FOSTER: Feature Boosting and Compression for Class-Incremental Learning

Shiwon Kim,
Digital Healthcare Lab,
Department of Digital Analytics,
Yonsei University College of Computing

Abstract

- Background
  - The ability to learn new concepts continually is necessary in this ever-changing world
  - Deep neural networks suffer from **catastrophic forgetting** when learning new concepts
  - Stability-plasticity dilemma or **too much computation or storage overhead**
- Proposed method
  - **F**eature bo**OST**ing and compr**E**ssion for class-inc**R**emental learning (FOSTER)
  - A novel two-stage learning paradigm for adaptive learning of new categories
    1) Dynamically expand new modules to fit the residual between the target and the previous output
    2) Remove redundant parameters and features via distillation to maintain a single backbone

* Saving all the streaming data for offline training (requires an unlimited memory budget for storage)

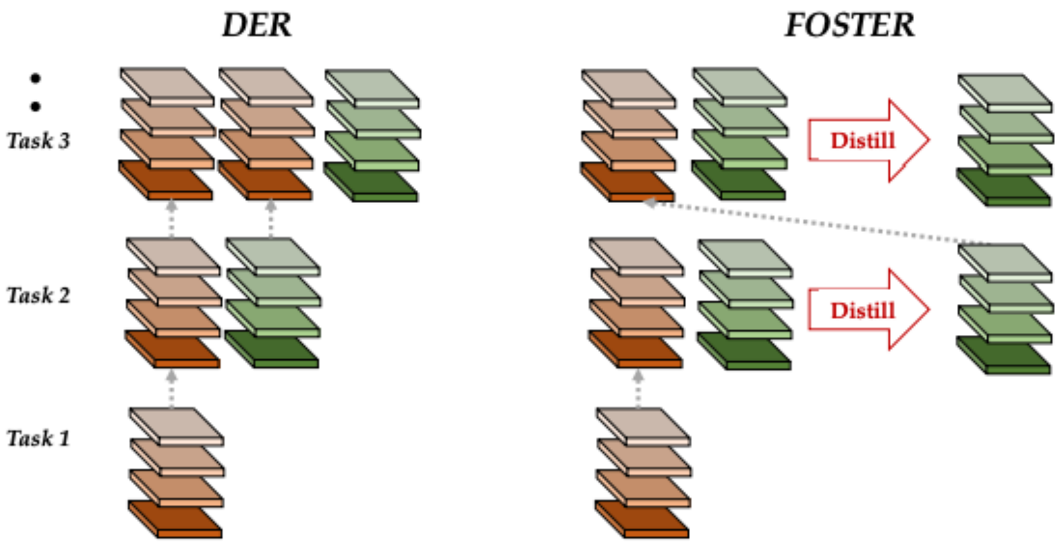
Introduction

- Class-incremental learning (CIL)
  - Retraining a model every time new classes emerge is impractical due to high training costs
  - Directly fine-tuning the original neural networks on new data causes **catastrophic forgetting**
- Knowledge distillation (KD)
  - The most widely recognized and utilized class-incremental learning strategy
  - Constrain the **outputs of the new model for old tasks** to be similar to that of the old model
- Dynamic architectures
  - Methods based on dynamic architectures achieve state-of-the-art performance
  - Preserve old modules (parameters frozen) and expand new trainable modules for new classes
  - Increase in the number of parameters and inconsistency between the old and new features

Introduction

- FOSTER
  - A two-step novel perspective from gradient boosting

1) Apply **feature level boosting** to alleviate the performance decline in incremental settings
2) Eliminate **redundant parameters** and meaningless dimensions caused by feature boosting



19. Wang, Fu Lee et al. "FOSTER: Feature Boosting and Compression for Class-Incremental Learning." European Conference on Computer Vision (2022).

