EDITS: An Easy-to-difficult Training Strategy for Cloud Failure Prediction

Tianci Li, Pu Zhao, Yudong Liu, Minghua Ma
Microsoft Research
Beijing, China

Lingling Zheng, Murali Chintalapati
Microsoft Azure
United States

Bo Liu, Paul Wang
Microsoft 365
Suzhou, China

Hongyu Zhang
Chongqing University
Chongqing, China

Yingnong Dang, Saravan Rajmohan
Microsoft Azure
United States

Qingwei Lin, Dongmei Zhang
Microsoft Research
Beijing, China

ABSTRACT

Cloud failures have been a major threat to the reliability of cloud services. Many failure prediction approaches have been proposed to predict cloud failures before they actually occur, so that proactive actions can be taken to ensure service reliability. In industrial practice, existing failure prediction approaches mainly focus on utilizing state-of-the-art time series models to enhance the performance of failure prediction but neglect the training strategy. However, as curriculum learning points out, models perform better when they are trained with data in an order of easy-to-difficult. In this paper, we propose *EDITS*, a novel training strategy for cloud failure prediction, which greatly improves the performance of the existing cloud failure prediction models. Our experimental results on industrial and public datasets show that *EDITS* can obviously enhance the performance of cloud failure prediction model. In addition, *EDITS* also outperforms other curriculum learning methods. More encouragingly, our proposed *EDITS* has been successfully applied to Microsoft 365 and Azure online service systems, and has obviously reduced financial losses caused by cloud failures.

1 INTRODUCTION

Cloud providers, such as Microsoft, Google, and Azure, have committed to improving the reliability of cloud services [1, 6, 7, 16, 17]. Cloud failures, such as node [4, 23] and disk failures [4], however, are still inevitable due to hardware [19] aging, software bugs [15, 36], etc. These failures adversely affect user experience and lead to economic losses [13, 20].

Over the years, to eliminate the loss of cloud failures, various approaches [4, 9, 12, 14, 15, 25, 26, 33–35, 37] have been proposed to make prediction and take proactive actions before cloud failures happen. These methods regard cloud failure prediction as a binary classification problem and utilize time series models such as TCNN [26], RNN [33], LSTM [35] and Transformer [22], which effectively predict cloud failures and reduce financial losses.

To achieve good prediction performance, existing cloud failure prediction methods mainly focus on the model design, while ignore the optimization of training strategies in industrial practice. A good training strategy like curriculum learning, however, is significant for the model performance in CV and NLP domains. As illustrated in curriculum learning [3], human and animals learn much better when the examples are not randomly presented, but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. Machine learning algorithms can also benefit from such learning process. In this paper, to further enhance the performance of cloud failure prediction, we leverage the human learning process mentioned above and propose *EDITS*, an easy-to-difficult training strategy. Different from previous curriculum methods, *EDITS* takes consideration of cloud failure prediction scenario and handle difficulty ranking problem for positive and negative samples separately. For positive samples, we adopt a data augmentation method [18] and apply prior knowledge to rank difficulty for augmented samples. In addition, we design a novel difficulty estimator for negative samples according to the loss values during training process.

To evaluate the effectiveness and robustness of *EDITS*, we conduct experiments on two industrial datasets and two public datasets to compare the performance of various models with and without *EDITS*. We also compare *EDITS* with three other training strategies. Experiments on four datasets show that *EDITS* significantly improves the performance of cloud failure prediction models and outperforms the three other training strategies. More encouragingly, *EDITS* has been successfully applied to Microsoft cloud platforms (including Microsoft Azure and Microsoft 365), which improves the reliability of Microsoft cloud platforms.

The main contributions of this paper are as follows:

- To the best of our knowledge, we firstly leverage training strategy method into cloud failure prediction.
- We propose *EDITS*, which utilizes prior knowledge to rank the difficulty for positive samples and designs a novel difficulty evaluator for negative samples.
- Extensive experiments on industrial datasets and public datasets demonstrate that our proposed *EDITS* considerably enhances the performance of various cloud failure prediction models.

