Knowledge Distillation and Beyond: From FitNet and Born-Again Networks to Noisy Time-Series Models

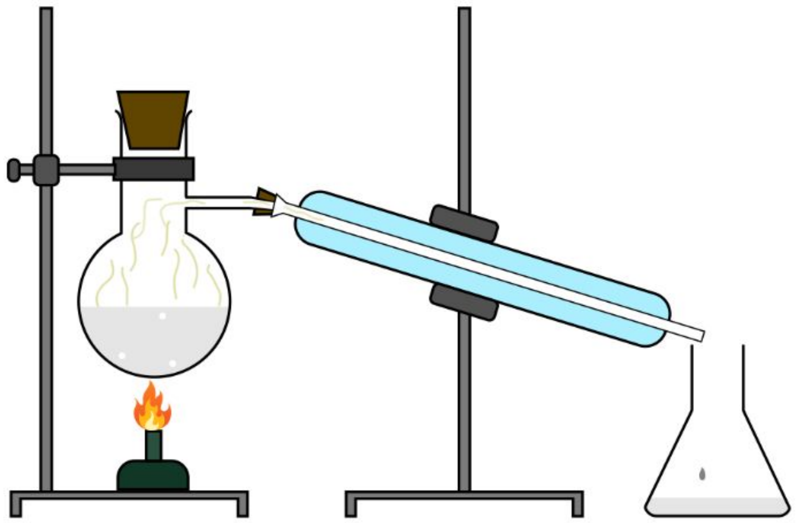
May 2, 2021

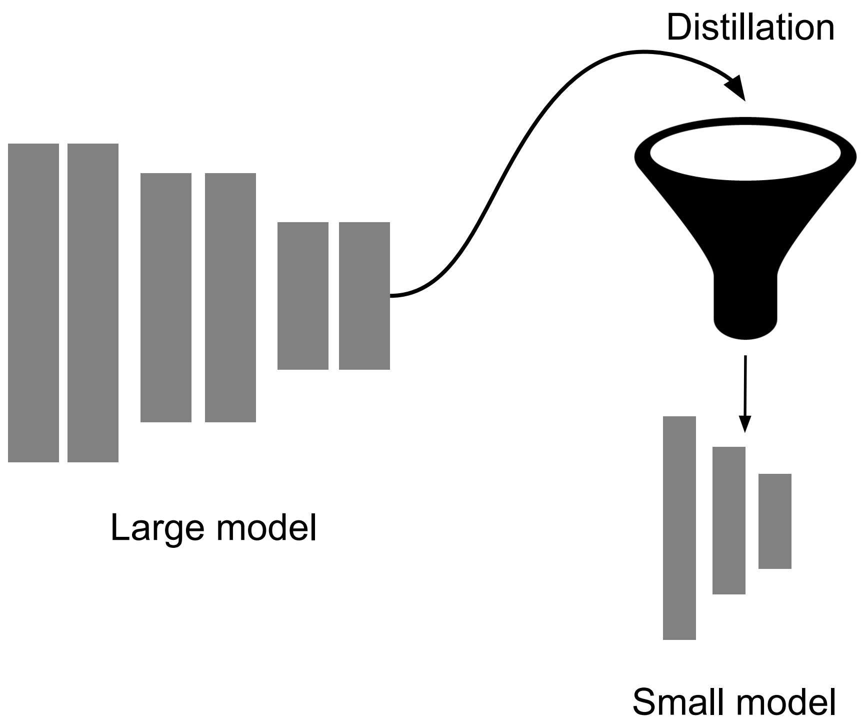
Seonyoung Kim

Contents

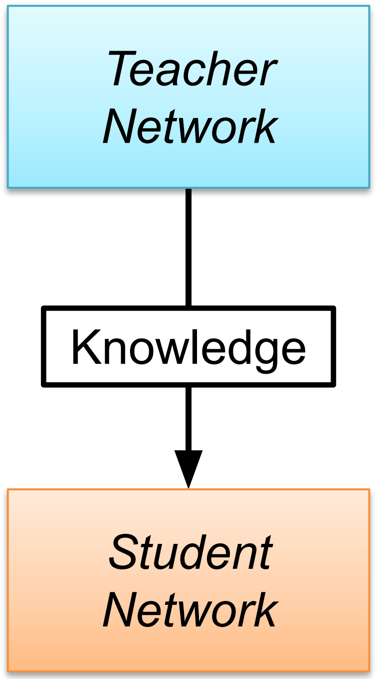
Knowledge distillation

- One of the techniques for model compression
- A method of distilling **important knowledge of a large neural network** and delivering it to a small neural network
- It retain the same or similar performance after compression





