HEDGING IS NOT ALL YOU NEED: A SIMPLE BASELINE FOR ONLINE LEARNING UNDER HAPHAZARD INPUTS

Himanshu Buckchash
UiT The Arctic University of Norway
Tromsø, Norway
himanshu.buckchash@uit.no

Momojit Biswas
Jadavpur University
Kolkata, India
mb16biswas@gmail.com

Rohit Agarwal
UiT The Arctic University of Norway
Tromsø, Norway
rohit.agarwal@uit.no

Dilip K. Prasad
UiT The Arctic University of Norway
Tromsø, Norway
dilip.prasad@uit.no

ABSTRACT

Handling haphazard streaming data, such as data from edge devices, presents a challenging problem. Over time, the incoming data becomes inconsistent, with missing, faulty, or new inputs reappearing. Therefore, it requires models that are reliable. Recent methods to solve this problem depend on a hedging-based solution and require specialized elements like auxiliary dropouts, forked architectures, and intricate network design. We observed that hedging can be reduced to a special case of weighted residual connection; this motivated us to approximate it with plain self-attention. In this work, we propose HapNet, a simple baseline that is scalable, does not require online backpropagation, and is adaptable to varying input types. All present methods are restricted to scaling with a fixed window; however, we introduce a more complex problem of scaling with a variable window where the data becomes positionally uncorrelated, and cannot be addressed by present methods. We demonstrate that a variant of the proposed approach can work even for this complex scenario. We extensively evaluated the proposed approach on five benchmarks and found competitive performance.

Keywords online learning · time series · neural network · deep learning · self-attention · fault tolerance · sensors · IoT

1 Introduction

Sensors are heavily used in a wide number of time-series, industrial and measurement problems, such as for energy management, industrial IoT, smart cities, agriculture, healthcare, smart homes, fraud detection, autonomous systems etc. However, the reliability of the models (based on these sensor inputs) or the predictions of these models is only as good as the reliability of data from the sensors. Many a times, due to faults, the sensors do not transmit data. In such scenarios, the input features to a model may vary, leading to the problem of variation in input size. Similarly, in data collection under a multi-entity environment, the failure of some actors may directly influence the model's input feature space. These challenging scenarios where the input space does not stay fixed, can be categorized under a single term called *haphazard inputs* [1]. This category of problems have not been typically studied until recent past, where Agarwal *et al.* have tried to formalize the problem domain, identifying important shared characteristics among problems [1, 2]. Making better models to address haphazard inputs is an important area of research and bears significant economic value as well.

The typical machine learning models assume that the dimensions of the input space are fixed and do not change during training or inference. However, haphazard inputs, challenge this assumption and force us to rethink about typical machine learning models. This is a very new field and a major contribution has been made by Agarwal *et al.* [1, 2]. Their idea works by making per input models that interact with each other based on the hedging algorithm [3]. Hedging regulates the weight/contribution of each input towards model's final prediction. However, we have

identified that hedging based approaches have multiple disadvantages like, they require to make models for each input, which is challenging to generalize and scale to other modalities such as images or videos, they require online gradient descent algorithm which adds to the complexity, they require more number of hyperparameters which makes them hyperparameter-sensitive, moreover, they are harder to implement and highly complex leading to limited ease of access for different applications.

In this work, we propose to model this problem with a simpler approach based on self-attention [4]. We first show that hedging can be approximated as weighted residual network. Using this intuition we design a self-attention based model called HapNet, which performs competitively with the current state-of-the-art models, however is quite straight forward to implement and thus have better generalizability and scalability. We study the haphazard inputs problem in detail and propose an even challenging and realistic version of the problem where the inputs are positionally uncorrelated. To address this challenging case, we extend the proposed HapNet model to HapNetPU model, which incorporates the idea of feed-back loop. In order to demonstrate the effectiveness of the proposed approach, we have used 5 benchmark datasets from different time-series problem domains. **Novelty.** In the spirit of Occam's razor principle, the novelty of our work lies in providing a simpler yet effective approach to solve haphazard inputs based problems in both positionally correlated and uncorrelated scenarios.