2 RELATED WORK

In recent years, many approaches for cloud failure prediction have been proposed in recent years, such as recurrent neural network (RNN) [33], long short-term memory (LSTM) [35] and temporal convolutional neural network (TCNN) [26] and Transformer [18]. These methods mainly focus on the model design, while ignore the optimization of training strategy, which is also significant to enhance the prediction performance.

Curriculum learning (CL) [3] has been attracting lots of attention in various fields such as computer vision (CV) [8, 10] and natural language processing (NLP) [21, 31], which is a training strategy where samples are trained in a *easy-to-difficult* order. The core step of CL is to evaluate the difficulty for samples. Curriculum learning concludes various approaches, such as SPL [11, 27, 29, 30], SuperLoss [5], and Cross Review method [32].

In this paper, we firstly leverage CL into could failure prediction, and design a novel difficulty evaluator, which enhances the performance of cloud failure prediction a lot.

3 OUR PROPOSED APPROACH

In this section, we illustrate our proposed *EDITS* in detail. Firstly, we give the problem definition of cloud failure prediction in Sec. 3.1. We introduce the difficulty ranking strategy for positive and negative samples in Sec. 3.2. Finally, we will introduce the training process based on ranked samples in Sec. 3.3. The whole architecture of *EDITS* is shown as Figure 1.

3.1 Problem definition

Generally, a large cloud platform consists of millions of units such as disks or nodes, which are denoted as components. To ensure the reliability of cloud, various monitors record the health state of components at interval, forming as multidimensional time series data[24, 28]. Assume d_t denotes time series data in the range of h consecutive time stamps from the time tamp $t_i - h + 1$ to the time stamp t_i. In simple terms, our dataset consists of N samples, which can be represented as D = $(X_1, y_1), ..., (X_N, y_N)$, where X_i represents the state data of i-th component in the format of d, and y_i denotes the label. If $y_i = 1$, it means that the component will fail in the near future, otherwise $y_i = 0$.

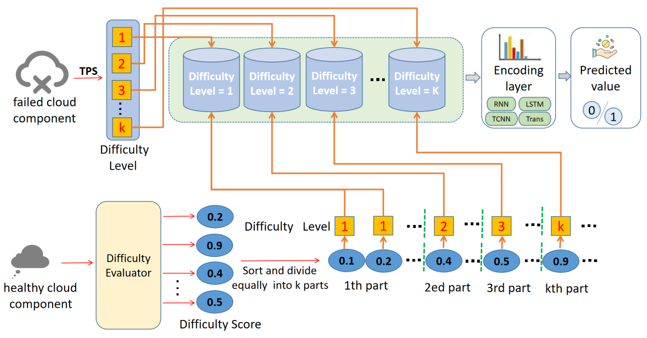

Figure 1: Architecture of *EDITS*.

3.2 Difficulty ranking

In this section, we illustrate the difficulty ranking for positive (failed) and negative (healthy) samples, which is the core step of our training strategy. We apply prior knowledge to discriminate the difficulty of positive samples. In addition, we design a novel difficulty estimator to evaluate the difficulty of negative samples.

3.2.1 *Positive samples.*

To handle the difficulty ranking for positive samples, we utilize Temporal Progressive Sampling (*TPS*) [18], a data augmentation approach. The core idea of *TPS* is to regard time series data within a small number of timestamps ahead failure as additional positive samples. Specifically, for positive sample d_t, *TPS* augments $d_{t-1}, d_{t-2}, ..., d_{t-k+1}$, where k denotes max leading time. Obviously, the difficulty of classifying these augmented positive samples is different. The nearer to failure time, the less difficult it is for the model to predict failure for the positive sample.

Based on the above prior knowledge, we firstly utilize *TPS* to perform data augmentation on positive samples, and then divide positive samples into k buckets according to their distance to the failure time (named leading time). Specifically, positive samples with leading time value l are allocated to the l-th bin. In this way, the difficulty arises as the bin number.