Knowledge distillation



1. Teacher network
   - Cumbersome model

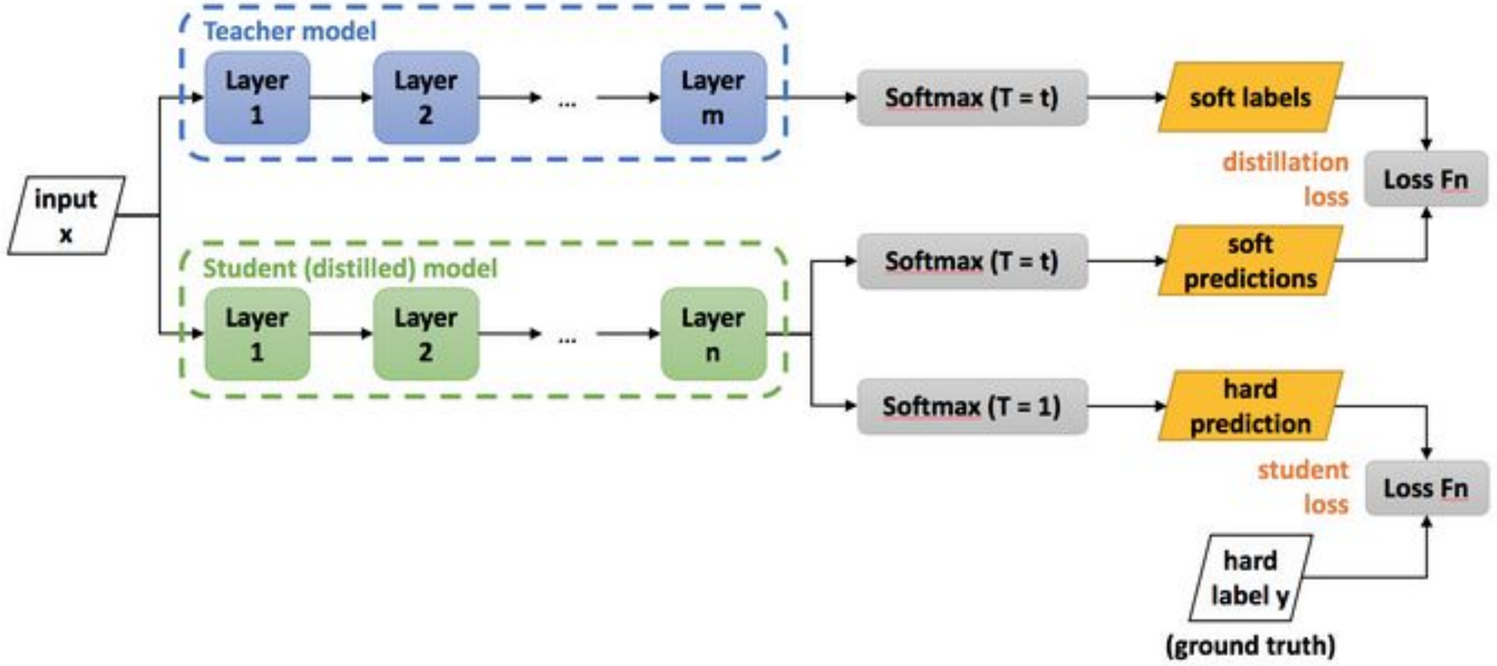
     e.g. ensemble / a large generalized model
   - (pros) excellent performance
   - (cons) computationally expensive
   - can not be deployed on devices with limited resources

2. Student network
   - Smaller model
   - (pros) fast inference
   - (cons) lower performance than the teacher network
   - suitable for deployment on devices with limited resource