Contributions. **(a)** Unlike AuxDrop or the other hedging based methods [3, 2], the proposed method does not need to build per feature model. It uses joint optimization which is a more general approach and may lead to enhancement in sharing and mutual learning of feature space for general problems of different modalities and larger input sizes like images or videos [5, 6]. **(b)** The proposed approach uses simple self-attention and therefore is free from online gradient descent based backpropagation. Furthermore, our work also contributes by showing that for one dimensional inputs, the embedding layer is not required in transformers, and the input can directly serve as its own embedding. This is in contrast to the previous findings [7]. **(c)** We contribute a more challenging and realistic case of haphazard inputs, where the features become more positionally uncorrelated than the typical case. We also propose HapNetPU model to tackle this challenging scenario. Note that HapNetPU is the only model that could work for this kind of problems. **(d)** We compare the proposed models with state-of-the-art models and perform extensive ablation study.

Related work. Earlier works on online machine learning utilized basic machine learning models like k nearest neighbors [8], decision trees [9], support vector machines [10], fuzzy models [11], neural networks [12] etc. Recent methods based on deep learning, like [3], have also shown competitive performance for online machine learning. However, the main challenge with these methods remain that they cannot be directly employed for non-fixed input space problems. Some other methods based on incremental learning [13] or online learning [14], have also attempted to address these problems, however, they are not based on deep learning. In this direction, important contributions have been made by Agarwal *et al.* [1, 2]. These works mainly rely on a learning mechanism called "hedging" [3]. Our work builds upon these hedging based methods.

2 Proposed Approach

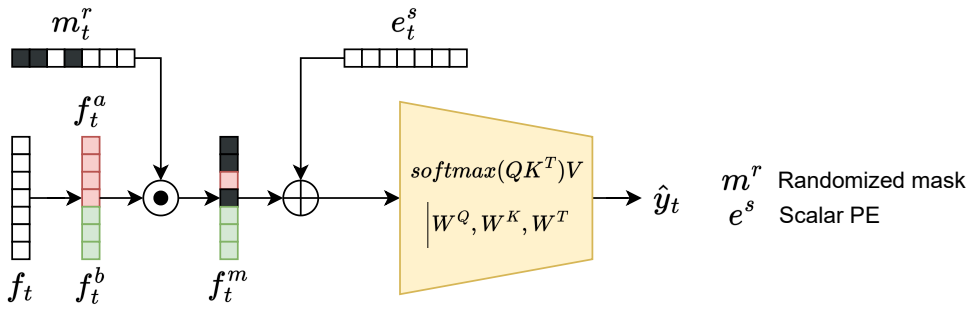
We use a self-attention based mechanism to capture the correlations among the input features. During training, each input feature at time t is randomly masked and used for training. The proposed approach is straight forward as shown in Fig. 1a. We first formally explain the haphazard input problem. Then the role of hedging is discussed, followed by the proposed approach.

Task formulation. Assuming a faulty data source or a sensor, generating the input data stream $\mathcal{D}$, the objective of online learning with haphazard inputs is to correctly label the sample $f_t \in \mathcal{D}$ arriving at time step t, and simultaneously update the model parameters with the gradient of prediction loss at t. We assume that since the data source is faulty, a subset of the sample f_t is called auxiliary, denoted by f_t^a, and contains reliable information with a probability of availability of data, p. When $p = 1$, all information in f_t^a is available and fully reliable. The difference or base subset of f_t, denoted by f_t^b, is mutually exclusive to f_t^a and is calculated as the difference between f_t and f_t^a. Unlike f_t^a, the data in f_t^b is always available and reliable. The online model works in a predict-then-update manner where the prediction for f_{t+1} cannot be made before the online model is updated with prediction for f_t. In other words, the time-series data offers a feature to be learned. Each such feature consists of reliable (base) and unreliable (auxiliary) features. The model is tasked with online learning with this time-series data.

