

Meta-Learning based Networking Architecture

Choong Seon Hong

Department of Computer Science and Engineering
Kyung Hee University, Republic of Korea.

- Introduction
- Types of Meta-learning
- Use Case : Meta-Learning-Based Deep Learning Model Deployment Scheme for Edge Caching
- Use Case : Edge Computing Resources Reservation in Vehicular Networks: A Meta-Learning Approach
- Conclusion

Introduction

- History of Meta-Learning
- Meta-Learning in AI
- Teachable Machine 2.0: Making AI easier for everyone by Google
- Normal Deep Learning Vs Meta-Learning
- What is meta-learning trying to do?
- Meta learning and few-shot
- Advantages of Meta Learning
- Application Example

Meta-Learning is originally a concept of **cognitive psychology**

- Originally described by **Donald B. Maudsley (1979)**
 - "the process by which learners become aware of and increasingly in control of habits of perception, inquiry, learning, and growth that they have internalized".[1]
 - Maudsley sets the conceptual basis of his theory as synthesized under headings of assumptions, structures, change process, and facilitation.
- The idea of meta learning was later used by **John Biggs (1985)** to describe the state of "being aware of and taking control of one's own learning".[2]

Meta learning can be defined as an *awareness and understanding of the phenomenon of learning itself as opposed to subject knowledge.*

https://en.wikipedia.org/wiki/Meta_learning

[1] Maudsley, D. B. (1979). A Theory of Meta-Learning and Principles of Facilitation: An Organismic Perspective. University of Toronto, 1979. (40, 8,4354-4355-A)

[2] Biggs, J. B. (1985). The role of meta-learning in study process. British Journal of Educational Psychology, 55, 185–212.

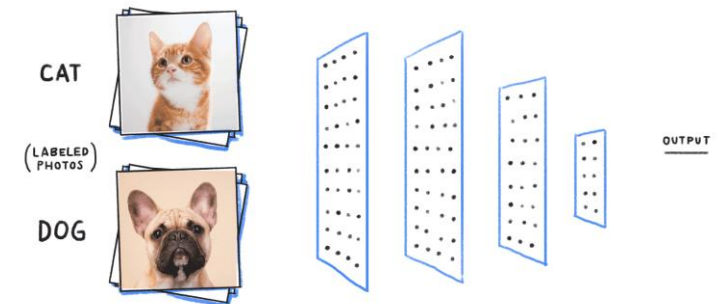


The basic concept of "**learning how to learn**" is extended to AI systems for the machine learning component of meta-learning.

- Meta-learning **bridges the knowledge transfer gap** among the learning models.
- Meta-learning models and techniques help the **AI not only to learn fast** but also how to **generalize learning methods** and gain **new skills faster**.
- Meta learning is an exhilarating research domain in the field of AI right now.
- With plenty of research papers and advancements, **meta learning is clearly making a major breakthrough in AI.**

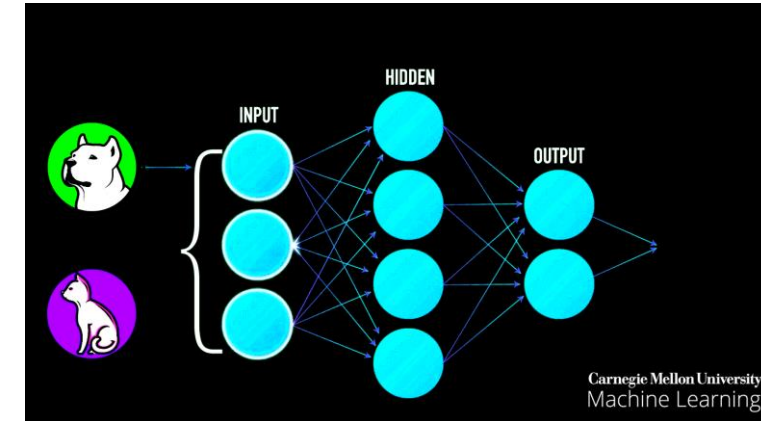


Learning How to Learn





- Deep learning has progressed rapidly in recent years with great algorithms such as **generative adversarial networks** and **capsule networks**.
- But **the problem with deep neural networks** is that we need to have a **large training set to train our model** and it will fail abruptly when we have very few data points.