3.2.2 *Negative samples.*

For negative samples, we design a difficulty evaluator according to the loss value. We train the whole data for a certain number of epoches and record the loss value of each epoch for negative samples during the training process. Based on the loss value, we calculate the difficulty score of each negative sample as following:

$$S_i = \alpha \cdot S_{i-1} + (1 - \alpha) \cdot L_i \tag{1}$$

As shown in Equation 1, for each negative sample, L_i denotes the cross entropy loss during the i-th epoch training, S_i denotes the cumulative difficulty score of the first i epoches, which is weighted by the cumulative difficulty of the first $i - 1$ epoch and the one of the i-th epoch. α denotes the weight parameter. After m epochs of training, we can get the final weighted difficulty score S_m for each negative sample. The larger S_m is, the harder the sample is to classify. Similar to positive samples, we also divide negative samples into k buckets according to difficulty scores in the incremental order.

3.3 Bucket training

As mentioned above, we process difficulty ranking for positive and negative samples respectively and divide them into k buckets. Based on these buckets, we train samples with an easy-to-difficult strategy. Specifically, in the i-th training iteration, we train the model based on positive and negative samples from corresponding i-th buckets, respectively. In this way, the model is firstly trained on easier samples, and then learns harder samples gradually.

It is worth noting that *TPS* may bring some noises when performing data augmentation. To handle this issue, we perform a fine-tuning operation when training bucket samples. Specifically, when we train the samples of the i-th bucket, we will load the model with the best performance in the first i − 1-th buckets, and train the samples of the i-th bucket on the basis of the loaded model. In other words, we keep the best-performing models of the first i-th buckets, and remove the training results of the middle buckets with poor performance, which can effectively eliminate noise.

4 EXPERIMENTS

In this section, we conduct experiments to demonstrate the effectiveness and robustness of our method. We firstly give the experimental setup in Sec. 4.1, and then introduce the dataset in Sec. 4.2. We anlayse the experimental results in Sec. 4.3.

Table 1: Comparison results of various training strategies for various time series models on four cloud failure prediction datasets. P, R and F1 refer to precision, recall and F1 score respectively.

Model	Approach	M365			Azure			Ali Cloud			BackBlaze		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
TCNN	*Baseline*	90.21	43.76	58.94	66.89	54.28	59.93	30.18	72.44	42.61	75.31	59.17	66.27
	SPL	83.12	48.57	61.32	74.91	50.47	60.31	30.82	72.15	43.19	74.91	61.26	67.40
	SuperLoss	91.79	45.86	61.17	71.52	54.53	61.88	30.21	74.75	43.03	77.65	56.63	65.49
	Cross Review	89.62	46.13	60.91	75.02	53.68	62.58	31.09	75.77	44.09	84.86	59.09	69.67
	TPS strategy	91.87	45.66	61.00	73.31	54.10	62.26	30.98	74.60	43.78	77.06	62.01	68.72
	Coarse strategy	89.29	46.96	61.55	72.44	54.14	61.97	31.56	76.12	44.62	74.95	62.56	68.20
	EDITS	90.88	49.31	**63.93**	71.95	57.88	**64.15**	32.56	78.76	**46.08**	80.33	65.12	**71.95**
RNN	*Baseline*	90.59	47.66	62.46	69.07	57.02	62.47	30.74	75.43	43.68	75.08	66.32	70.43
	SPL	93.87	47.01	62.59	71.02	55.30	62.18	30.54	81.23	44.39	77.26	67.53	72.07
	SuperLoss	91.85	50.04	64.78	72.78	58.86	63.84	31.09	74.78	43.92	78.78	64.86	71.14
	Cross Review	90.33	47.78	62.50	71.60	59.43	64.95	31.39	77.07	44.61	89.80	58.25	70.66
	TPS strategy	90.44	49.96	64.37	70.88	58.65	64.19	30.98	84.86	45.39	76.65	66.69	71.78
	Coarse strategy	91.48	49.38	64.14	72.16	59.18	65.03	32.51	84.53	46.96	76.18	67.29	71.46
	EDITS	92.28	50.47	**66.25**	74.11	62.43	**67.77**	33.62	87.20	**48.53**	83.05	65.50	**73.24**
LSTM	*Baseline*	90.29	47.79	62.50	72.58	57.83	64.37	30.54	83.61	44.74	75.26	66.19	70.43
	SPL	92.85	49.60	64.66	73.62	59.33	65.71	30.32	80.04	43.98	76.35	67.72	71.78
	SuperLoss	92.22	47.57	62.76	72.53	61.02	66.28	31.04	83.87	45.31	76.02	66.11	70.72
	Cross Review	91.04	50.34	64.83	71.86	61.35	66.19	31.52	84.61	45.93	88.24	58.68	70.49
	TPS strategy	93.81	49.19	64.54	71.65	61.88	66.41	31.88	84.73	46.33	88.24	58.68	70.21
	Coarse strategy	89.57	50.41	64.52	74.09	60.85	66.82	33.70	84.96	48.26	79.65	64.98	71.57
	EDITS	97.10	50.23	**66.21**	76.41	64.45	**69.92**	49.53	85.78	**49.53**	84.51	65.51	**73.81**
Transformer	*Baseline*	90.38	48.60	63.21	73.51	60.00	66.07	32.06	80.89	45.92	77.78	67.12	72.06
	SPL	89.82	48.19	62.73	72.55	61.77	66.73	31.73	75.14	44.62	77.40	67.27	71.98
	SuperLoss	89.04	49.55	63.67	71.92	64.04	67.75	31.33	84.83	45.76	76.53	68.05	72.04
	Cross Review	90.62	49.58	64.09	74.03	61.16	66.98	31.17	88.70	46.13	77.01	67.92	72.18
	TPS strategy	93.30	49.94	65.06	71.69	64.33	67.81	33.17	86.35	47.93	78.54	68.49	73.17
	Coarse strategy	92.87	49.15	64.28	72.80	62.82	67.44	32.82	88.07	47.82	78.54	67.92	73.18
	EDITS	92.79	51.83	**66.51**	76.91	65.06	**70.49**	35.73	82.42	**49.85**	69.36	81.54	**75.96**