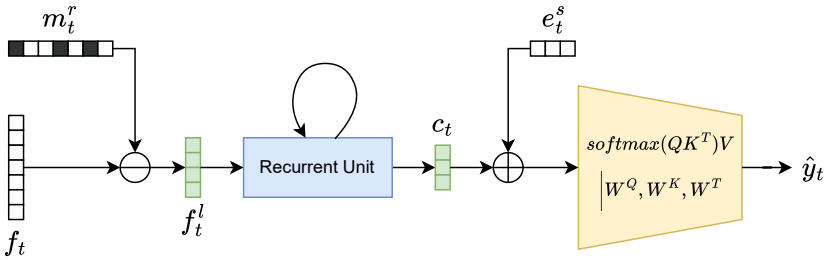
Hedging as weighted residuals. Hedging algorithm proposed by [3] has severed as the primary driving force of many haphazard data processing methods. The main idea is to create an ensemble of classifiers where the outcome of each classifier is weighted by a scalar value, as shown below:

$$\hat{y}_t^E = \sum \alpha_i \, \hat{y}_{t,i}^C, \tag{1}$$

$$\hat{y}_{t,i}^C = \sigma(\Theta_i \, \sigma(W_i \, h_{i-1})), \tag{2}$$



(a) HapNet



(b) HapNetPU

Figure 1: The two proposed models are shown in (a) HapNet (for positionally correlated), (b) HapNetPU (for positionally uncorrelated case). Both explain how an input feature f_t is processed. $\odot$ implies Hadamard product, $\ominus$ implies inverse Hadamard, the operation of removing the values from specific positions and compressing the dimensions of the input feature at time t in order to make them positionally uncorrelated. f_t is input features at time t, f_t^b is base features at time t, f_t^a is auxiliary features at time t, f_t^m is masked features at time t, m_t^r is randomized mask at time t, e_t^s is scalar position encoding at time t, f_t^l is remaining features after feature loss at time t, c_t is context at time t, $\hat{y}_t$ is predicted label at time t.

where, $\hat{y}_t^E$ is the ensemble's label prediction, $\hat{y}_{t,i}^C$ is the layer's label prediction of the i^{th} classifier, Θ_i is the classification weight matrix of the i^{th} classifier, α_i is classifier weight, σ is the activation function, W_i is the layer's weight matrix, and h_{i-1} is the activation from the previous layer. In hedging, a label prediction loss is imposed on every classifier's prediction $\hat{y}_{t,i}^C$ and also on the ensemble's prediction $\hat{y}_t^C$. The output of a fully residual network (like densenet), without an ensemble loss unlike Eq. (2), can also be written as:

$$\begin{aligned} \hat{y}_t^R &= \sum \alpha_i' \, \sigma(\Theta_i' \, \sigma(W_i' \, h_{i-1}')) \\ &= \sum \alpha_i' \, \sigma(W_i'' \, h_{i-1}'). \end{aligned} \tag{3}$$

where, W_i'' approximates the compound layer operation. In Eq. (3), if $\alpha_i' = 1$, the model behaves like a vanilla residual network, and if $\alpha_i' \neq \{1, 0\}$, then the model becomes a weighted residual network. This architectural similarity between hedging and weighted residual approach indicates that hedging uses coarse-level attention. Inspired by this similarity, we propose to replace this approach with a simple yet effective, fine intra attention approach.

