Week 4 Lecture 1

0 mins

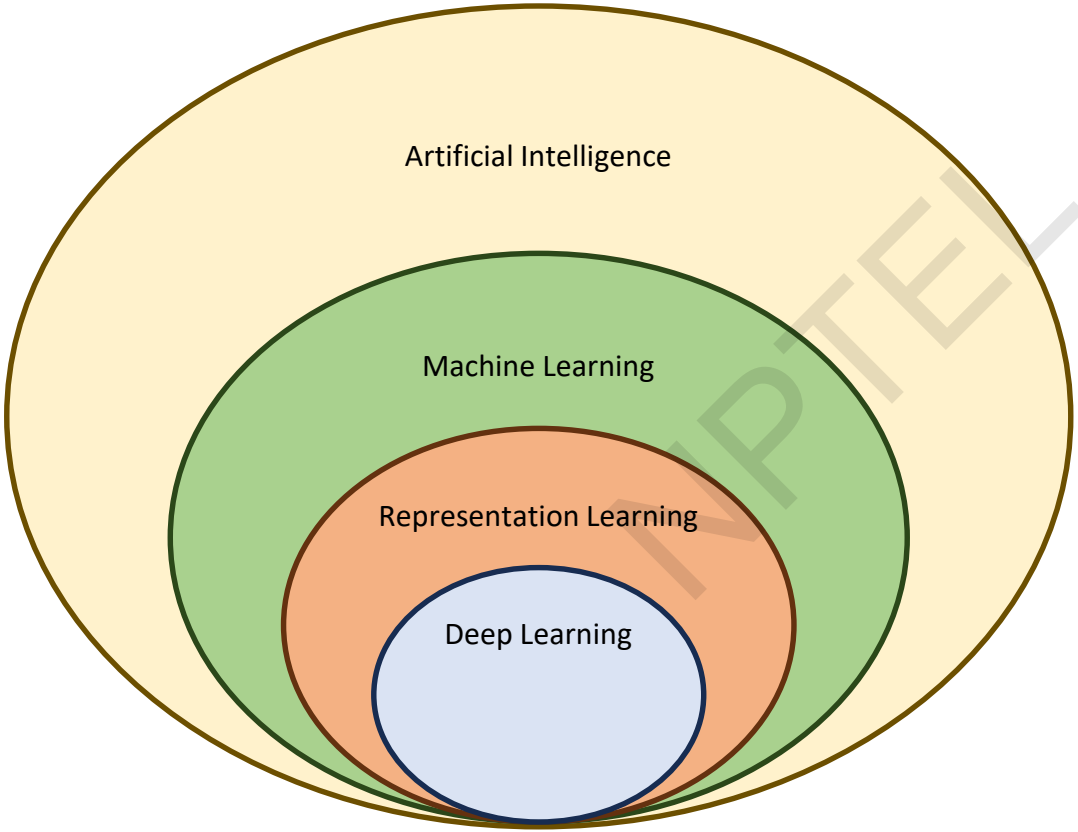
Edge AI - Intelligence at the Edge

Dr. Rajiv Misra, Professor
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Contents

- Review of Deep Learning
- Optimizing AI for the Edge
- Knowledge Distillation
- Federated Learning

Review of Deep Learning



Source: Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press 2016

Review of Deep Learning

Source: Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press 2016

Optimizing Models

Model compression and acceleration techniques can be divided in the following categories:

- **Parameter pruning and sharing**:
  - **model quantization**
  - **model binarization**
  - **parameter sharing**
- **Low-rank factorization**
- **Transferred compact convolutional filters**
- **Knowledge distillation (KD)**

Knowledge Distillation

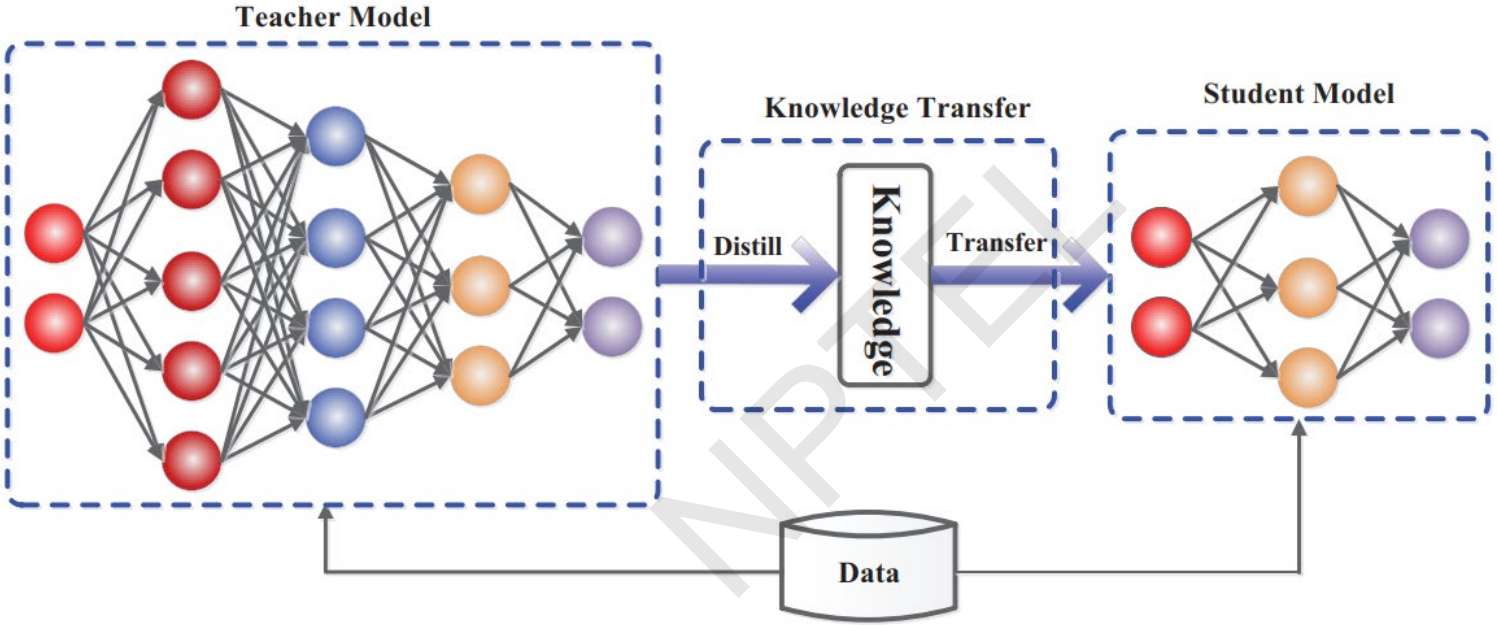
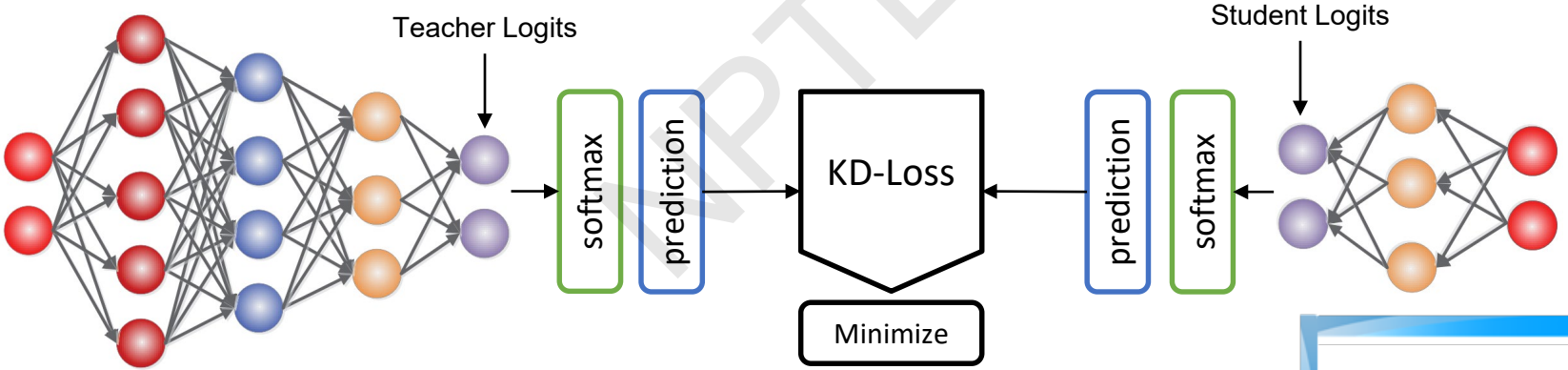


Figure Source: *Knowledge Distillation A Survey Jianping Gou, Baosheng Yu, Stephen J. Maybank, Dacheng Tao in arXiv:2006.05525v7*

Knowledge Distillation

- In vanilla knowledge distillation, knowledge is transferred from the teacher model to the student model by minimizing a loss function in which the target is the distribution of class probabilities predicted by the teacher model.
- Specifically, Knowledge Distillation (KD) is accomplished by minimizing the Kullback-Leibler (KL) divergence between the predictions of teacher and student.