4.1 Experimental Setup

We adopt *EDITS* on four state-of-the-art methods which are widely used in the cloud failure prediction: RNN, LSTM and TCNN and Transformer. In addition, we compare *EDITS* with several state-of-the-art training strategies such as SPL, SuperLoss and Cross Review. We also conduct experiments on two training strategies which are partly similar to *EDITS*. The brief illustration for each method is as following:

- *Baseline* does not utilize any training strategy.
- *SPL* [11] proposes a iterative self-paced learning algorithm where each iteration simultaneously selects easy samples and learns a new parameter vector.
- *SuperLoss* [5] uses a general loss function to automatically reduce the contribution of samples with larger losses, i.e. hard samples, to simulate the curriculum learning process.
- *Cross Review* [32] randomly divides the data into k parts and trains k models as reviewers. The difficulty of the i^{th} data is determined by the evaluation loss of other $k-1$ reviewers training this data.
- *TPS strategy* only utilizes the prior knowledge for the difficulty ranking of positive samples, and perform the average random bucketing operation. This strategy is similar with our proposed *EDITS* in the operation for positive samples.
- *Coarse strategy* utilizes a difficulty estimator to score all samples for difficulty, and then divide all samples into buckets according to their difficulty scores. This strategy is similar with our proposed *EDITS* in the operation for negative samples.
- *EDITS* (our method) utilizes the prior knowledge for the difficulty ranking of positive samples and scores negative samples with a difficulty evaluator. Then it allocates the positive and negative samples of the same difficulty level into the same bucket.

All experiments are conducted on a workstation equipped with NVIDIA Tesla P100 GPU and CUDA 10.2. The code is implemented based on PyTorch 1.8. During the training process, we utilize Adam optimizer and set the initial learning rate as 1e-3. In addition, the training epoch is set to 100 and the batch size is 128.

4.2 Datasets

We collect two industrial datasets from Microsoft 365 (M365) and Microsoft Azure, both serving a large number of customer workloads and both containing over two months of cloud component time series data. Specifically, M365 datasets contain state data for large amount of disks and Azure dataset also contains the state data of hundreds of thousands of cloud nodes. In addition, in these two industrial datasets, the status data of each cloud component is recorded every hour, and we take the last 72 hours of time series data as a sample each time.

We also conduct experiments on two public datasets, Ali dataset[1] and Backblaze dataset[2] [2]. Backblaze takes a snapshot of each running hard drive, which includes basic drive information in SMART statistics format. It contains 18 months of data (January 2021 to June 2022). Ali dataset is a real industrial data collected by Alibaba Cloud data centers which is widely used to evaluate the performance of cloud failure prediction methods.