HapNet. The proposed HapNet model (Fig. 1a) is based on self-attention. Each feature f_t is randomized/bootstrapped on its auxiliary features to produce multiple masked versions of f_t^a which leads to multiple copies of f_t. These copies are directly passed to the positional embedding layer e_t^s, that adds the positional information. Note that the feature values do not pass through any embedding layer themselves and are directly used as their corresponding embeddings. This was experimentally discovered by us that it is better to pass the values directly. This discovery is in direct contrast to [7]. The resulting vector is passed through an encoder based on self-attention. In our specific model we have used transformers to model that. At the inference time, the randomization module on f_t^a is turned off, so that the feature f_t directly passes through the encoder. We have used cross-entropy loss for training of HapNet.

HapNetPU. We discovered that although haphazard inputs pose significant challenge to typical deep learning models, however, a more realistic situation could be when the inputs are completely unavailable and in those scenarios it is hard to match their unavailability with their positions. Assumption is that the order of features is maintained. Under such

Table 1: The CE loss on Italy power demand dataset are shown for AuxNet, AuxDrop (ODL), HapNet, along with errors for HapNet. Best results are marked as **bold** and the second best are underlined at each probability value. Each HapNet experiment was repeated 20 times. Probability values corresponds to the availability of auxiliary features.

Probability	AuxNet	AuxDrop	HapNet	HapNet (Error)
0.5	0.6975	0.6031±0.0081	**0.4512±0.0187**	234.92±17.36
0.6	0.6831	0.5839±0.0111	**0.4092±0.0146**	204.30±11.35
0.7	0.6788	0.5497±0.0082	**0.3726±0.0231**	179.14±14.29
0.8	0.6130	0.5321±0.0071	**0.3203±0.0127**	146.10±7.66
0.9	0.5456	0.5149±0.0119	**0.2780±0.0151**	119.50±7.26
0.95	0.5168	0.5013±0.0108	**0.2462±0.0152**	100.78±9.63
0.99	0.5165	0.4788±0.0101	**0.2254±0.0158**	89.26±10.12

Table 2: Comparison of error among OLVF, AuxDrop (ODL), AuxDrop (OGD), HapNet is shown for different datasets. Best results are marked as **bold** and the second best are underlined at each probability value. The error is reported as the mean ± standard deviation of the 20 experiments performed with random seeds. Probability values corresponds to the availability of auxiliary features.

Probability	Dataset	OLVF	AuxDrop (ODL)	AuxDrop (OGD)	HapNet
0.73	*german*	333.4±9.7	**300.4±4.4**	312.8±19.3	307.6±1.6
0.72	*svmguide3*	346.4±11.6	**297.2±2.0**	297.5±1.5	299.8±0.2
0.68	*magic04*	6152.4±54.7	5536.7±59.3	**5382.8±98.9**	6391.7±124.1
0.75	*a8a*	8993.8±40.3	6710.7±117.8	7313.5±277.7	**6514.4±87.6**

Table 3: Comparison of average number of errors among OLSF, OLVF, AuxDrop (ODL), AuxDrop (OGD), HapNet is shown on trapezoidal data streams. Best results are marked as **bold** and the second best are underlined at each probability value. The error is reported as the mean ± standard deviation of the 20 experiments performed with random seeds.

Dataset	OLSF	OLVF	AuxDrop (ODL)	AuxDrop (OGD)	HapNet
german	385.5±10.2	329.2±9.8	312.2±8.0	320.9±39.4	**301.2±1.9**
svmguide3	361.7±29.7	351.6±25.9	296.9±1.0	297.0±0.9	**295.7±8.3**
magic04	6147.4±65.3	5784.0±52.7	6361.25±319.6	**5635.8±100.9**	6671.5±98.1
a8a	9420.4±549.9	8649.8±526.7	7850.9±15.9	7848.8±10.3	**7831.5±21.0**

scenarios, no existing models could work. To tackle this problem, we present HapNetPU (Fig. 1b. HapNetPU builds upon HapNet by employing a feed-back operator (can be realized by any recurrent network like LSTM or GRU). The resulting feature f_t^l loses its original dimension and is truly haphazard in behavior. HapNetPU, due to its recurrence operation, is able to process it though.