Related work (1)

- Distilling the knowledge in a neural network(Hinton Geoffrey et al., NIPS 2014)
  - The baseline knowledge distillation method
  - Knowledge is softmax outputs of the teacher networks

Pre-trained model



Related work (1)

- Distilling the knowledge in a neural network(Hinton Geoffrey et al., NIPS 2014)
  - Softmax output
    - Probability distribution as the output
    - It highlights the larger value, but loses **the relativeness with other value**
  - Soft label

$$p_i = \frac{exp(\frac{z_j}{T})}{\sum_j exp(\frac{z_j}{T})}$$

p_i : *Probability of class*
z_j : *Logits of class*
T : *Temperature hyperparameter*

- Make the value of logits smaller before passing them to softmax
- It may be **a smoother probability distribution**
- Can get the relativeness with other value

dog

Cow	Dog	Cat	Car	
0	1	0	0	Ground truth(i.e., hard label)
0.005	0.9	0.084	0.001	Softmax output
0.096	0.61	0.20	0.083	Soft label

KAIST School of Computing

Knowledge distillation

- Distilling the knowledge in a neural network(Hinton Geoffrey et al., NIPS 2014)
  - Experiment
    - Dataset
      - MNIST
        - » 60,000 for training
        - » 10,000 for testing
    - Model
      - Teacher network = A neural network with two hidden layers of 1200 rectified linear hidden units(784-1200-1200-10)
      - Student network = A neural network with two hidden layers of 800 rectified linear hidden units(784-800-800-10)
    - Regularization
      - 50% dropout for all hidden units and 20% dropout for visible units
      - 100-sized minibatches with SGD
      - An exponentially decaying learning rate is used that starts at the value of 10.0
        - » Multiplied by 0.998 after each epoch of training

Knowledge distillation

- Distilling the knowledge in a neural network(Hinton Geoffrey et al., NIPS 2014)
  - Result

Network	Error
Teacher network	67 test errors
Student network w/o distillation	146 test errors
Student network with distillation	74 test errors

Born–Again Neural Networks

- ICML 2018
- Introduction
  - This paper explores knowledge distillation(KD) from the perspective of **transferring knowledge between 2 networks of identical capacity**
  - This is in contrast to much of the previous work in KD which has focused on transferring knowledge from a large network to a smaller network
  - This paper reports that these Born Again Networks(BANs) outperform their teachers by significant margins in many cases

Born-Again Neural Networks

- Approach
  - The standard KD setting is as follows:
    1. Start with an untrained network and train them(☐ Teacher network)
    2. Start with another untrained network (smaller size than the teacher network) and train it using the output of the teacher network(☐ Student network)
  - This paper augments this setting with an extra cross-entropy loss between the output of the teacher and the student networks
    - The student tried to predict the correct answer while matching the output distribution of the teacher