Knowledge Distillation KL-Divergence

- KL divergence is a measure of how much a probability distribution diverges from another.

For discrete probability distributions P and Q defined on the same probability space, $\mathcal{X}$, the KL divergence from P to Q is:

$$D_{KL}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log(\frac{P(x)}{Q(x)}). \quad (1)$$

For distributions P and Q of a continuous random variable, the KL divergence from P to Q is:

$$D_{KL}(P||Q) = \int_{x \in \mathcal{X}} P(x) \log(\frac{P(x)}{Q(x)}) dx. \quad (2)$$

Non-negativity

$$D_{\mathrm{KL}}(P\|Q) \geq 0.$$

Not Symmetric

$$D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$$

Knowledge Distillation

We define a teacher network $T(.)$ and a student network $S(.)$. Typically, networks can produce class probabilities that converts the logits z_i, computed for each class into a probability, $\boldsymbol{q}_i$ by using softmax function. We define the class probabilities $\boldsymbol{p}_i$ which is generated by teacher network $T(.)$ and $\boldsymbol{q}_i$ which is generated by student network $S(.)$ correspondingly. Then the KD Loss can be given by:

$$\mathcal{L}_{KD} = -\sum_{i=1}^{N} p(\boldsymbol{x}_i) \log(q(\boldsymbol{x}_i)) \tag{3}$$

where $p(\boldsymbol{x}_i) = \frac{\exp(\boldsymbol{v}_i/\tau)}{\sum_{j=1}^{C-1} \exp(\boldsymbol{v}_j/\tau)}$ and $q(\boldsymbol{x}_i) = \frac{\exp(\boldsymbol{z}_i/\tau)}{\sum_{j=1}^{C-1} \exp(\boldsymbol{z}_j/\tau)}$. Here C denotes the number of classes, τ iss the temperature parameter. $\boldsymbol{v}$ and $\boldsymbol{z}$ indicates the logits generated by teacher network and student network respectively.

Knowledge Distillation

If we regard the class probablities p as soft label, then the KL Loss can be regarded as a softmax cross entropy. It's not hard to derive the relationships between cross entropy and KL divergence. Given p and q, we define entropy as $H(.)$, we can have:

$$\mathcal{L}_{KD} = -\sum_{i=1}^{N} p(\boldsymbol{x}_i) \log(q(\boldsymbol{x}_i)) = H(p, q). \tag{4}$$

$$\begin{aligned} D_{KL}(p||q) &= \mathbb{E}_p[-\log \frac{q}{p}] \\ &= \mathbb{E}_p[-\log q + \log p] \\ &= \mathbb{E}_p[-\log q] - \mathbb{E}_p[-\log p] \\ &= H(p, q) - H(p) \end{aligned} \tag{5}$$

Then we can have:

$$\mathcal{L}_{KD} = D_{KL} + H(p) \tag{6}$$

Knowledge Distillation

In knowledge distillation, we attempt to optimize the student network that can mimic the teacher network, then we can rewrite the our loss function:

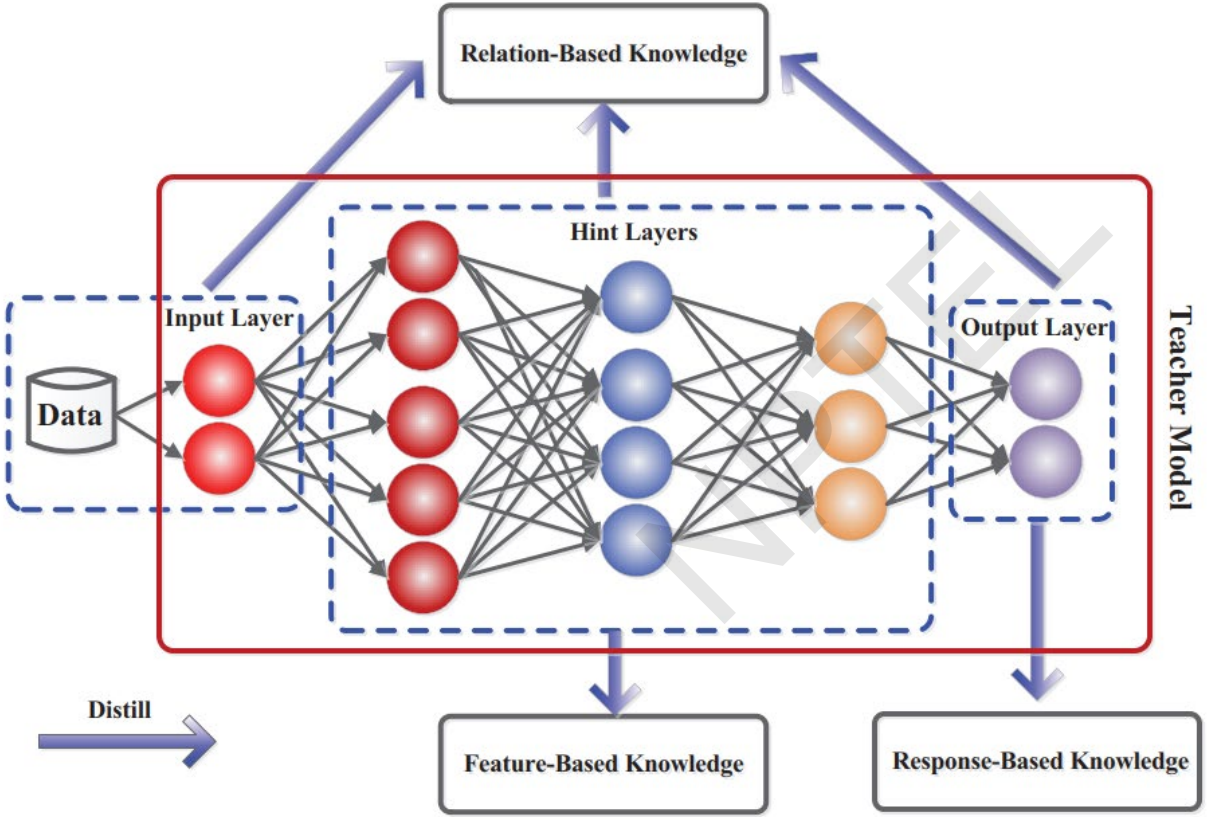
$$\begin{aligned} \mathcal{L}_{KD} &= H(p, q_\theta) \\ &= D_{KL}(p||q_\theta) + H(p) \end{aligned} \tag{7}$$

Since $H(p)$ is independent of θ, the optimization goal then becomes:

$$\begin{aligned} \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{KD} &= \underset{\theta}{\operatorname{argmin}}\, H(p, q_\theta) \\ &= \underset{\theta}{\operatorname{argmin}}\, D_{KL}(p||q_\theta) \\ &= \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{KL} \end{aligned} \tag{8}$$

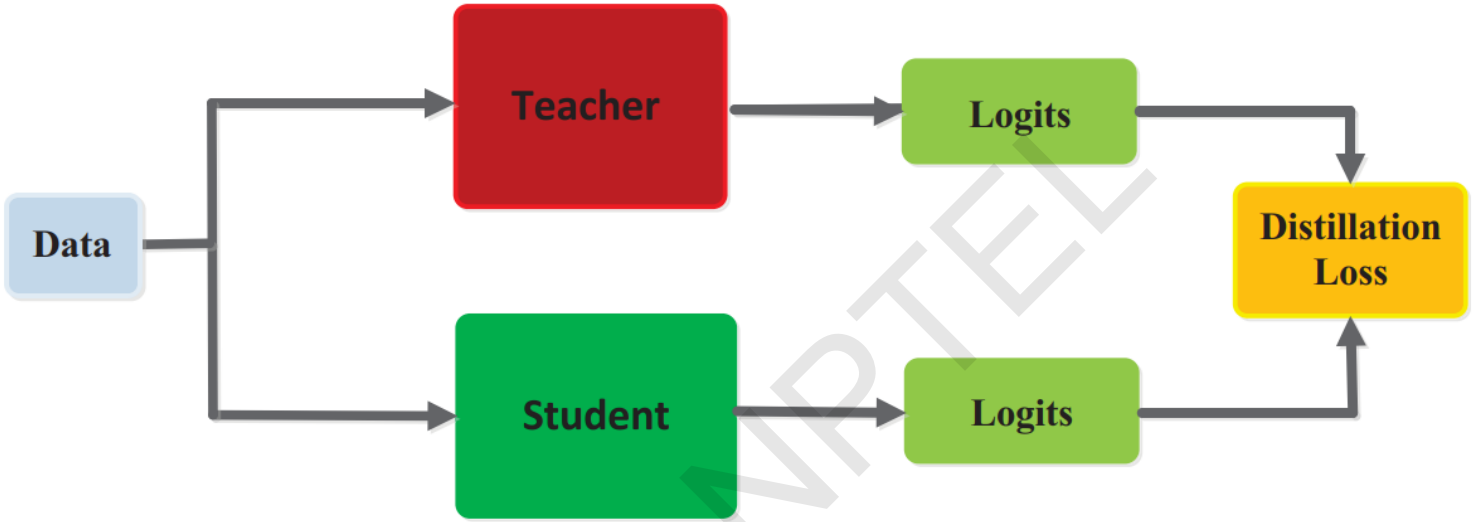
Then we can find that optimizing KD loss is equivalent to optimize KL Loss in the knowledge distillation setting.