3 Experiments

This section presents the experimental protocol and details, the dataset used, results, and analysis. We use state-of-the-art models for processing haphazard inputs [1, 2] and other deep learning methods [3, 14] for comparison to the proposed method. Average error i.e. number of incorrect predictions out of total samples tested is used as the evaluation metric. Along with that, the cross entropy (CE) loss is used to train and test the models. Rarely, macro and micro F1 scores and accuracy have been used as alternative metrics. The encoder consists of a transformer with 6 blocks, a dropout of .15 is used with layer normalization and two liner layers on the outputs of self-attention. If not reported otherwise, the learning rate was 0.0001 with Adam optimizer. All experiments were repeated 20 times and their mean and standard deviations have been reported.

Datasets Five datasets have been used in our experiments. These are (a) Italy power demand dataset [15] (b) *german*, *svmguide3*, *magic04*, *a8a* [16]. All these datasets contain time-series data from different domains and applications. More details can be found in Agarwal *et al.* [1].

Table 4: Comparison in terms of the average error and CE loss is shown for the proposed HapNetPU on three datasets. Each experiment was repeated 20 times.

Dataset	Probability	Error	Loss
Italy power	0.8	512.7±8.8	0.7086±0.0035
german	0.73	326.4±0.4	0.6470±0.0020
svmguide3	0.72	301.0±1.4	0.5695±0.0022

Table 5: In this ablation the dropout scores are varied on the *german* dataset to see its effect on the performance of the HapNet method, both in terms of the average error and CE loss.

Data stream	Dropout	Probability	Error	CE Loss
Trapezoid	0.5	0.73	322.8±3.4	0.6433±0.0030
Trapezoid	0.3	0.73	305.4±8.7	0.6241±0.0065
Haphazard	0.5	0.73	325.6±0.3	0.6389±0.0023
Haphazard	0.3	0.73	317.2±1.3	0.6305±0.0011

Table 6: In this ablation the probability of presence of auxiliary inputs is varied on the *german* dataset to see its effect on the performance of the HapNet method, both in terms of the average error and CE loss.

Probability	0.6	0.7	0.8	0.9
Loss	0.6268	0.6209	0.6193	0.6091
Error	310.1	308.4	303.9	294.4
Macro F1	0.4772	0.4841	0.4959	0.5101
Micro F1	0.6898	0.6915	0.6960	0.7055
Accuracy	0.6898	0.6915	0.6960	0.7055

Table 7: In this ablation the learning rate is varied on the *german* and *svmguide3* datasets with six encoders and batchsize 64 to see its effect on the performance of the HapNet method, both in terms of the average error and CE loss.

Learning rate	Dataset	Error	Loss
0.001	*german*	325.2	0.6473
0.001	*german*	319.9	0.6403
0.00001	*german*	316.4	0.6219
0.001	svmguide3	303.6	0.5762
0.001	*svmguide3*	300.8	0.5657
0.00001	*svmguide3*	299.3	0.5552

Table 8: In this ablation the number of encoders are varied on the *german* and *svmguide3* datasets with learning rate 0.0001 and batchsize 64 to see its effect on the performance of the HapNet method, both in terms of the average error and CE loss.

Number of encoders	Dataset	Error	Loss
12	*german*	306.5	0.6205
24	*german*	306.6	0.6197
12	*svmguide3*	301.2	0.5615
24	*svmguide3*	301.7	0.5616

Results and ablation. We performed several experiments with different types of datasets Table 1, 2, 3 to evaluate the performance of the proposed method. The proposed approach has shown consistent performance across different datasets, metrics, and evaluation settings. Note that Table 3 uses trapezoid inputs [1]. It is particularly worth noting in Table 1, HapNet outperforms AuxNet, approximately, by a factor of two. Table 4 shows the results for the positionally non-correlated setting. Note that the performance of HapNetPU is not lagging significantly on *german* and *svmguide3* datasets. This shows strong capability of HapNetPU to model even under positionally non-correlated haphazard inputs.