Born-Again Neural Networks

- ICML 2018

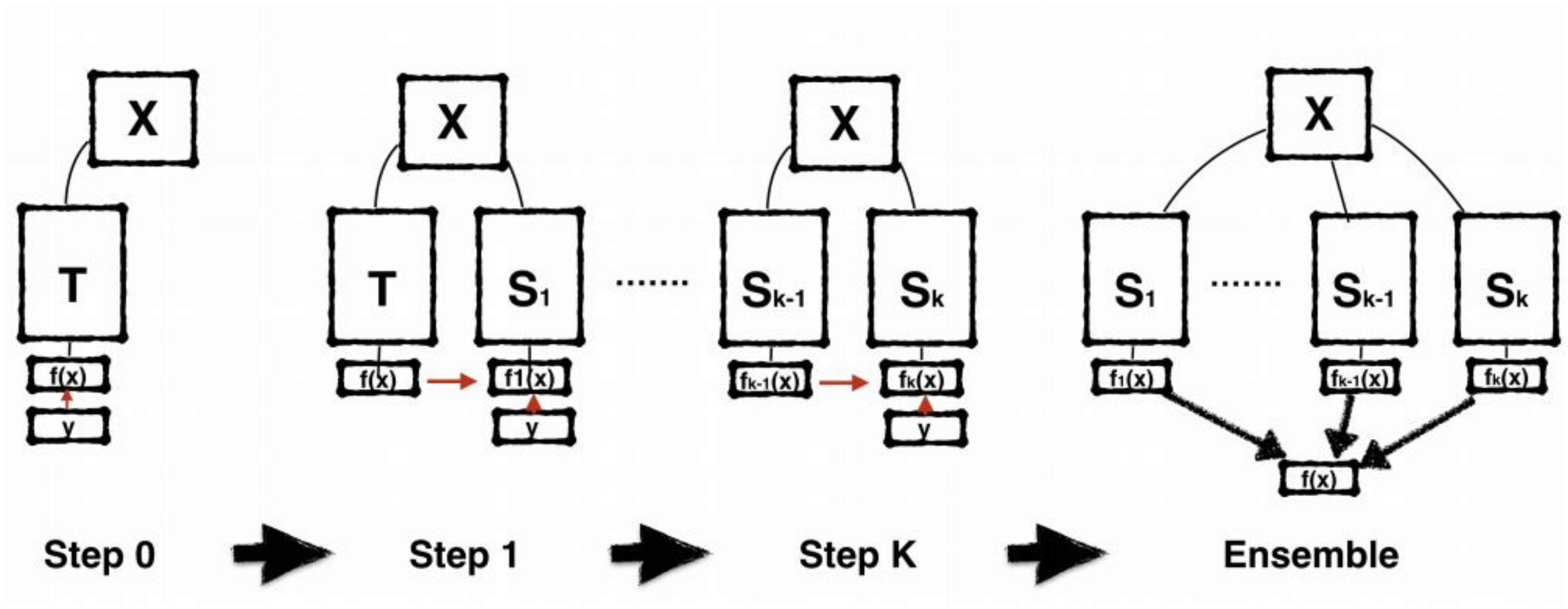


Figure 1. Graphical representation of the BAN training procedure

Born–Again Neural Networks

1. The teacher model T is trained from the labels Y
   - $\theta_1^* = arg\min_{\theta_1} \mathcal{L}(y, f(x, \theta_1))$
     - $tuples\ of\ images\ and\ labels\ (x, y) \in \mathcal{X} \times \mathcal{Y},\ f(x):\ \mathcal{X} \rightarrow \mathcal{Y}$
     - $f(x) is\ parametrized\ by\ a\ neural\ network\ f(x, \theta_1), \theta_1\ with\ parameters\ in\ some\ space\ \Theta_1$
     - $\theta_1^*\ is\ a\ model\ that\ minimizes\ some\ loss\ function$
2. At each consecutive step, a new identical model is initialized from a different random seed and trained from the supervision of the earlier generation
   - $\theta_2^* = \mathcal{L}(f\left(x, arg\min_{\theta_1} \mathcal{L}(y, f(x, \theta_1))\right), f(x, \theta_2))$
3. At the end of the procedure, additional gains can be achieved with an ensemble of multiple students generations
   - $\hat{f}^k(x) = \sum_{i=1}^{k} f(x, \theta_i)/k$: average the prediction of multiple generations of BANs

Born-Again Neural Networks – Experiment

- Image Data
  - Datasets
    - CIFAR10
    - CIFAR100
  - Baselines
    - ResNets
    - DenseNets
  - BAN Variants
    - BAN-DenseNet and BAN-ResNet
      - Train a sequence of 2 or 3 BANs using DenseNets and ResNets
    - Two settings with CWTM and DKPP
    - BAN-Resnet with DenseNet teacher and BAN-DenseNet with ResNet teacher
- Text Data
  - Datasets
    - PTB Dataset
  - Baselines
    - CNN-LSTM model