Knowledge Distillation

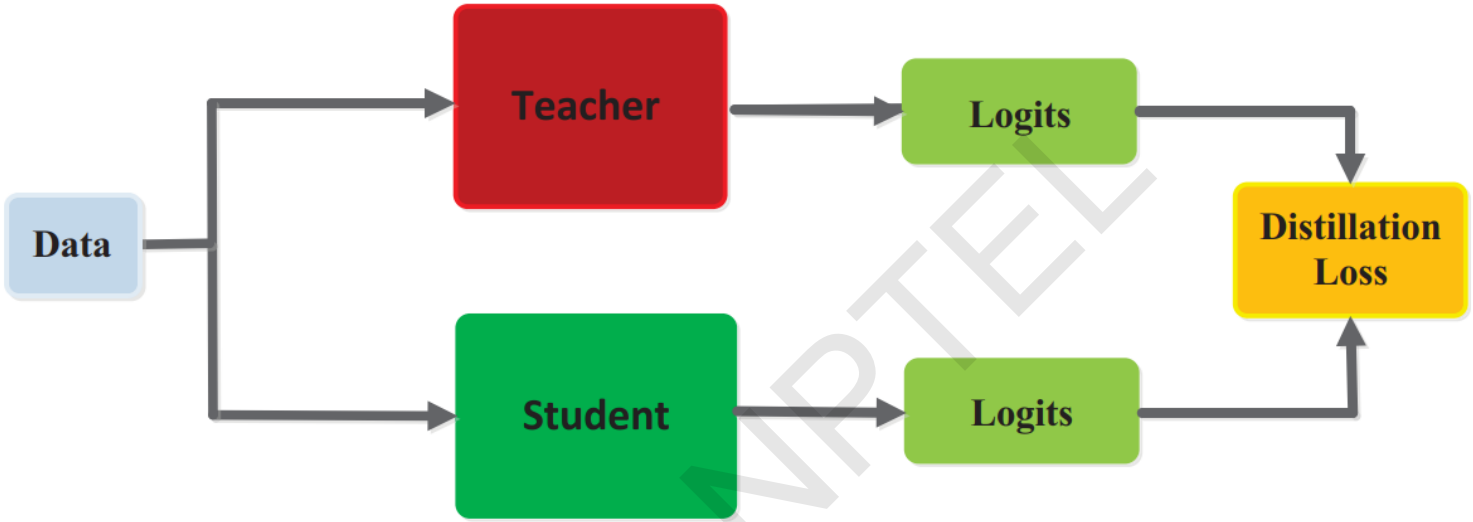


- ✓ response-based
- ✓ feature-based
- ✓ relation-based

Knowledge Types Response-Based Knowledge

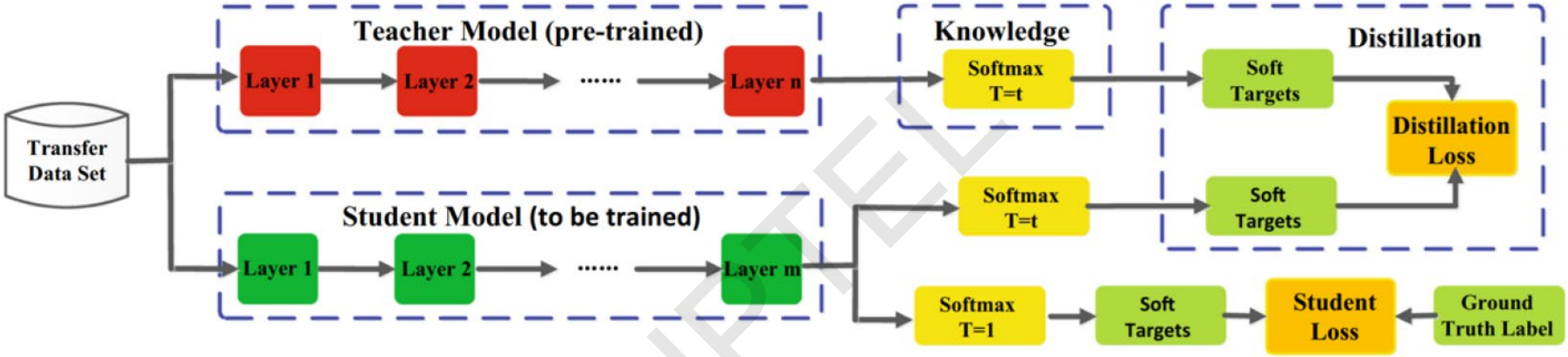


Knowledge Types Response-Based Knowledge



Why not use "**hard-targets**" ?

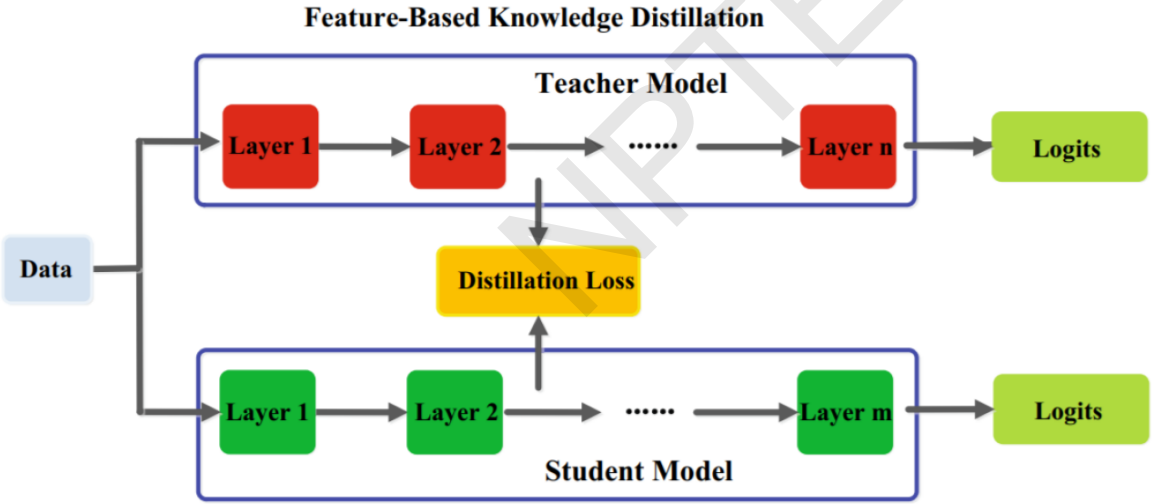
Knowledge Modeling



Knowledge Modeling

Feature-Based Knowledge

The output of intermediate layers, *i.e.*, feature maps, can also be used as the knowledge to supervise the training of the student model, which forged feature-based knowledge distillation. It is the improvement of response-based knowledge distillation.



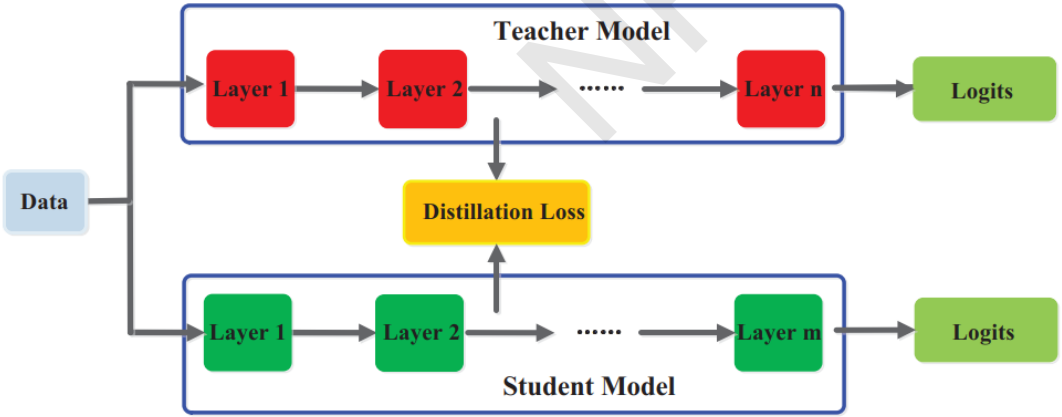
Knowledge Modeling

Formulation

Generally, the distillation loss for feature-based knowledge transfer can be formulated as