The number of positive samples (failed cloud components, denoted as '#Pos'), negative samples (healthy cloud components, denoted as '#Neg') and feature dimensions for each dataset is shown in Table 2.

Table 2: Summary of M365, Azure, Ali and BackBlaze Dataset.

Dataset	Dimension	Training Set		Testing Set	
		#Pos	#Neg	#Pos	#Neg
M365	32	2799	478096	3552	462939
Azure	23	1706	864623	1673	864656
Ali Cloud	11	987	128569	459	53523
BackBlaze	39	7325	108659	9015	106970

4.3 Experimental Results

In this section, we conduct experiments on four cloud prediction datasets: M365 dataset, Azure dataset, ali dataset and backblaze dataset, to compare the results with and without *EDITS*. And we also compare *EDITS* with other training strategies. The result is shown in Table 1.

4.3.1 Result on Industrial Dataset. We conduct experiments on M365 dataset and Azure dataset. According to the result, models with *EDITS* perform significantly better than those without. Specifically, the f1 score of TCNN, RNN, LSTM, and Transformer enhanced by 4.99%, 3.79%, 3.71%, and 3.30% respectively after using *EDITS*. In addition, compared with other training strategies., *EDITS* is more suitable for cloud failure prediction scenarios.

Taking the Transformer model as an example, the F1 score with *EDITS* is 3.78%, 2.84%, and 2.42% higher than those with SPL, Cross-Review, and SuperLoss, respectively. In addition, *EDITS* also outperforms the *TPS strategy* and *Coarse strategy*. Based on *EDITS*, the F1 score of Transformer is enhanced by 1.45% and 2.23% compared with *TPS strategy* and *Coarse strategy*, respectively.

[1]https://tianchi.aliyun.com/competition/entrance/231775/information?lang=en-us
[2]https://tianchi.aliyun.com/competition/entrance/231775/rankingList/1

We also obtained similar results on the Azure dataset. The F1 score of TCNN, RNN, LSTM, and Transformer are enhanced by 4.22%, 5.30%, 5.55%, and 4.42% respectively based on *EDITS*. In addition, for the best-performing Transformer model, *EDITS* still outperforms other curriculum learning methods. based on *EDITS*, the F1 score of Transformer is enhanced by 3.76%, 2.74%, and 3.51% compared with SPL, Cross review, and SuperLoss, respectively. *EDITS* also improves the F1 score by 2.68% and 3.05% on the Transformer model compared with the *TPS strategy* and *Coarse strategy*.

4.3.2 Result on Public Datasets. We also conduct experiments on two public datasets. For the Ali dataset, it is obvious that the models with *EDITS* perform significantly better than those without. Specifically, the F1 score of TCNN, RNN, LSTM, and Transformer are enhanced by 3.47%, 4.85%, 4.79%, and 3.93% respectively based on *EDITS*. In addition, our method also achieves better performance compared to other curriculum learning methods. Taking the best-performing transformer model as an example, the F1 score based on our proposed *EDITS* is 5.23%, 4.09%, and 3.72% higher than that after using SPL, Cross Review, and SuperLoss, respectively. In addition, *EDITS* also outperforms with F1 scores 1.92%, 2.03% higher than *TPS strategy* and *Coarse strategy*, respectively.

Similarly, the models based on *EDITS* performs significantly better than those without. Specifically, the F1 score of TCNN, RNN, LSTM, and Transformer are enhanced by 5.68%, 2.81%, 3.38%, and 3.90% respectively after using *EDITS*. In addition, *EDITS* improves F1 score by 3.98%, 3.92%, 3.78%, 2.79%, 2.78% over the SPL, Cross Review, SuperLoss, *TPS strategy* and *Coarse strategy* for Transformer model.

5 APPLICATION IN PRACTICE

Our training strategy greatly improves the performance of the cloud fault prediction model without changing the data composition, thereby improving the reliability of cloud platform services. Microsoft Azure and Microsoft 365 large-scale distributed systems benefit a lot from *EDITS*. The improvement in reliability will also further promote the globalization of Microsoft's cloud service business. In addition, we also collect virtual machine outage data from Microsoft cloud platforms, and the results show that our proposed training strategy significantly reduces the number of virtual machine outages on these cloud platforms, which demonstrates the effectiveness of *EDITS*.