Introduction

- Gradient boosting
  - Iteratively trains the weak learners that **predict the error (residual)** between the outputs
  - $e_t(x_i)$ is decomposed into round t prediction $h_t(x_i)$ and the residual $e_{t+1}(x_i)$; $h_0(x_i) = 0$

$$y_i = h_0(x_i) + e_1(x_i)$$

$$e_1(x_i) = h_1(x_i) + e_2(x_i)$$

$$e_2(x_i) = h_2(x_i) + e_3(x_i)$$

$$\dots$$

$$e_t(x_i) = h_t(x_i) + e_{t+1}(x_i)$$

$Error$ $decreases$

Methods

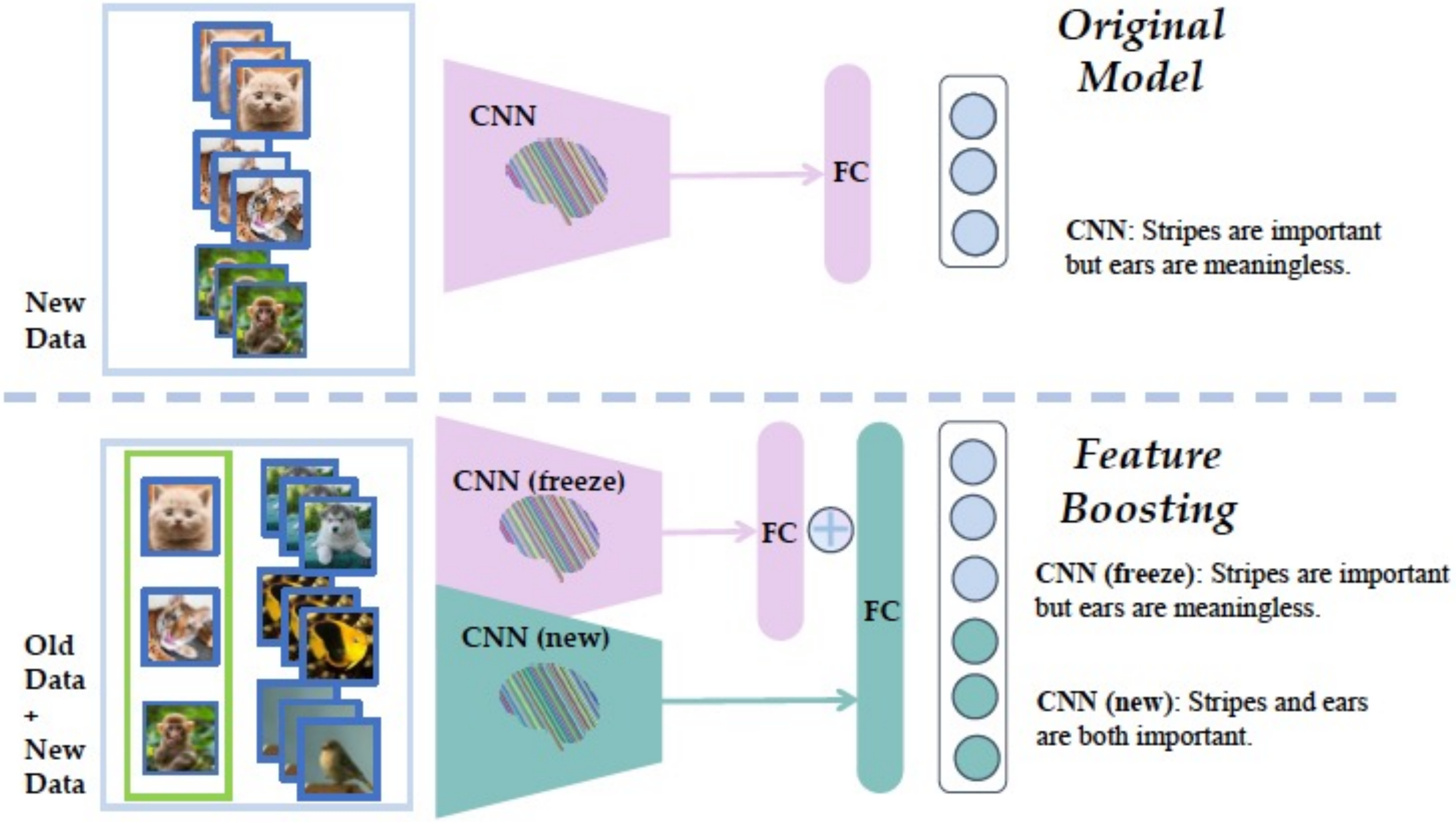
I. Feature boosting

- Freeze the old model and create **a new module to fit the residuals** from the old model
- The new module helps the model learn both the old and new classes better