$$L_{FeaD}(f_t(x), f_s(x)) = \mathcal{L}_F(\phi_t(f_t(x)), \phi_s(f_s(x)))$$

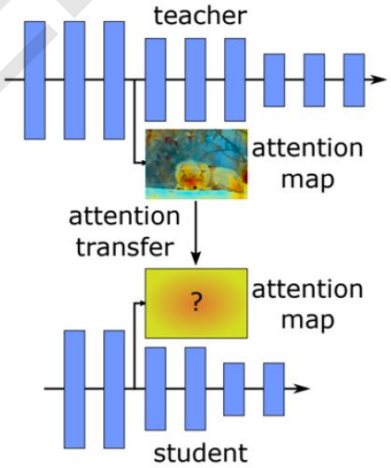
where $f_t(x)$ and $f_s(x)$ are the feature maps of the intermediate layers of teacher and student models, respectively. The transformation functions, $\phi_t(f_t(x)$ and $\phi_s(f_s(x))$, are usually applied when the feature maps of teacher and student models are not in the same shape. $\mathcal{L}_F(\cdot)$ indicates the similarity function used to match the feature maps of teacher and student models. $\mathcal{L}_F(\cdot)$ can be $\mathcal{L}_2(\cdot)$, $\mathcal{L}_1(\cdot)$, $\mathcal{L}_{CE}(\cdot)$ and *etc*.



Knowledge Modeling

Example

[6] proposed a feature-based knowledge distillation using attention mechanism. Specifically, the student network learns attention information from teacher network.



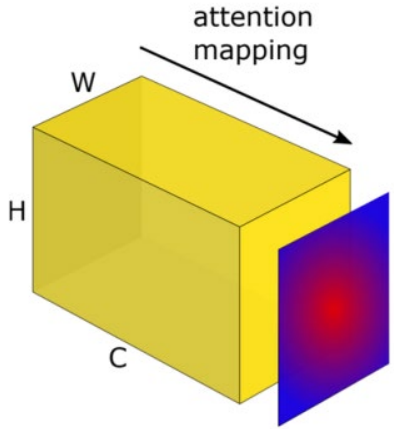
[6] Sergey Zagoruyko and Nikos Komodakis (2016). "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer". In: *arXiv preprint arXiv:1612.03928*.

Knowledge Modeling

Example

Considering a CNN layer and its corresponding activation tensor $A \in R^{C \times H \times W}$, which consists of C feature planes with spatial dimensions $H \times W$. An activation-based mapping function $\mathcal{F}$ (w.r.t. that layer) takes as input the above 3D tensor A and outputs a **spatial attention map**, i.e., a flattened 2D tensor defined over the spatial dimensions, or

$$\mathcal{F} : R^{C \times H \times W} \longrightarrow R^{H \times W}$$



Example

Specifically, in this work we consider the following activation-based spatial attention maps:

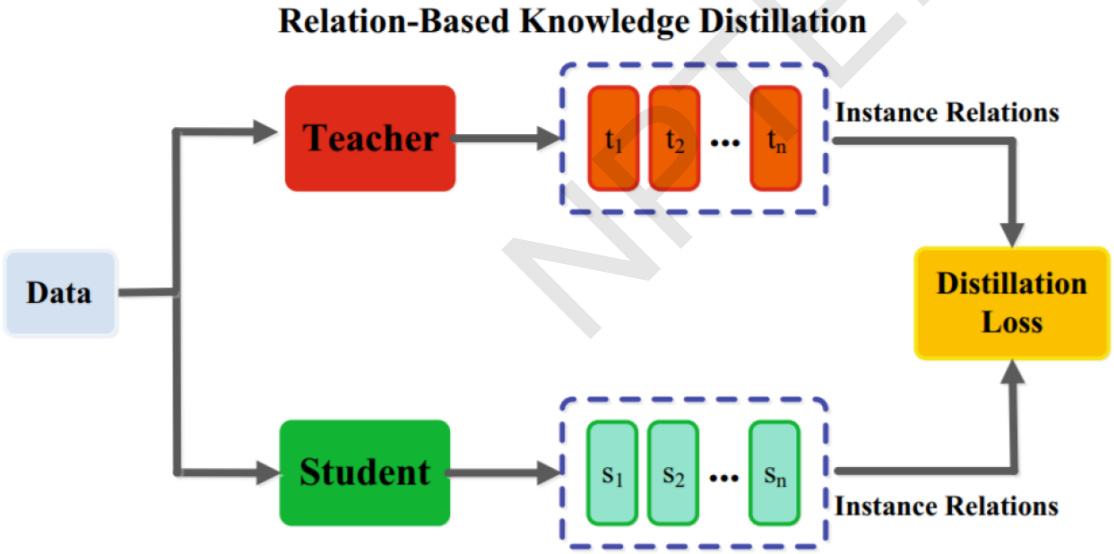
$$\mathcal{F}(A) = \sum_{i=1}^{C} |A_i|$$

Then we can define $\mathcal{I}$ as the indices of all teacher-student activation layer pairs for which we want to transfer attention maps. Also, we define $Q_S^j = \mathcal{F}(A_S^j)$ and $Q_T^j = \mathcal{F}(A_T^j)$ as the j-th ($j \in \mathcal{I}$) pair of student and teacher attention maps.

Knowledge Modeling

Relation-Based Knowledge

Both response-based and feature-based knowledge use the outputs of specific layers in the teacher model. Relationbased knowledge further explores the relationships between different layers or data samples.



Knowledge Modeling

Formulation

In general, the distillation loss of relation-based knowledge based on the relations of feature maps can be formulated as

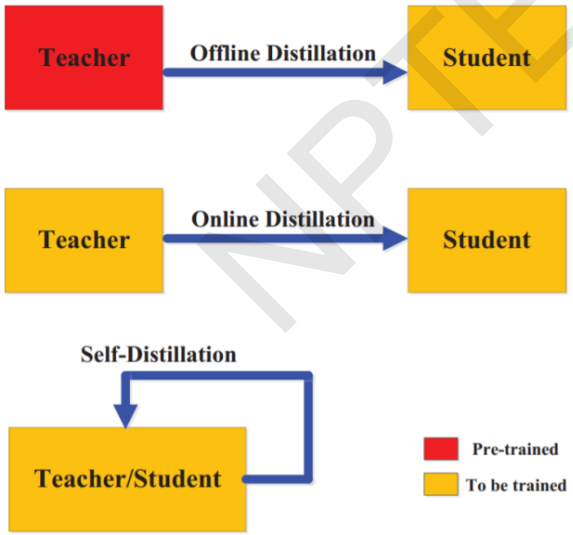
$$L_{RelD}(f_t, f_s) = \mathcal{L}_R(\Phi_t(\hat{f}_t, \check{f}_t), \Phi_s(\hat{f}_s, \check{f}_s))$$

where f_t and f_t are the feature maps of teacher and student models, respectively. Pairs of feature maps are chosen from the teacher model, $\hat{f}_t$ and $\check{f}_t$, and from the student model, $\hat{f}_s$ and $\check{f}_s$. $\Phi_t(\cdot)$ and $\Phi_s(\cdot)$ are the similarity functions for pairs of feature maps from the teacher and student models. $\mathcal{L}_R(\cdot)$ indicates the correlation function between the teacher and student feature maps.

Distillation Methods

Offline Distillation

Most of previous knowledge distillation methods work offline. In offline knowledge distillation, the knowledge is transferred from a pre-trained teacher model into a student model.

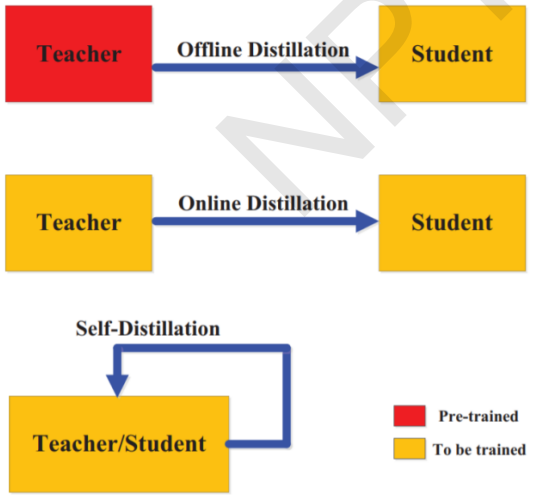


Distillation Methods

Offline Distillation

Therefore, the whole training process has two stages:

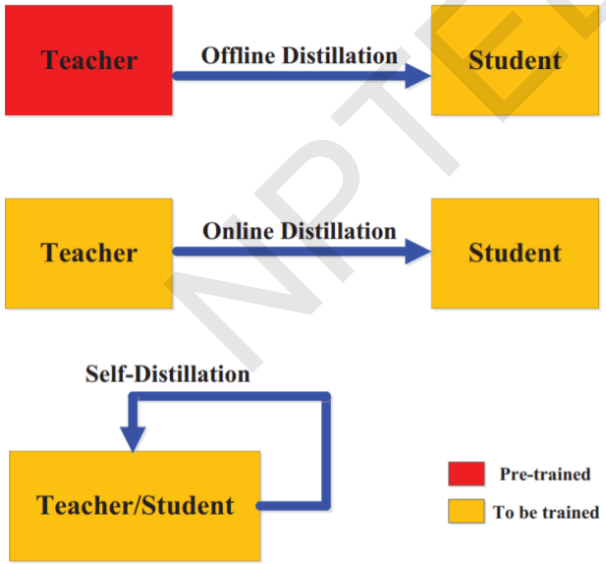
- The large teacher model is first trained on a set of training samples before distillation.
- The teacher model is used to extract the knowledge in the forms of logits or the intermediate features, which are then used to guide the training of the student model during distillation.