6 CONCLUSION

Cloud failure is one of the main reasons for the unreliability of cloud platforms, which plays a vital role in industrial practice. Inspired by curriculum learning, in this paper, we propose *EDITS*, a new training strategy to enhance the performance of cloud failure prediction. Compared with the existing methods that only focus on the model design, *EDITS* considers difficulty of training samples and follow the order of easy-to-difficult. Extensive experiments show that *EDITS* greatly improves the performance of cloud failure prediction models. Furthermore, our proposed *EDITS* outperforms other curriculum learning strategies, which demonstrates the effectiveness and robustness of our method. We have applied *EDITS* into Microsoft cloud platforms and bring considerable financial benefits.

REFERENCES

[1] Danilo Ardagna, Barbara Panicucci, and Mauro Passacantando. 2011. A game theoretic formulation of the service provisioning problem in cloud systems. In *Proceedings of the 20th international conference on World wide web*. 177–186.

[2] Backblaze. 2019. The Backblzae Hard Drive Data and Stats. https://www.backblaze.com/b2/hard-drive-test-data.html.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

[4] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. 2016. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 39–48.

[5] Thibault Castells, Philippe Weinzaepfel, and Jerome Revaud. 2020. Superloss: A generic loss for robust curriculum learning. *Advances in Neural Information Processing Systems* 33 (2020), 4308–4319.

[6] Jiazhen Gu, Chuan Luo, Si Qin, Bo Qiao, Qingwei Lin, Hongyu Zhang, Ze Li, Yingnong Dang, Shaowei Cai, Wei Wu, et al. 2020. Efficient incident identification from multi-dimensional issue reports via meta-heuristic search. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 292–303.

[7] Jiazhen Gu, Jiaqi Wen, Zijian Wang, Pu Zhao, Chuan Luo, Yu Kang, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, Bo Qiao, Liqun Li, Qingwei Lin, and Dongmei Zhang. 2020. Efficient customer incident triage via linking with system incidents. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*. 1296–1307.

[8] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. 2018. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European conference on computer vision (ECCV)*. 135–150.

[9] Xiaohong Huang. 2017. *Hard drive failure prediction for large scale storage system.* Ph. D. Dissertation. UCLA.

[10] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. 2014. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia*. 547–556.

[11] M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *Advances in neural information processing systems* 23 (2010).

[12] Jing Li, Xinpu Ji, Yuhan Jia, Bingpeng Zhu, Gang Wang, Zhongwei Li, and Xiaoguang Liu. 2014. Hard drive failure prediction using classification and regression trees. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 383–394.

[13] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, et al. 2020. Gandalf: An Intelligent,{End-To-End} Analytics Service for Safe Deployment in {Large-Scale} Cloud Infrastructure. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 389–402.

[14] Yudong Liu, Hailan Yang, Pu Zhao, Minghua Ma, Chengwu Wen, Hongyu Zhang, Chuan Luo, Qingwei Lin, Chang Yi, Jiaojian Wang, et al. 2022. Multi-task Hierarchical Classification for Disk Failure Prediction in Online Service Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3438–3446.

[15] Sidi Lu, Bing Luo, Tirthak Patel, Yongtao Yao, Devesh Tiwari, and Weisong Shi. 2020. Making Disk Failure Predictions {SMARTer}!. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*. 151–167.

[16] Chuan Luo, Bo Qiao, Xin Chen, Pu Zhao, Randolph Yao, Hongyu Zhang, Wei Wu, Andrew Zhou, and Qingwei Lin. 2020. Intelligent Virtual Machine Provisioning in Cloud Computing.. In *IJCAI*. 1495–1502.

[17] Chuan Luo, Bo Qiao, Wenqian Xing, Xin Chen, Pu Zhao, Chao Du, Randolph Yao, Hongyu Zhang, Wei Wu, Shaowei Cai, et al. 2021. Correlation-aware heuristic search for intelligent virtual machine provisioning in cloud systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 12363–12372.