Born-Again Neural Networks – Experiment

Network	Teacher	BAN	BAN+L	CWTM	DKPP	BAN-1	BAN-2	BAN-3	Ens*2	Ens*3
DenseNet-112-33	18.25	**16.95**	17.68	17.84	17.84	17.61	17.22	**16.59**	15.77	15.68
DenseNet-90-60	17.69	**16.69**	16.93	16.93	17.43	16.62	**16.44**	16.72	15.39	15.74
DenseNet-80-80	17.16	**16.36**	16.5	16.5	16.84	16.26	16.30	**15.5**	15.46	15.14
DenseNet-80-120	16.87	**16.00**	16.41	16.41	16.34	**16.13**	16.13	/	**15.13**	**14.9**

Table 1. Test error on CIFAR-100

- BAN : trained only with the teacher loss
- BAN+L : trained with label and teacher loss
- BAN-1, BAN-2, BAN-3 : sequence of BAN-DenseNet
  - BAN and BAN-1 are trained from Teacher but have different random seeds

Born-Again Neural Networks – Experiment

Network	Teacher	BAN	BAN+L
Wide-ResNet-28-1	30.05	29.43	24.93
Wide-ResNet-28-2	25.32	24.38	18.49
Wide-ResNet-28-5	20.88	20.93	17.52
Wide-ResNet-28-10	19.08	18.25	16.79

Table 2. Test error on CIFAR-10

Network	Parameters	Teacher Val	BAN+L Val	Teacher Test	BAN+L Test
ConvLSTM	19M	83.69	80.27	80.05	76.97
LSTM	52M	75.11	71.19	71.87	68.56

Table 3. Validation/Test perplexity on PTM for BAN-LSTM language model of different complexity

Motivation

- The amount of real world time series are limited mainly due to the difficulties of finding labelled real world time-series
- The auto-encoder (AE) is a type of artificial neural network, which can learn efficient data coding using an unsupervised manner
- However, the AE doesn't perform well when data samples are very different
- Time-series data are complicated in that there are multiple states and patterns in the normal data

Denoising Autoencoder

- Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion(2010)
  - Observed that the reconstruction criterion alone is unable to guarantee the extraction of useful features
  - It can lead to the obvious solution "simply copy the input"
  - **Change the reconstruction criterion** for a both more challenging and more interesting objective

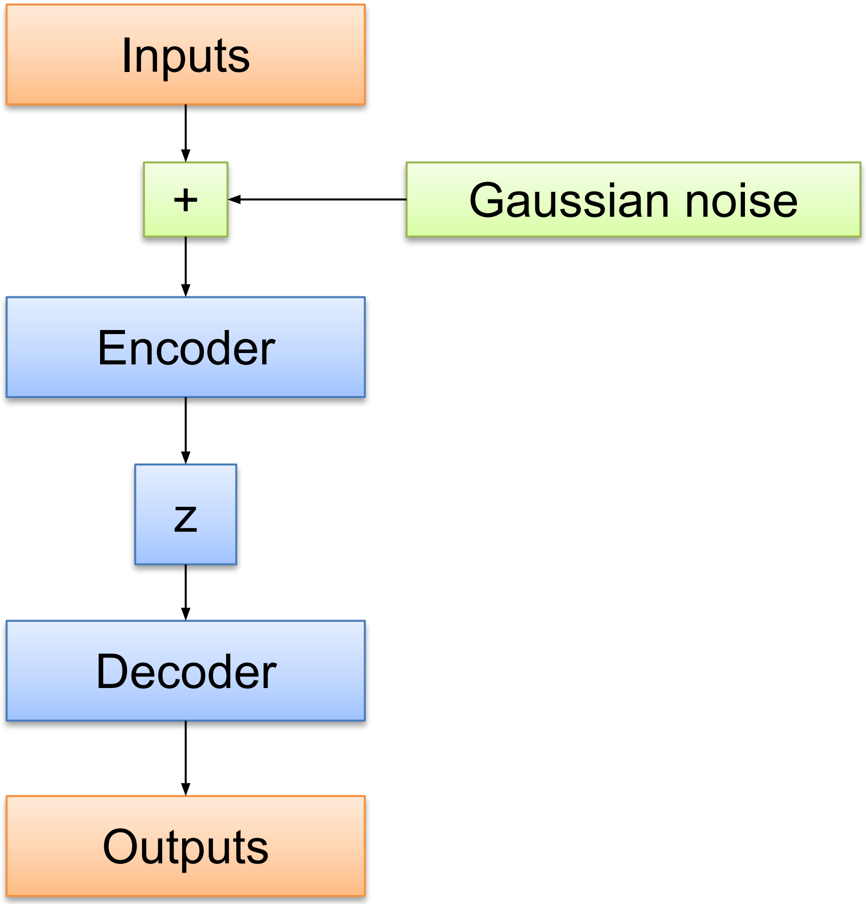
Denoising Autoencoder

- Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion(2010)