Distillation Methods

Online Distillation

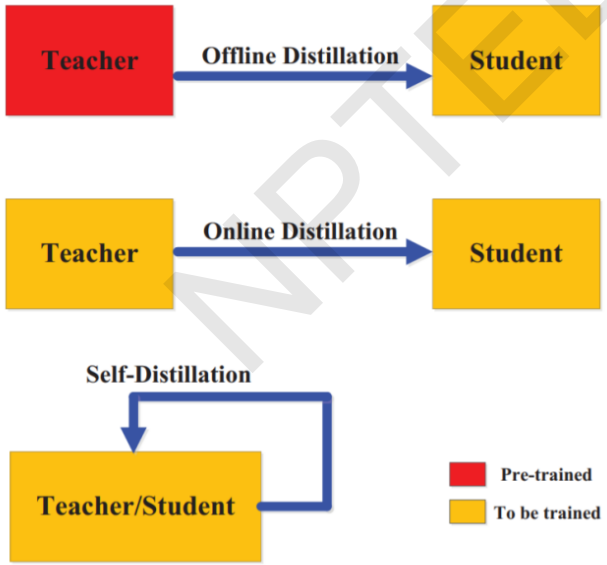
In online distillation, both the teacher model and the student model are updated and the whole knowledge distillation framework is end-to-end trainable.



Distillation Methods

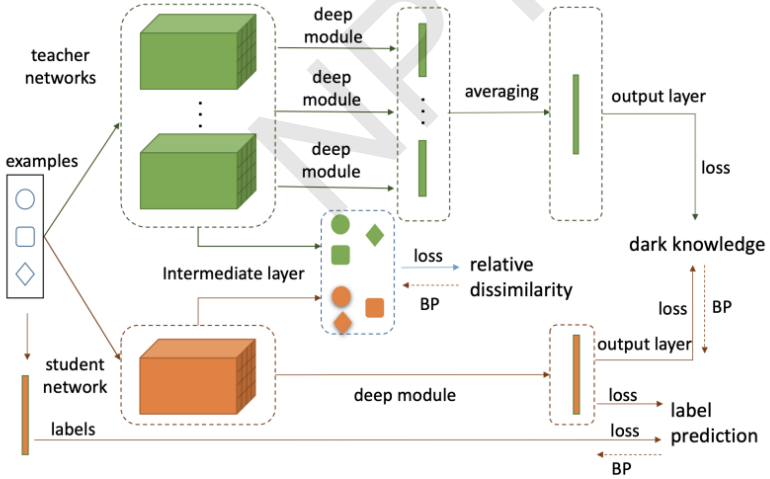
Self Distillation

In self-distillation, the same networks are used for the teacher and the student models. This can be regarded as a special case of online distillation.



Distillation Methods

Combine KD and Ensemble learning. A graphical diagram for the proposed method to train a new thin deep student network by incorporating multiple comparable teacher networks. The method consists of three losses, including label prediction loss, dark knowledge loss and the relative similarity loss. The incorporation of multiple teacher networks exists in two places. One is in the output layers via averaging the softened output targets; the other lies in the intermediate layer by determining the best triplet ordering relationships.



Distillation Methods

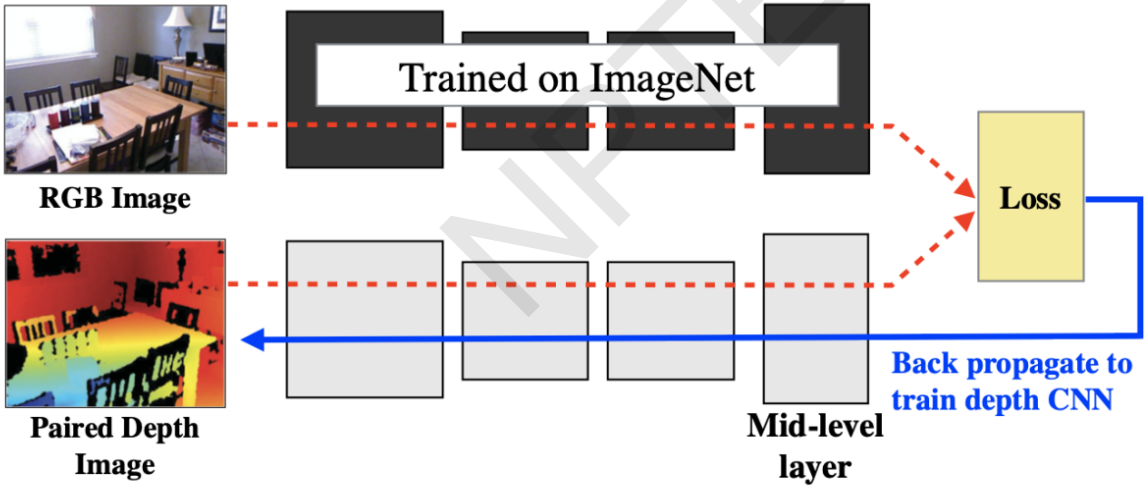
Here, the label prediction loss is a simple softmax cross entropy loss. Relative similarity loss is triplet loss, dark knowledge loss is ensemble KD loss. Given m teacher networks $\mathcal{N}_{T_1}, \mathcal{N}_{T_2}, \cdot, \mathcal{N}_{T_m}$ and one student network $\mathcal{N}_S$, we can have.

$$\mathcal{L}_{final} = \sum [\mathcal{H}(\boldsymbol{y}_i, \mathcal{N}(\boldsymbol{x}_i)) + \alpha \mathcal{H}(\frac{1}{m} \sum_{t=1}^{m} \mathcal{N}_{T_t}^{\tau}(\boldsymbol{x}_i), \mathcal{N}_S^{\tau}(\boldsymbol{x}_i))] + \beta \mathcal{L}_{RD}(w_s; \boldsymbol{x}_i, \boldsymbol{x}_i^+, \boldsymbol{x}_i^-), \quad (9)$$

where $\mathcal{H}$ means the entropy function, w_s indicates the parameters of feature extractor, $\boldsymbol{x}_i, \boldsymbol{x}_i^+, \boldsymbol{x}_i^-$ means the triplet pairs.

Distillation Methods

Combine KD and Cross Modal Learning. Architecture: We train a CNN model for a new image modality (like depth images), by teaching the network to reproduce the mid-level semantic repre- sentations learned from a well labeled image modality (such as RGB images) for modalities for which there are paired images.



Distillation Methods

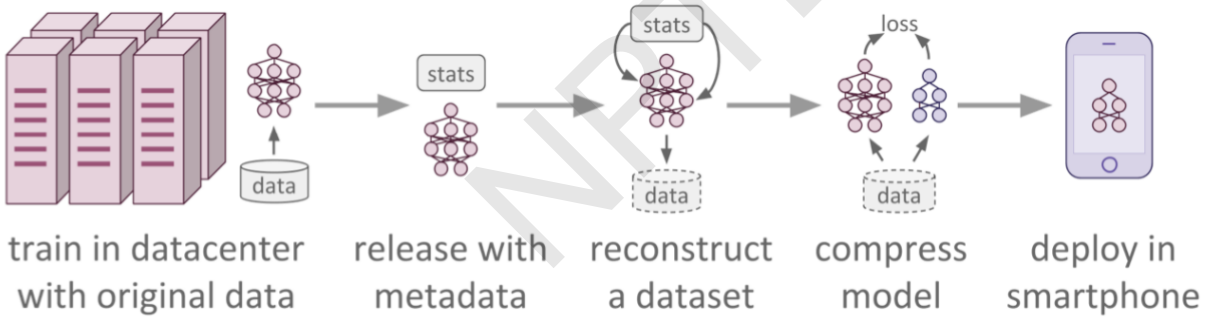
The proposed scheme for learning rich representations for images of modality $\mathcal{M}_d$. t indicates the functions that maps the $\left(\psi^L_{\mathcal{M}_d}(I_d)\right)$ to the same dimension with $\phi^{i^*}_{\mathcal{M}_s,D_s}(I_s)$. for some chosen and fixed layer $i^* \in [1 \ldots K]$, we measure the similarity between the representations using an appropriate loss function f (for example, euclidean loss).

$$\mathcal{L}_{final} = \sum_{(I_s,I_d)\in U_{s,d}} f\left(t\left(\psi^L_{\mathcal{M}_d}(I_d)\right),\ \phi^{i^*}_{\mathcal{M}_s,D_s}(I_s)\right) \tag{10}$$

[12]Saurabh Gupta, Judy Hoffman, and Jitendra Malik (2016). "Cross modal distillation for supervision transfer". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2827–2836.