II. Feature compression

- Remove insignificant dimensions and parameters to make **compact representations**
- **Instruct the compressed model** using the outputs of the dual branch model
- Different weights are assigned to old and new classes to alleviate the classification bias

Feature Boosting



Feature Boosting

- Incremental settings
  - Assume in the t^{th} stage, F_{t-1} is the model saved from the last stage
  - F_{t-1} can be further decomposed into **feature embedding** and **linear classifier**

$$\mathrm{F}_{t-1}(\boldsymbol{x}) = (\mathbf{W}_{t-1})^{\top}\Phi_{t-1}(\boldsymbol{x})$$

$$where\ \ \Phi_{t-1}(\cdot) : \mathbb{R}^{D} \rightarrow \mathbb{R}^{d},\ \ \mathbf{W}_{t-1} \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_{t-1}|}$$

  - Directly fine-tuning F_{t-1} on new data will impair its capacity for old classes
  - Simply freezing F_{t-1} causes it to lose plasticity for new classes
  - What if we train a new model to **fit the residuals** between target y and $\mathrm{F}_{t-1}(\boldsymbol{x})$?

Feature Boosting

- Training process
  - The new model $\mathcal{F}_t$ consists of a feature extractor $\phi_t(\cdot)$ and a linear classifier $\mathcal{W}_t$

$$\phi_t(\cdot):\ \mathbb{R}^D \to \mathbb{R}^d,\ \mathcal{W}_t \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_t|},\ \mathcal{W}_t = \left[\mathcal{W}_t^{(o)}, \mathcal{W}_t^{(n)}\right]$$

$$where\ \mathcal{W}_t^{(o)} \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_{t-1}|},\ \mathcal{W}_t^{(n)} \in \mathbb{R}^{d \times |\mathcal{Y}_t|}$$

  - The **training process** can be represented as below (let $\ell(\cdot,\cdot)$ be the mean squared error)

$$\mathrm{F}_t(\boldsymbol{x}) = \mathrm{F}_{t-1}(\boldsymbol{x}) + \arg\min_{\mathcal{F}_t} \mathbb{E}_{(\boldsymbol{x},\, y)\in \hat{\mathcal{D}}_t} \left[\ell(y,\, \mathrm{F}_{t-1}(\boldsymbol{x}) + \mathcal{F}_t(\boldsymbol{x}))\right]$$

Feature Boosting

- Ideal expression of target y
  - We expect $\mathcal{F}_t(\boldsymbol{x})$ can fit residuals of y and $\mathrm{F}_{t-1}(\boldsymbol{x})$ for every $(\boldsymbol{x}, y) \in \widehat{\mathcal{D}}_t$
  - $S(\cdot)$ is the softmax and $\mathbf{O} \in \mathbb{R}^{d \times |\mathcal{Y}_t|}$ set to zero matrix or fine-tuned on $\widehat{\mathcal{D}}_t$ with Φ_{t-1} frozen
  - Set $\mathbf{O}$ to zero matrix as default for experiments

$$\boldsymbol{y} = \mathrm{F}_{t-1}(\boldsymbol{x}) + \mathcal{F}_t(\boldsymbol{x}) = S\left(\begin{bmatrix} \mathbf{W}_{t-1}^{\top} \\ \mathbf{O} \end{bmatrix} \Phi_{t-1}(\boldsymbol{x})\right) + S\left(\begin{bmatrix} (\mathcal{W}_t^{(o)})^{\top} \\ (\mathcal{W}_t^{(n)})^{\top} \end{bmatrix} \phi_t(\boldsymbol{x})\right)$$

