran 4 experiments whereby we created 6, 12, 18, and 24 mixed pseudo functions. The original 12 functions were always included. We sampled mixes such that they were always unique. Increments of 6 are needed for N=2 in order to ensure equal sampling of the original 12 functions.
  - Other mixing combinations for N=3, 4, 6

## Results




## Future work

- Mixing different functions in this way can be expanded to include an informed version of TSMix using mutual information or cosine similarity as a metric to combine dissimilar functions.
- Decomposition into trend, seasonal, and residual components and applying TSMix to those and then recombining may also be a straightforward yet promising option.

## References

1. Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).
2. Mohammad Shahrad, Rodrigo Fonseca, Inigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider. In 2020 *USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 205–218.
3. Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. 2022. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886* (2022).
4. Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.