1. Add gaussian noise to input data(i.e., a corrupted version of input)
2. Corrupted input is mapped with the basic autoencoder
3. Decoder is trained to reconstruct the original input

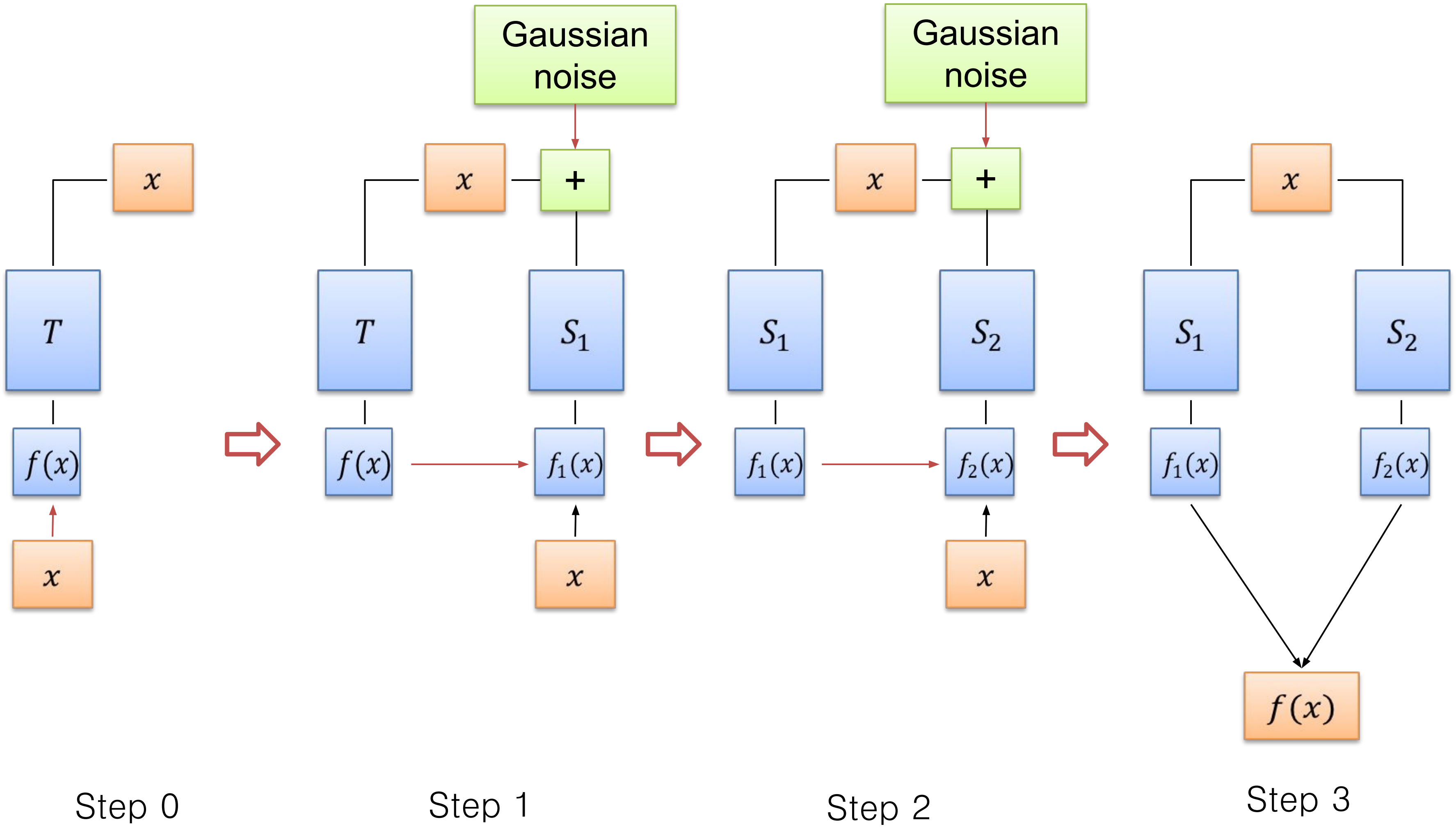
$$\mathcal{L} = \mathcal{L}_{MSE}(x, f(x_{noise}))$$