Feature Boosting

- Optimization problem
  - Denote the parameters of $\mathcal{F}_t$ as θ_t and $\mathrm{Dis}(\cdot,\cdot)$ as a **distance matrix** (e.g., Euclidean metric)

$$\theta_t^* = \arg\min_{\theta_t} \mathrm{Dis}\left(\boldsymbol{y}, S\left(\begin{bmatrix} \mathbf{W}_{t-1}^\top \\ \mathbf{O} \end{bmatrix} \Phi_{t-1}(\boldsymbol{x}) \right) + S\left(\begin{bmatrix} (\mathcal{W}_t^{(o)})^\top \\ (\mathcal{W}_t^{(n)})^\top \end{bmatrix} \phi_t(\boldsymbol{x}) \right) \right)$$

  - *Replace $S(\cdot) + S(\cdot)$ with $S(\cdot + \cdot)$ and substitute $\mathrm{Dis}(\cdot,\cdot)$ for the* ***Kullback-Leibler divergence***

$$\theta_t^* = \arg\min_{\theta_t} \mathrm{KL}\left(\boldsymbol{y} \,\middle\|\, S\left(\begin{bmatrix} \mathbf{W}_{t-1}^\top & (\mathcal{W}_t^{(o)})^\top \\ \mathbf{O} & (\mathcal{W}_t^{(n)})^\top \end{bmatrix} \begin{bmatrix} \Phi_{t-1}(\boldsymbol{x}) \\ \phi_t(\boldsymbol{x}) \end{bmatrix} \right) \right)$$

Feature Boosting

- Module expansion
  - F_t consists of an expanded classifier $\mathbf{W}_t$ and a concatenated super feature extractor $\Phi_t(\cdot)$

$$\mathbf{W}_t^\top = \begin{bmatrix} \mathbf{W}_{t-1}^\top & (\mathcal{W}_t^{(o)})^\top \\ \mathbf{O} & (\mathcal{W}_t^{(n)})^\top \end{bmatrix}, \qquad \Phi_t(\boldsymbol{x}) = \begin{bmatrix} \Phi_{t-1}(\boldsymbol{x}) \\ \phi_t(\boldsymbol{x}) \end{bmatrix}$$

  - $\mathbf{W}_{t-1}^\top$, $\mathbf{O}$, Φ_{t-1} are all frozen and the trainable modules are ϕ_t, $\mathcal{W}_t^{(o)}$, $\mathcal{W}_t^{(n)}$

$$\mathrm{F}_t(x) = \mathbf{W}_t^\top \Phi_t(\boldsymbol{x}) = \begin{bmatrix} \mathbf{W}_{t-1}^\top \Phi_{t-1}(\boldsymbol{x}) + (\mathcal{W}_t^{(o)})^\top \phi_t(\boldsymbol{x}) \\ (\mathcal{W}_t^{(n)})^\top \phi_t(\boldsymbol{x}) \end{bmatrix}$$

Feature Boosting

- Logits of F_t explained
  a) The logits of the old classes ensure the new module to fit the residuals between y and F_{t-1}
  b) The logits of new classes guide the new module $\mathcal{F}_t$ to learn to correctly classify new classes

$$\mathrm{F}_t(x) = \mathbf{W}_t^\top \Phi_t(\boldsymbol{x}) = \begin{bmatrix} \mathbf{W}_{t-1}^\top \Phi_{t-1}(\boldsymbol{x}) + (\mathcal{W}_t^{(o)})^\top \phi_t(\boldsymbol{x}) \\ (\mathcal{W}_t^{(n)})^\top \phi_t(\boldsymbol{x}) \end{bmatrix} \begin{matrix} \cdots & a) \\ \cdots & b) \end{matrix}$$

Feature Boosting

- Calibration for old and new
  - **Imbalanced training set** when training on new tasks ($\widehat{\mathcal{D}}_t = \mathcal{D}_t \cup \mathcal{V}_t$)
  - The imbalance on categories of $\widehat{\mathcal{D}}_t$ result in a strong classification bias in the model
- Logits alignment (LA)
  - Strengthen the learning of old instances and mitigate the classification bias
  - **Add scale factor** γ (a diagonal matrix) to the logits of the old and new classes

$$\gamma \mathbf{W}_t^\top \Phi_t(\boldsymbol{x}) = \begin{bmatrix} \gamma_1 \left(\mathbf{W}_{t-1}^\top \Phi_{t-1}(\boldsymbol{x}) + \left(\mathcal{W}_t^{(o)}\right)^\top \phi_t(\boldsymbol{x}) \right) \\ \gamma_2 (\mathcal{W}_t^{(n)})^\top \phi_t(\boldsymbol{x}) \end{bmatrix}, \qquad 0 < \gamma_1 < 1, \;\; \gamma_2 > 1$$