Distillation Methods

Combine KD and Data Free Compression. The proposed model compression pipeline: a model is trained in a datacenter and released along with some metadata. Then, another entity uses that metadata to reconstruct a dataset, which is then used to compress the model with Knowledge Distillation. Finally, the model is deployed in a smartphone.

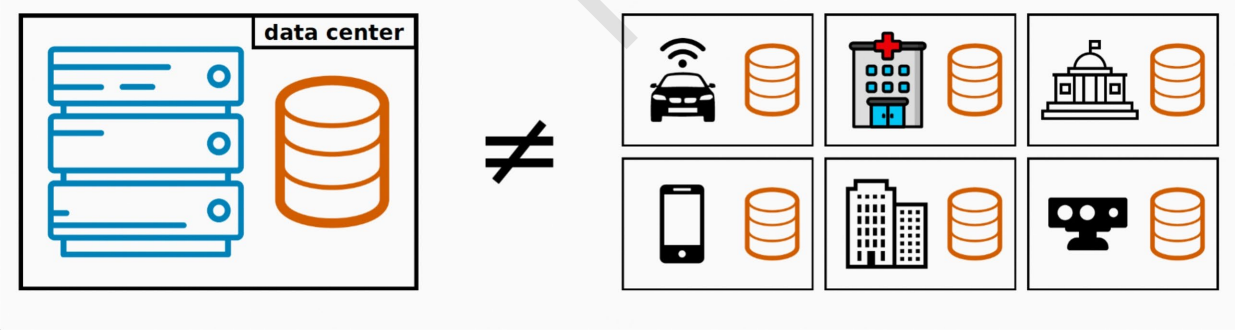


[13] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner (2017). "Data-free knowledge distillation for deep neural networks". In: *arXiv preprint arXiv:1710.07535*.

Federated Learning

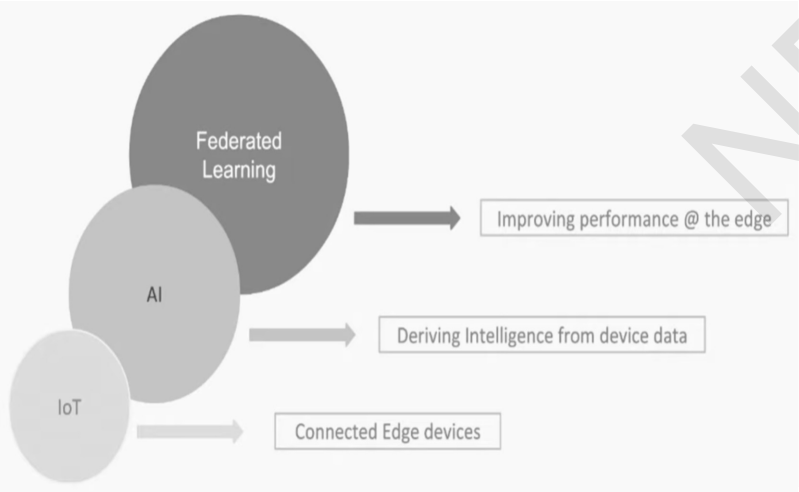
Federated Learning: Distributed ML

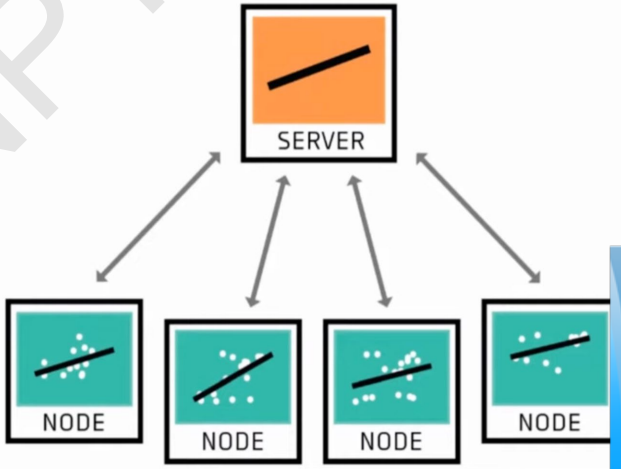
- 2016: the term FL is first coined by Google researchers
- We have already seen some real-world deployments by companies and researchers for large scale IOT devices
- Several open-source libraries are under development: PySyft, TensorFlow Federated, FATE, Flower, Substra...
- FL is highly multidisciplinary: it involves machine learning, numerical optimization, privacy & security, networks, systems, hardware...



Federated Learning: Decentralised data

- Federated Learning (FL) aims to collaboratively train a ML model while keeping the data decentralized
- Enabling devices to learn from each other (ML training is brought close
- A network of nodes and all nodes with their own central server but instead of sharing data with the central server, we share model we don't send data from node to server instead send our model to server





Gradient Descent Procedure

The procedure starts off with initial values for the coefficient or coefficients for the function. These could be 0.0 or a small random value.

coefficient = 0.0

The cost of the coefficients is evaluated by plugging them into the function and calculating the cost.

cost = f(coefficient) or cost = evaluate(f(coefficient))

We need to know the slope so that we know the direction (sign) to move the coefficient values in order to get a lower cost on the next iteration.

delta = derivative(cost)

we can now update the coefficient values.

A learning rate parameter (alpha) must be specified that controls how much the coefficients can change on each update.

coefficient = coefficient – (alpha * delta)

This process is repeated until the cost of the coefficients (cost) is 0.0 or close to 0 It does require you to know the gradient of your cost function or the function you are optimizing

Gradient Descent Algorithm

Gradient Descent *algorithm:*

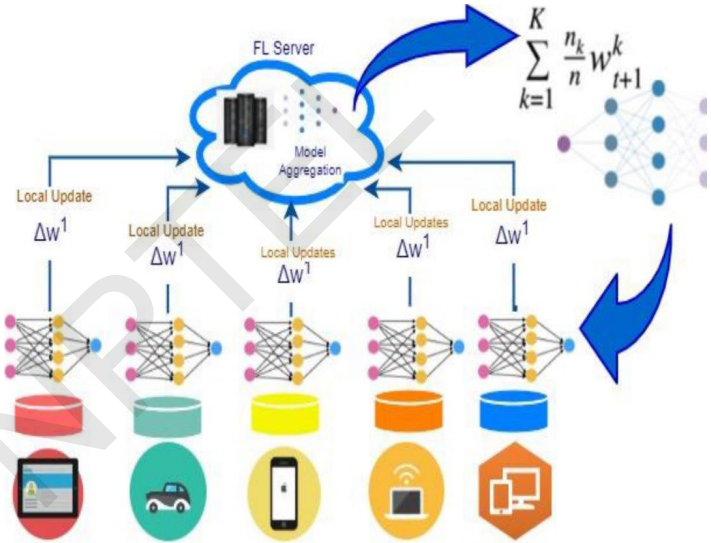
$$\vartheta = \vartheta - \alpha \cdot \nabla J(\vartheta)$$

Advantages:

- Easy computation
- Easy to implement
- Easy to understand

Edge Computing ML: FL

- moves the processing over the edge nodes so that the clients' data can be maintained.
- trains an ML algorithm with the local data samples distributed over multiple edge devices or servers without any exchange of data.
- Federated learning distributes deep learning by eliminating the necessity of pooling the data into a single place.
- model is trained at different sites in numerous iterations



Edge Computing ML: FL

Finding the function: model training

- Given a training dataset containing n input-output pairs $(x_i, y_i), i \in [1, n]$, the goal of deep learning model training is to find a set of parameters w, such that the average of $p(y_i)$ is maximized given x_i.

- That is,

$$maximize \quad \frac{1}{n}\sum_{i=1}^{n} p(y_i|x_i, w)$$

Which is equivalent to

$$minimize \quad \frac{1}{n}\sum_{i=1}^{n} -\log(p(y_i|x_i, w))$$

A basic component for loss function $l(x_i, y_i, w)$ given sample (x_i, y_i):

Let $f_i(w) = l(x_i, y_i, w)$ denote the loss function.

Deep Learning model training

- Given one input sample pair (x_0, y_0), the goal of deep learning model training is to find a set of parameters w, to maximize the probability of outputting y_0 given x_0.