[18] Chuan Luo, Pu Zhao, Bo Qiao, Youjiang Wu, Hongyu Zhang, Wei Wu, Weihai Lu, Yingnong Dang, Saravanakumar Rajmohan, title=NTAM: neighborhood-temporal attention model for disk failure prediction in cloud platforms, Qingwei Lin, et al. 2021. In *Proceedings of WWW 2021*. 1181–1191.

[19] Ao Ma, Rachel Traylor, Fred Douglis, Mark Chamness, Guanlin Lu, Darren Sawyer, Surendar Chandra, and Windsor Hsu. 2015. RAIDShield: characterizing, monitoring, and proactively protecting against disk failures. *ACM Transactions on Storage (TOS)* 11, 4 (2015), 1–28.

[20] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. 2020. Automap: Diagnose your microservice-based web applications automatically. In *Proceedings of The Web Conference 2020*. 246–258.

[21] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M Mitchell. 2019. Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848* (2019).

[22] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive Transformers for Long-Range Sequence Modelling. In *Proceedings of ICLR*.

[23] Nashid Shahriar, Reaz Ahmed, Shihabur Rahman Chowdhury, Aimal Khan, Raouf Boutaba, and Jeebak Mitra. 2017. Generalized recovery from node failure in virtual network embedding. *IEEE Transactions on Network and Service Management* 14, 2 (2017), 261–274.

[24] Huasong Shan, Yuan Chen, Haifeng Liu, Yunpeng Zhang, Xiao Xiao, Xiaofeng He, Min Li, and Wei Ding. 2019. ?-diagnosis: Unsupervised and real-time diagnosis of small-window long-tail latency in large-scale microservice platforms. In *The World Wide Web Conference*. 3215–3222.

[25] Jing Shen, Jian Wan, Se-Jung Lim, and Lifeng Yu. 2018. Random-forest-based failure prediction for hard disk drives. *International Journal of Distributed Sensor Networks* 14, 11 (2018), 1550147718806480.

[26] Xiaoyi Sun, Krishnendu Chakrabarty, Ruirui Huang, Yiquan Chen, Bing Zhao, Hai Cao, Yinhe Han, Xiaoyao Liang, and Li Jiang. 2019. System-level hardware failure prediction using deep learning. In *Proceedings of DAC 2019*. 1–6.

[27] James S Supancic and Deva Ramanan. 2013. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2379–2386.

[28] Amoghavarsha Suresh and Anshul Gandhi. 2019. Using variability as a guiding principle to reduce latency in web applications via OS profiling. In *The World Wide Web Conference*. 1759–1770.

[29] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. 2012. Shifting weights: Adapting object detectors from image to video. *Advances in Neural Information Processing Systems* 25 (2012).

[30] Ye Tang, Yu-Bin Yang, and Yang Gao. 2012. Self-paced dictionary learning for image classification. In *Proceedings of the 20th ACM international conference on Multimedia*. 833–836.

[31] Yi Tay, Shuohang Wang, Luu Anh Tuan, Jie Fu, Minh C Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847* (2019).

[32] Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6095–6104.

[33] Chang Xu, Gang Wang, Xiaoguang Liu, Dongdong Guo, and Tie-Yan Liu. 2016. Health status assessment and failure prediction for hard drives with recurrent neural networks. *IEEE Trans. Comput.* 65, 11 (2016), 3502–3508.

[34] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, et al. 2018. Improving service availability of cloud systems by predicting disk error. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 481–494.

[35] Jianguo Zhang, Ji Wang, Lifang He, Zhao Li, and S Yu Philip. 2018. Layerwise perturbation-based adversarial training for hard drive health degree prediction. In *2018 IEEE International Conference on Data Mining (ICDM)*. 1428–1433.

[36] Qiao Zhang, Guo Yu, Chuanxiong Guo, Yingnong Dang, Nick Swanson, Xinsheng Yang, Randolph Yao, Murali Chintalapati, Arvind Krishnamurthy, and Thomas Anderson. 2018. Deepview: Virtual disk failure diagnosis and pattern detection for azure. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 519–532.

[37] Yu Zhang. 2015. Multi-task learning and algorithmic stability. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.