Feature Boosting

- Logits alignment continued
  - The absolute value of logits reduced for old classes and enlarged for new classes
  - Force F_t to produce **larger logits for old classes** and **smaller logits for new classes**
  - Scale factors γ_1, γ_2 are acquired from the normalized effective number E_n of each class

$$E_n = \begin{cases} \dfrac{1-\beta^n}{1-\beta}, & \beta \in [0,1) \\ n, & \beta = 1 \end{cases}$$

$$(\gamma_1, \gamma_2) = \left(\frac{E_{n_{old}}}{E_{n_{old}} + E_{n_{new}}}, \frac{E_{n_{new}}}{E_{n_{old}} + E_{n_{new}}} \right)$$

Cui, Yin, et al. "Class-balanced loss based on effective number of samples." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2019).

Feature Boosting

- Feature enhancement (FE)
  - Assume an extreme scenario where the **residuals of** $\mathrm{F}_{t-1}(\boldsymbol{x})$ **and** y **is zero**
  - The new module will not learn anything about the old classes and damage the performance
  - Prompt the new module $\mathcal{F}_t$ to further learn old categories through feature enhancement
  - Initialize a new linear classifier $\mathbf{W}_t^{(a)} \in \mathbb{R}^{d \times |\hat{\mathcal{Y}}_t|}$ for the **classification of all seen classes**
  - The new feature extractor can still learn to classify the old classes even if the residual is zero

$$\mathcal{L}_{FE} = \mathrm{KL}\left(\boldsymbol{y} \,\middle\|\, \mathcal{S}\left(\left(\mathbf{W}_t^{(a)}\right)^{\top} \phi_t(\boldsymbol{x})\right)\right)$$

Feature Boosting

- Feature enhancement continued
  - Training $\phi_t(\cdot)$ in an imbalanced dataset might lead to **overfitting to small classes**
  - **Knowledge distillation** to make $\mathrm{F}_t(\boldsymbol{x})$ have similar output distribution as $\mathrm{F}_{t-1}(\boldsymbol{x})$ on old data
  - Learn a feature representation with good generalizability for old classes

$$\mathcal{L}_{KD} = \mathrm{KL}\left(S(\mathrm{F}_{t-1}(\boldsymbol{x})) \middle\| S\left(\mathrm{F}_{t-1}(\boldsymbol{x}) + \left(\mathcal{W}_t^{(o)}\right)^{\top} \phi_t(\boldsymbol{x})\right)\right)$$