For a training dataset containing n samples $(x_i, y_i), 1 \leq i \leq n$, the training objective is:

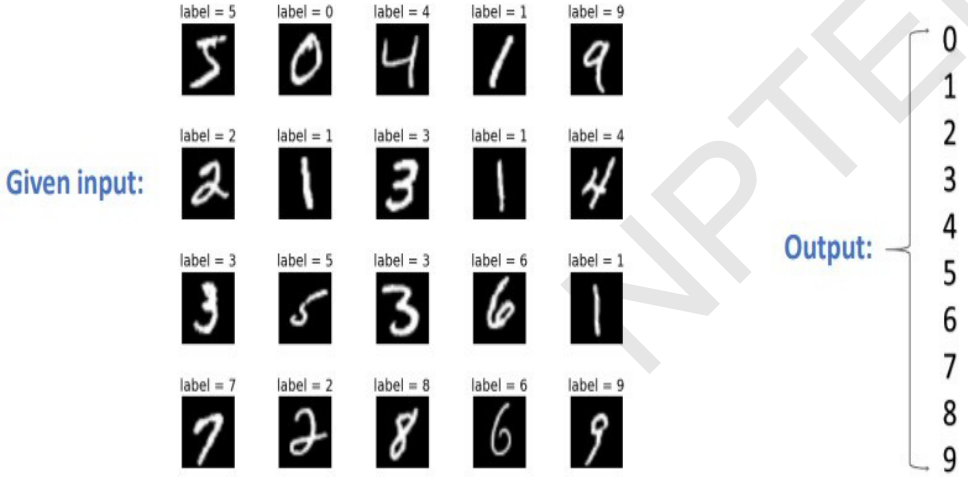
$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where } f(w) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

$f_i(w) = l(x_i, y_i, w)$ is the loss of the prediction on example (x_i, y_i)

Edge Computing ML: FL

Finding the function: model training

- Given one input sample pair (x_0, y_0), the goal of deep learning model training is to find a set of parameters w, to maximize the probability of outputting y_0 given x_0.



Federated learning – *FederatedAveraging (FedAvg)*

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

  $m \leftarrow \max(C \cdot K, 1)$

  $S_t \leftarrow$ (random set of m clients)

  for each client $k \in S_t$ **in parallel do**

    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*

  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

  for each local epoch i from 1 to E **do**

    for batch $b \in \mathcal{B}$ **do**

      $w \leftarrow w - \eta \nabla \ell(w; b)$

  return w to server

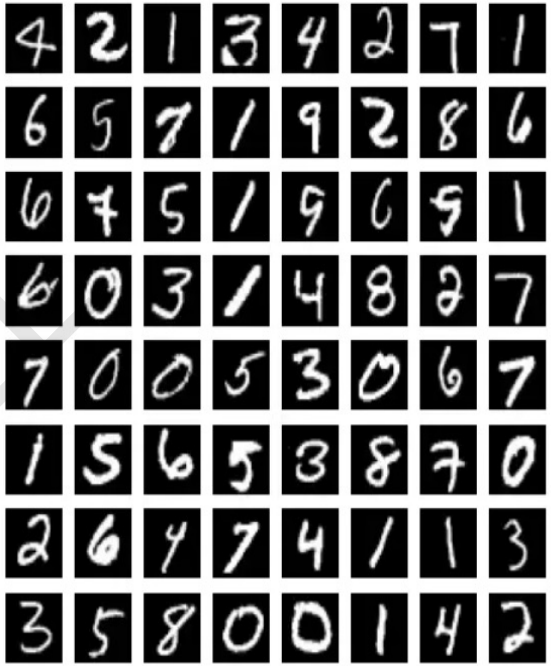
1. At first, a model is randomly initialized on the central server.
2. For each round *t:*
   - *i. A random set of clients are chosen;*
   - *ii. Each client performs local gradient descent steps;*
   - *iii. The server aggregates model parameters submitted by the clients.*

Example: FL with i.i.d.

Each client trains its model decentral - the model training process is carried out separately for each client.

Only learned model parameters are sent to a trusted center to combine and feed the aggregated main model.

Then the trusted center sent back the aggregated main model back to these clients, and this process is circulated.



Handwritten Digits from the MNIST datase

Apple personalizes Siri without hoovering up data

The tech giant is using privacy-preserving machine learning to improve its voice assistant while keeping your data on your phone.

It relies primarily on a technique called federated learning.

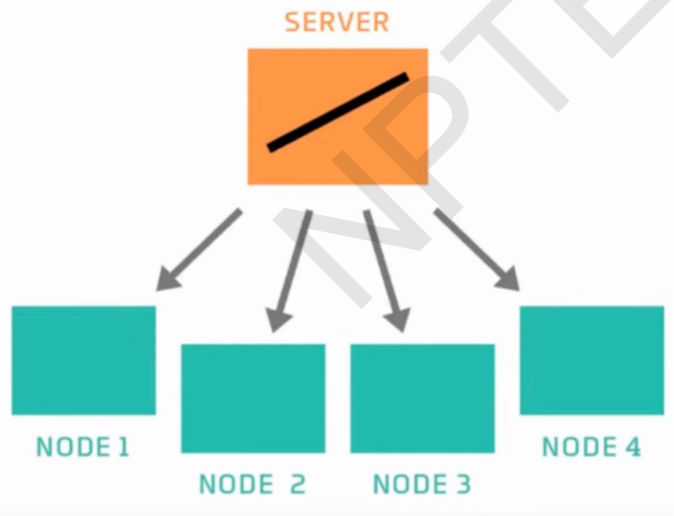
It allows Apple to train different copies of a speaker recognition model across all its users' devices, using only the audio data available locally.

It then sends just the updated models back to a central server to be combined into a master model.

In this way, raw audio of users' Siri requests never leaves their iPhones and iPads, but the assistant continuously gets better at identifying the right speaker. In addition to federated learning, Apple also uses something called differential privacy to add a further layer of protection. The technique injects a small amount of noise into any raw data before it is fed into a local machine-learning model. The additional step makes it exceedingly difficult for malicious actors to reverse-engineer the original audio files from the trained model.

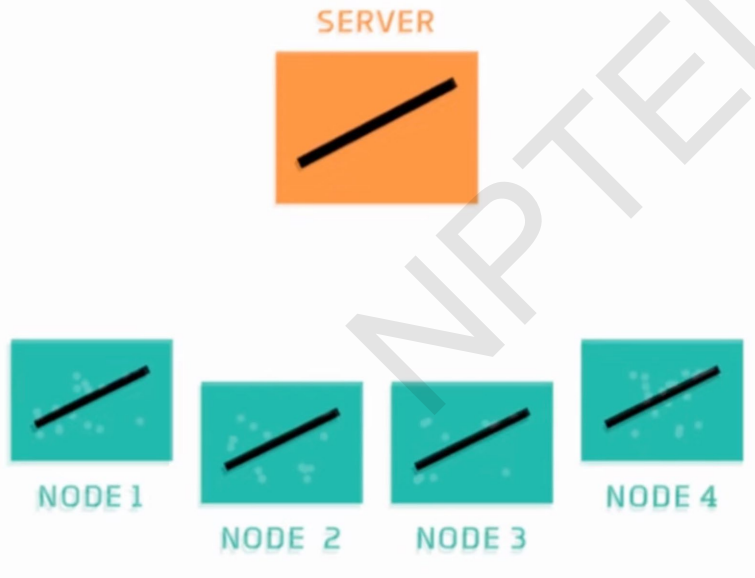
Federated Learning: Training

- There are connected devices let's say we have cluster of four devices from four of the devices and there is one central server that has an untrained model.
- We will send a copy of the model to each of the node.
- Each node would receive a copy of that model.



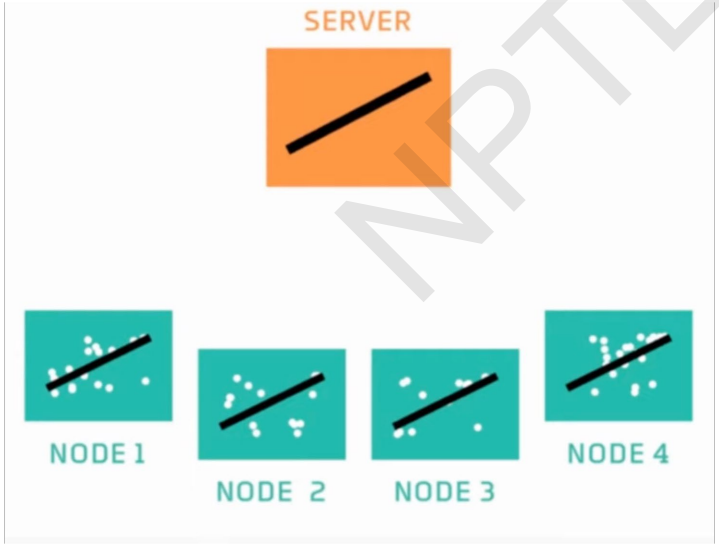
Federated Learning: Training

- Now all the nodes in the network has that untrained model that is received from the server.