Table 9: In this ablation the batchsize is varied on the *german* and *svmguide3* datasets with learning rate 0.0001 and 6 number of encoders to see its effect on the performance of the HapNet method, both in terms of the average error and CE loss.

Batch size	Dataset	Error	Loss
16	*german*	306.9	0.6193
32	*german*	309.8	0.6211
128	*german*	306.1	0.6184
16	*svmguide3*	300.1	0.5605
32	*svmguide3*	301.6	0.5616
128	*svmguide3*	301.1	0.5619

Further, we have performed several ablations on the role of dropout, suitability of number of encoders, batchsizes, learning rates and the variation of model performance with changing probability. These can be referenced in Table 9, 5, 8, 7, 6.

3.1 Conclusion

This work propose a simple yet effective algorithm for haphazard input based problems. It shows that the more widely used hedging based algorithm can be well estimated by a more generic version – self-attention, leading to better generalization, scalability, ease of use and adaptability to different modalities. The proposed methods achieved competitive performance to other state-of-the-art methods. We further introduced a more challenging case of haphazard inputs where the features values are positionally non-correlated. We showed that a feed-back based modification of the proposed HapNet may address this challenging case as well. Extensive ablations on the five datasets revealed the effectiveness of the proposed HapNet approach. However there are some limitations to the proposed work. **Limitation.** Although the HapNet approach shows competitive performance, however, in some cases the other methods outperform it. It shows that a more optimized variant of HapNet can be developed. In place of LSTMs in HapNetPU, some other network like GRU or transformers may be used to further improve its performance. **Future scope.** Our approach can be extended to images and videos as the transformer architecture is omnipresent and the data can be easily divided. Different techniques may be used to replace the feed-back module in HapNetPU to further improve its performance.

References

[1] Rohit Agarwal, Deepak Gupta, Alexander Horsch, and Dilip K Prasad. Aux-drop: Handling haphazard inputs in online learning using auxiliary dropouts. *arXiv preprint arXiv:2303.05155*, 2023. 1, 2, 4, 5

[2] Rohit Agarwal, Krishna Agarwal, Alexander Horsch, and Dilip K Prasad. Auxiliary network: Scalable and agile online learning for dynamic system with inconsistently available inputs. In *International Conference on Neural Information Processing*, pages 549–561. Springer, 2022. 1, 2, 4

[3] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: learning deep neural networks on the fly. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2660–2666, 2018. 1, 2, 4

[4] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 2

[5] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2020. 2

[6] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 2

[7] Jannis Born and Matteo Manica. Regression transformer enables concurrent sequence regression and generation for molecular language modelling. *Nature Machine Intelligence*, 5(4):432–444, 2023. 2, 3

[8] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for on-demand classification of evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):577–589, 2006. 2

[9] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000. 2

[10] Ivor W Tsang, Andras Kocsor, and James T Kwok. Simpler core vector machines with enclosing balls. In *Proceedings of the 24th international conference on Machine learning*, pages 911–918, 2007. 2

[11] Tor Das et al. A novel incremental rough set-based pseudo outer-product with ensemble learning (ierspop) neuro-fuzzy system for forecasting volatility. 2010. 2

[12] Daniel Leite, Pyramo Costa, and Fernando Gomide. Evolving granular neural networks from fuzzy data streams. *Neural Networks*, 38:1–16, 2013. 2

[13] Robi Polikar. Ensemble learning. *Ensemble machine learning: Methods and applications*, pages 1–34, 2012. 2

[14] Ege Beyazit, Jeevithan Alagurajah, and Xindong Wu. Online learning from data streams with varying feature spaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3232–3239, 2019. 2, 4

[15] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. 4

[16] Arthur Asuncion, David Newman, et al. Uci machine learning repository, 2007. 4