Feature Boosting

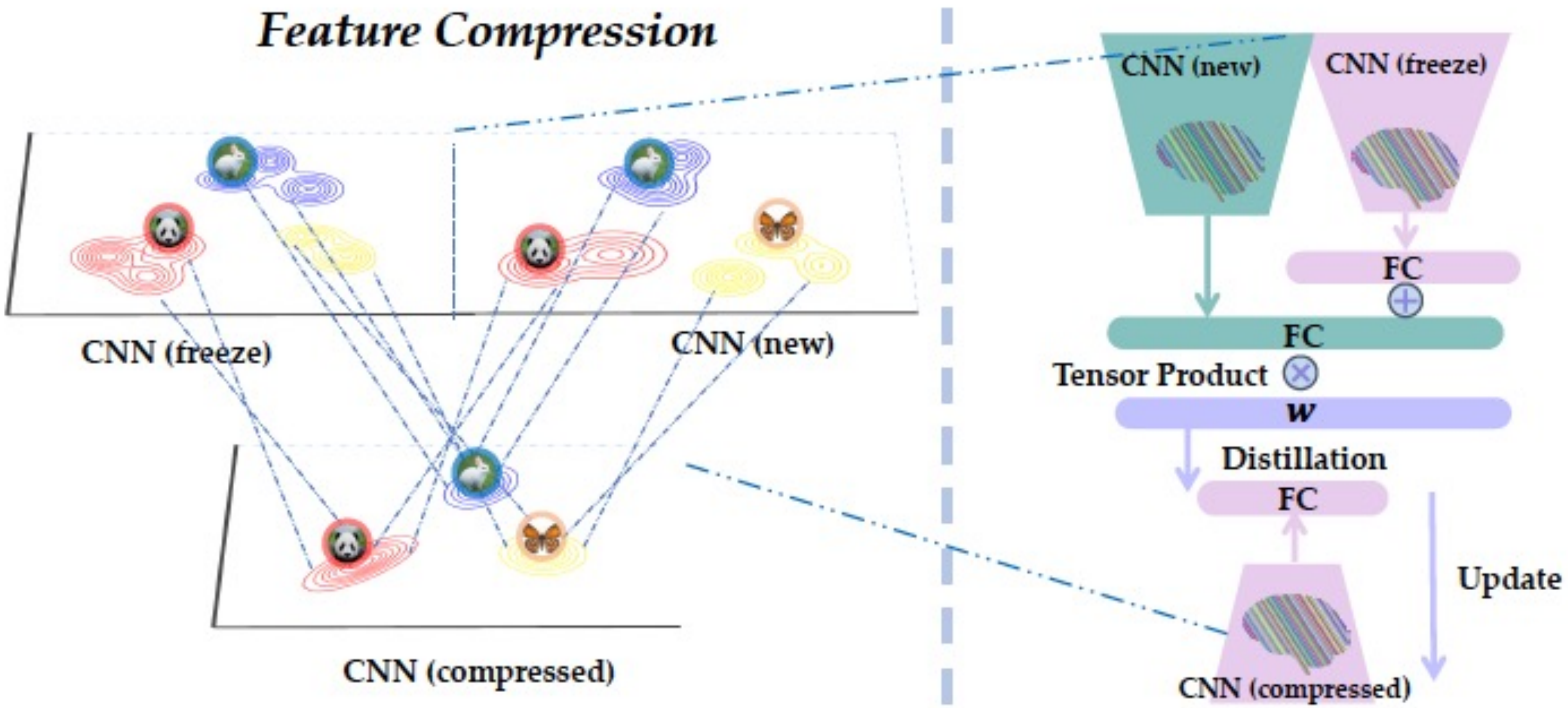
- Loss function
  - The final FOSTER loss for boosting combines the three components aforementioned

$$\mathcal{L}_{Boosting} = \mathcal{L}_{LA} + \mathcal{L}_{FE} + \mathcal{L}_{KD}$$

  - Logits alignment loss is as follows:

$$\mathcal{L}_{LA} = \mathrm{KL}\left(\boldsymbol{y} \,\middle\|\, \mathcal{S}\left(\gamma \mathbf{W}_t^{\top} \Phi_t(\boldsymbol{x})\right)\right)$$

Feature Compression



Feature Compression

- Loss function
  - Gradually adding a new module leads to overfull of parameters and feature dimensions
  - **Compress the expanded feature space** to a smaller one for long-term applicability
  - **Knowledge distillation** is a simple yet effective way to achieve this goal
- Balanced distillation (BKD)
  - Suppose there is a **single backbone** student model $\mathrm{F}_t^{(s)}$ to be distilled
  - Adjust the weights of distilled information for different classes to mitigate classification bias
  - $\mathcal{w}$ is the weighted vector obtained from E_n and $\otimes$ refers to the tensor product

$$\mathcal{L}_{BKD} = \mathrm{KL}\left(\mathcal{w} \otimes S(\mathrm{F}_t(\boldsymbol{x})) \,\middle\|\, S\left(\mathrm{F}_t^{(s)}(\boldsymbol{x})\right)\right)$$

YONSEI UNIVERSITY
COLLEGE OF MEDICINE