For example: We trained a deep learning model to perform **task A**.

When we have **a new task, B**, that is **closely related to A**, we **can't use the same model**.

We need to **train the model from scratch for task B**.

So, for each task, we need to **train the model from scratch** although they might be **related**.

- Deep learning has progressed rapidly in recent years with great algorithms such as generative adversarial networks and capsule networks.
- But the problem with deep neural networks is that we need to have a

How can we solve the aforementioned problems?

How do we humans learn?

When we have a new task, by that is closely related to A, we can use the same model.

- We need to train the model from scratch for task B.
- So, for each task, we need to train the model from scratch although they might be related.

What is meta-learning trying to do?

- **Children** who have seen **snakes and dogs** just a **few times** can **differentiate them quickly**.
- We can apply the **same principles in machine learning area**.
- So, we can design **machine learning models** that will enable the machine to **learn a skill quickly and with few training examples**.



A **good meta-learning model will adapt or generalize well to new tasks and new environments** that were never experienced during the training period.

- Few-shot learning is a kind of meta learning.
- Learning from **fewer data points (i.e., images)** is called **few-shot learning**.

Support Set for the training

Armadillo



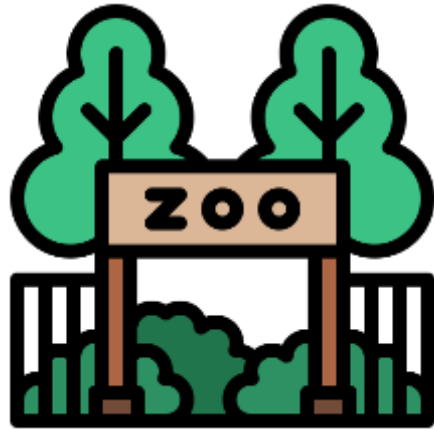
Pangolin



Armadillo or Pangolin ?

Query for testing





Normal Learning

Answer : Otter

Meta Learning

Give him the cards to compare

Fox



Squirrel



Rabbit



Hamster



Otter



Beaver



Query

What is this ?