☐ Extract useful features, not simply copy input



Proposed approach

- Training with Noisy time-series data

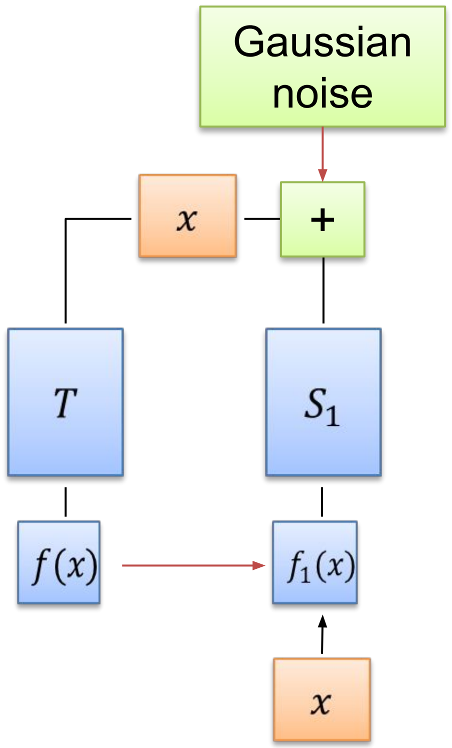


Proposed approach

- Training with Noisy time-series data

1. Add noise to data of student
2. Student model is trained to minimize the reconstruction error with reconstructed sequence from teacher model and original sequence

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{MSE}(x, f_i(x_{noise})) + \alpha\mathcal{L}_{MSE}(f_i(x), f_{i-1}(x))$$

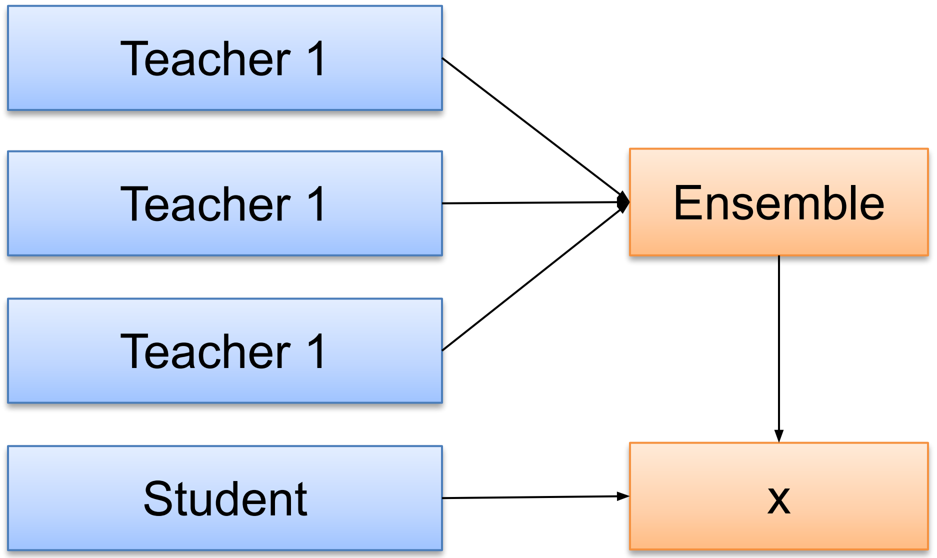


Proposed approach

- Training with Noisy time-series data
  - Add different noise to each sample of Teacher
    - use a standard normal distribution to add different noises to each sample in the dataset, where the mean of normal distribution is μ and the standard deviation is σ_2
    - only add small random noise to the source data, here the value of μ is 0 and the value of σ_2 is 0.05
  - Select samples from the mini-batch with some fixed probability α
  - Soft label was not used as anomaly was detected using reconstruction error