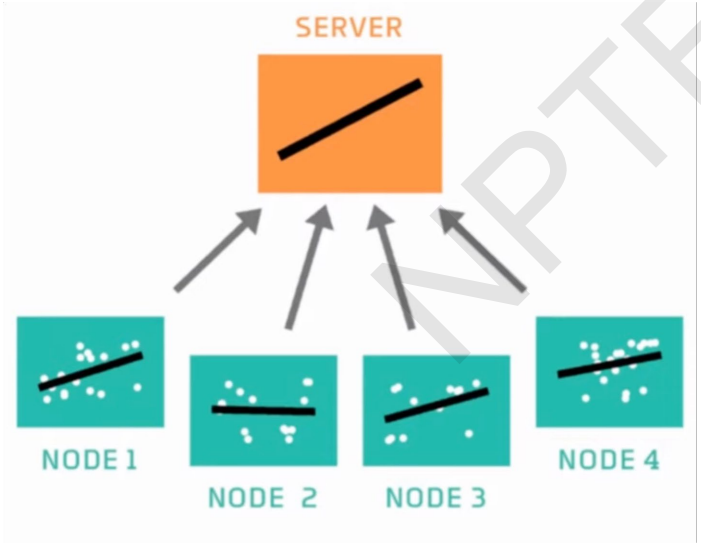
Federated Learning: Training

- In the next step, we are taking data from each node by taking data it doesn't mean that we are sharing data.
- Every node has its own data based on which it is going to train a model.



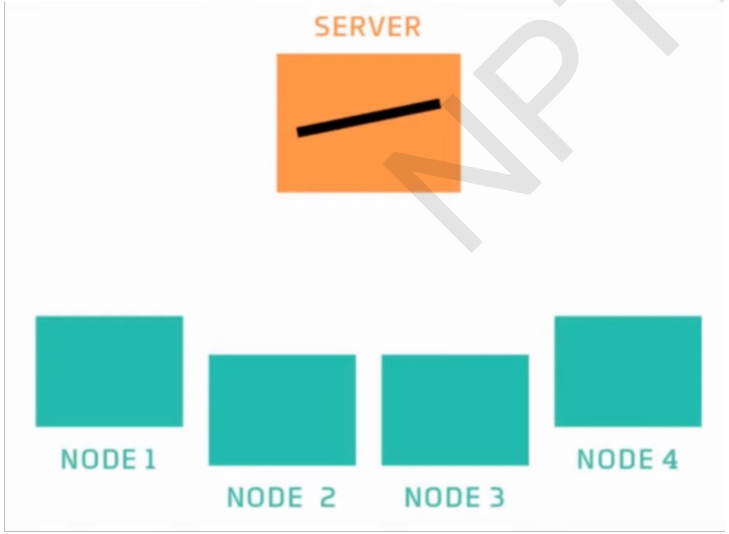
Federated Learning: Training

- Each node is training the model to fit the data that they have and it will train the model accordingly to its data.



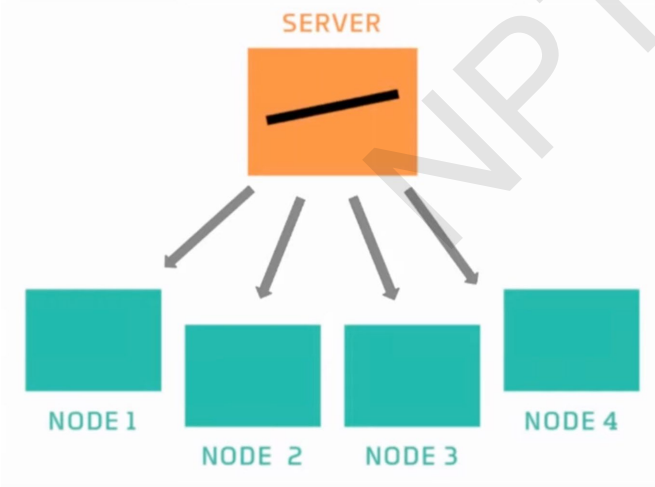
Federated Learning: Training

- Now the server would combine all these model received from each node by taking an average or it will aggregate all the models received from the nodes.
- Then the server will train that a central model, this model which is now trained by aggregating the models from each node. It captures the pattern in the training data on all the nodes it is an aggregated one



Federated Learning: Training

- Once the model is aggregated, the server will send the copy of the updated model back to the nodes.
- Everything is being achieved at the edge so no data sharing is done which means there is privacy preservation and also very less communication overhead.



Federated Learning: Challenges

Systems heterogeneity

- Size of data
- Computational power
- Network stability
- Local learner
- Learning rate

Expensive Communication

- Communication in the network can be slower than local computation by many order of magnitude.

Statistical Heterogeneity:

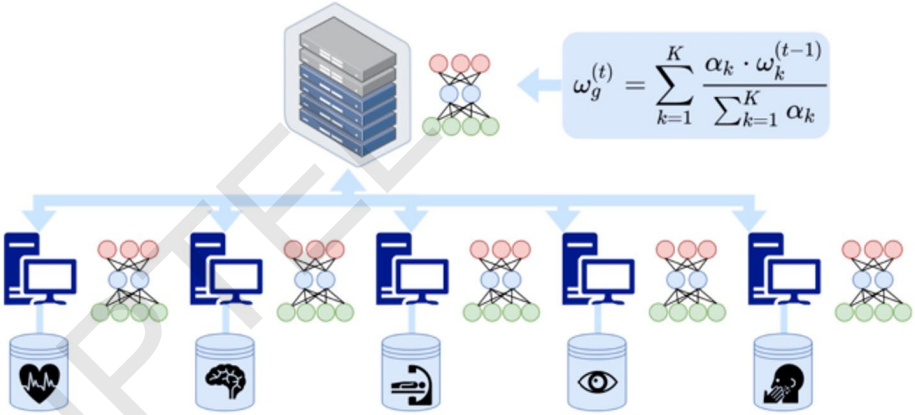


Fig. 1: Federated learning with non-iid data - The data has different distributions among clients.

Federated Learning: Challenges

Dealing with Non-I.I.D. data i.i.d (independent and identical distributed)

- Learning from non-i.i.d. data is difficult/slow because each IOT device needs the model to go in a particular direction
- If data distributions are very different, learning a single model which performs well for all IOT devices may require a very large number of parameters
- Another direction to deal with non-i.i.d. data is thus to lift the requirement that the learned model should be the same for all IOT devices (“one size fits all”)
- Instead, we can allow each IOT k to learn a (potentially simpler) personalized model θ_k but design the objective so as to enforce some kind of collaboration
- When local datasets are non-i.i.d., FedAvg suffers from client drift
- To avoid this drift, one must use fewer local updates and/or smaller learning rates, which hurts convergence

Federated Learning: Challenges

Preserving Privacy

- ML models are susceptible to various attacks on data privacy
- Membership inference attacks try to infer the presence of a known individual in the training set, e.g., by exploiting the confidence in model predictions
- Reconstruction attacks try to infer some of the points used to train the model, e.g., by differencing attacks
- Federated Learning offers an additional attack surface because the server and/or other clients observe model updates (not only the final model)

Key differences with Distributed Learning

Data distribution

- In distributed learning, data is centrally stored (e.g., in a data center)
  - The main goal is just to train faster
  - We control how data is distributed across workers: usually, it is distributed uniformly at random across workers
- In FL, data is naturally distributed and generated locally
  - Data is not independent and identically distributed (non-i.i.d.), and it is imbalanced

Additional challenges that arise in FL

- Enforcing privacy constraints
- Dealing with the possibly limited reliability/availability of participants
- Achieving robustness against malicious parties

Federated Learning: Concerns

When to apply Federated Learning

- Data privacy needed
- Bandwidth and power consumptions are concerns
- High cost of data transfer

When NOT to apply Federated Learning

- When more data won't improve your model (construct a learning cure)
- When additional data is uncorrelated
- Performance is already at ceiling

Federated Learning: Applications

- **Predictive maintenance/industrial IOT**
- **Smartphones**
- **Healthcare (wearables, drug discovery, prognostics, etc.)**
- **Enterprise/corporate IT (chat, issue trackers, emails, etc.)**

Conclusion

In this lecture we discussed about

- What is Knowledge distillation
- Types of knowledge distillation
- Methods of knowledge distillation
- Understanding of Federated Learning
- Different issues with federated learning

Thank You!

References

- Deep Learning, Ian Goodfellow and Yoshua Bengio and Aaron Courville, MIT Press, http://www.deeplearningbook.org
- Knowledge Distillation: A Survey Jianping Gou · Baosheng Yu · Stephen J. Maybank · Dacheng Tao arXiv:2006.05525v7