Support Set

Fox



Squirrel



Rabbit



Hamster



Otter



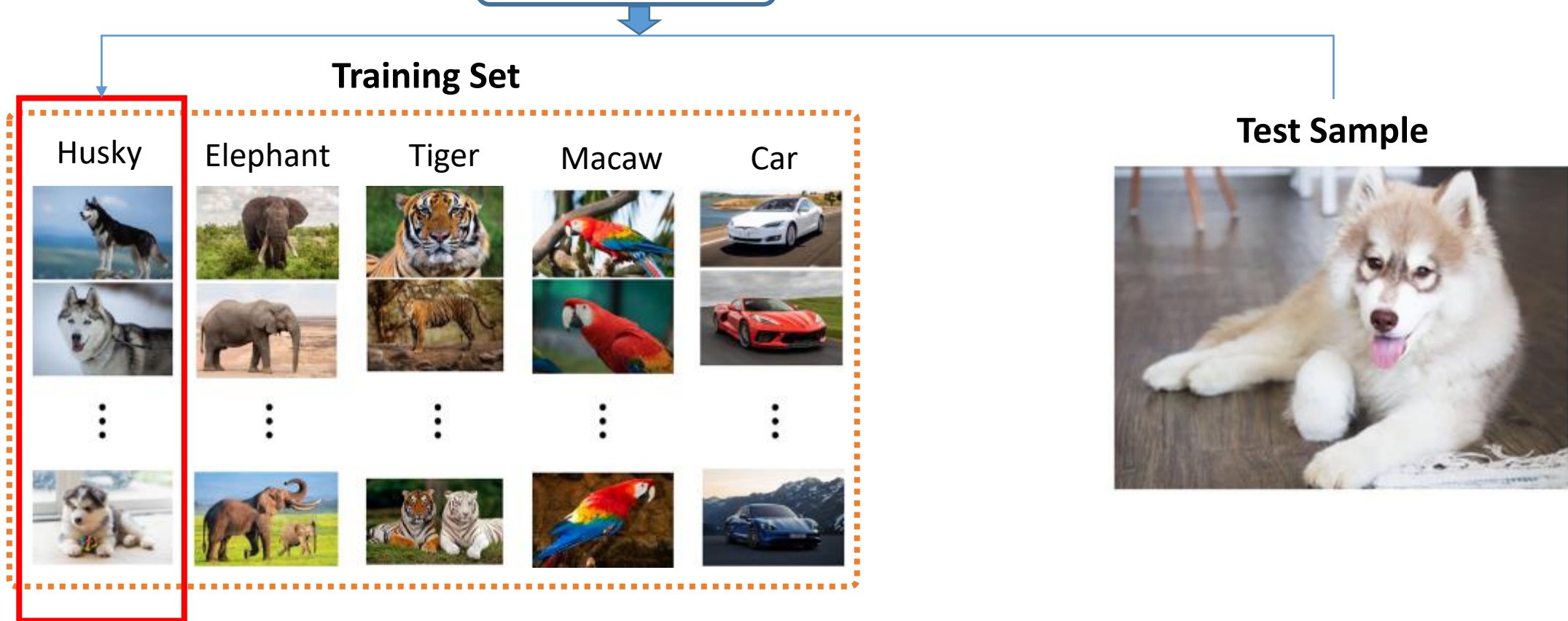
Beaver



You can answer which animal is the same one.

Supervised Learning VS. Few-Shot Learning

- Traditional Supervised Learning:
 - Test samples are **never seen before**.
 - Test samples are from **known classes**.



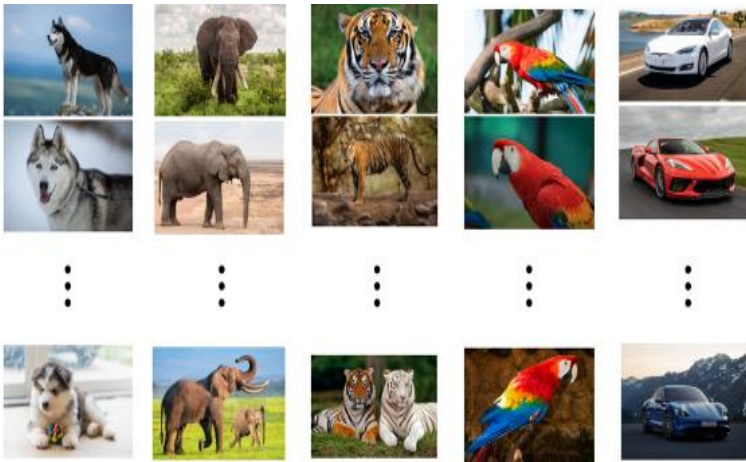
Supervised Learning VS. Few-Shot Learning

Test labels are used during training

Training Phase

Examples

Husky Elephant Tiger Macaw Car



Classifier

Labels

Husky
Elephant
Tiger
Macaw
Car

Predicting Phase



Classifier

0	Husky
1	Elephant
0	Tiger
0	Macaw
0	Car

Supervised Learning VS. **Few-Shot Learning**

- Few-Shot Learning:
 - **Query** samples are **never seen before**.
 - **Query** samples are from **unknown classes**.