Experiment

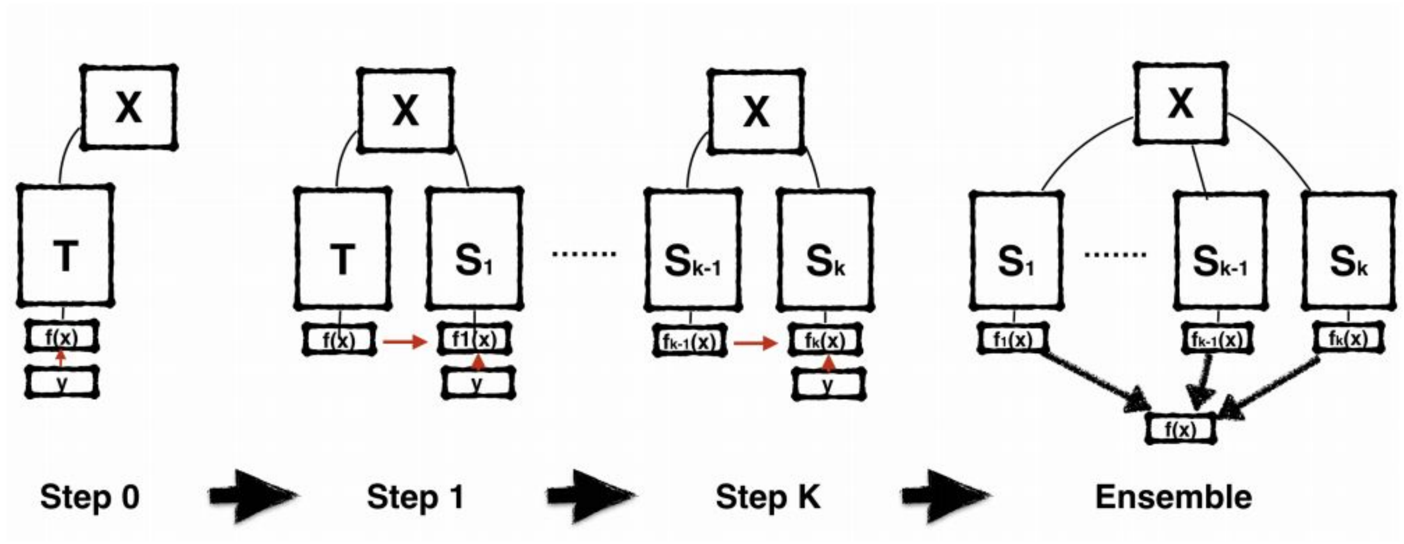
1. Knowledge Distillation



- KD is trained using ensemble of networks as the teacher

Experiment

2. Sequential training



– Ensemble

- $Ensemble\ of\ teacher = \frac{\sum_{i=1}^{k} Reconstructed\ sequence\ of\ S_i}{k}$

Experiment

- Time-series data
  - Datasets
    - SMD(Server machine dataset)
  - Baselines
    - LSTM-Autoencoder
  - Variants
    - Knowledge Distillation
    - BAN
    - BAN with noisy data

Experiment

Method	Teacher			Student-1			Student-2			Student-3		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
KD	0.3937	0.6402	0.4031	0.4243	0.6430	0.4050	-	-	-	-	-	-
Sequential training	0.3937	0.6402	0.4031	0.4243	0.6430	0.4050	0.3957	0.6507	0.4112	0.3755	0.6470	0.4072
Sequential training with noisy data	0.3937	0.6402	0.4031	0.3979	0.6490	0.4171	0.3957	0.6514	0.4182	0.3827	0.6468	0.4012

Method	Ensemble*2			Ensemble*3		
	P	R	F1	P	R	F1
Sequential training	0.3757	0.6561	0.4062	0.3857	0.6595	0.4115
Sequential training with noisy data	0.3857	0.6713	0.4129	0.3986	0.6793	0.4241