Training Set



Query Sample



Training Set, Support Set, and Query

Support Set

Fox



Squirrel



Rabbit



Hamster



Otter



Beaver



Training Set

Husky



Elephant



Tiger



Macaw



Car



⋮

⋮

⋮

⋮

⋮



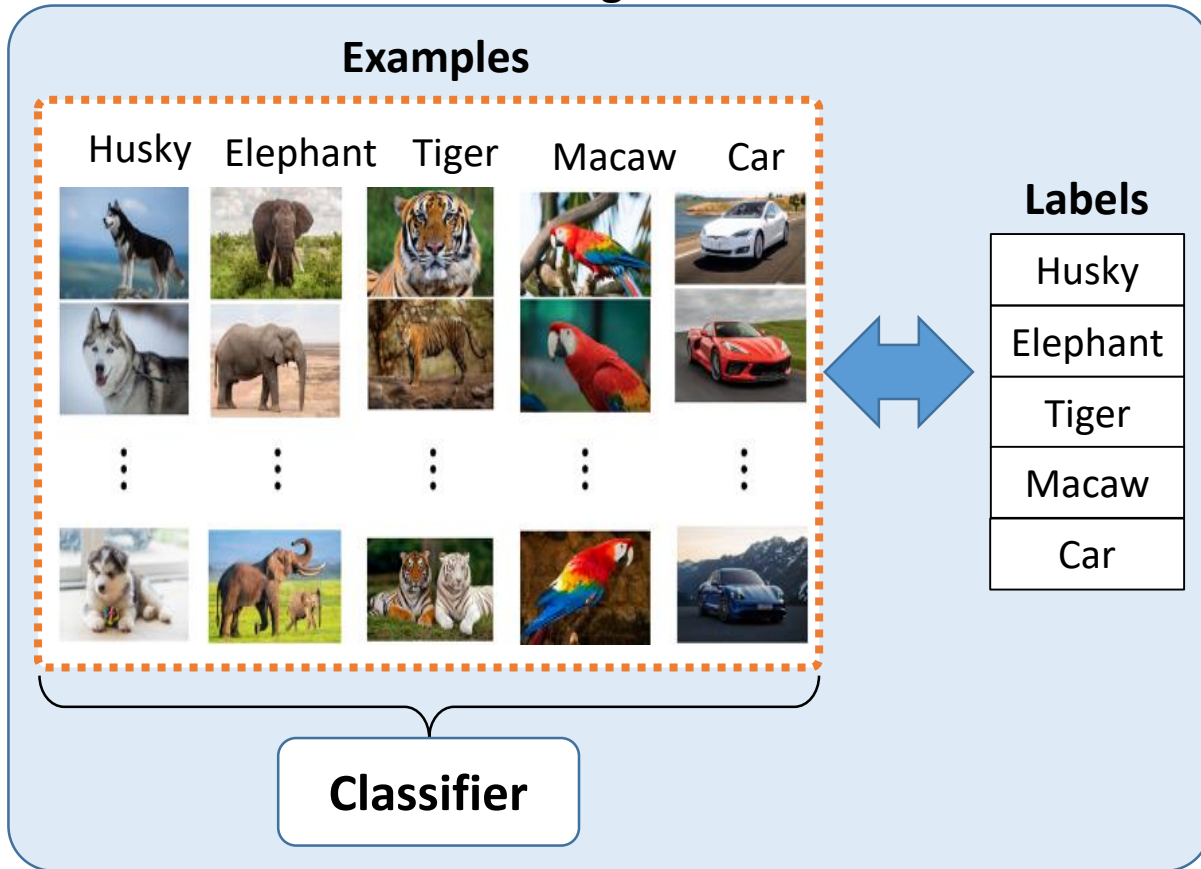
Query Sample



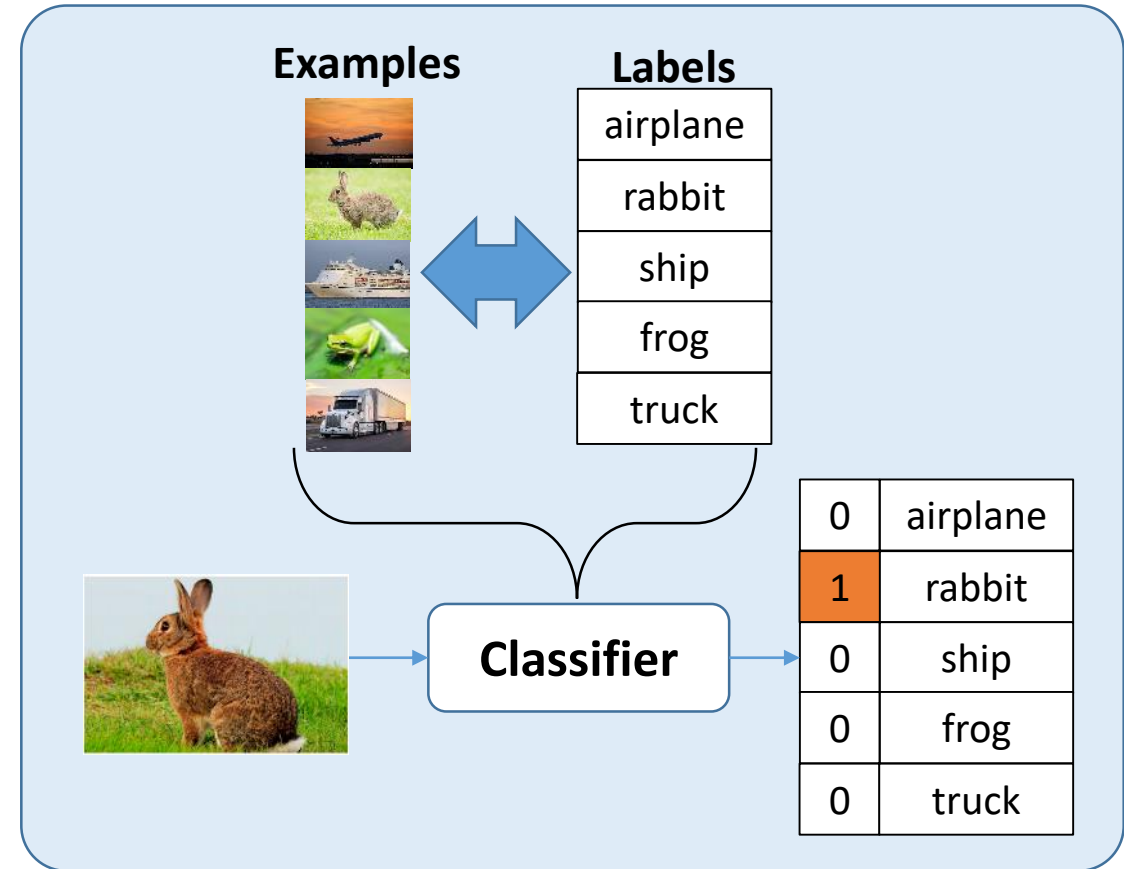
Supervised Learning VS. **Few-Shot Learning**

Test labels are not used during training. Disjoint label space.

Training Phase



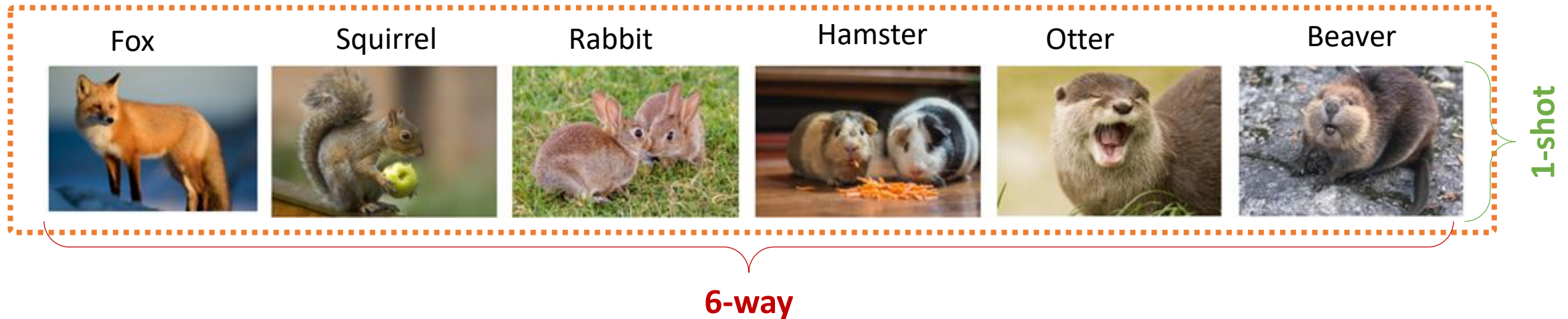
Predicting Phase



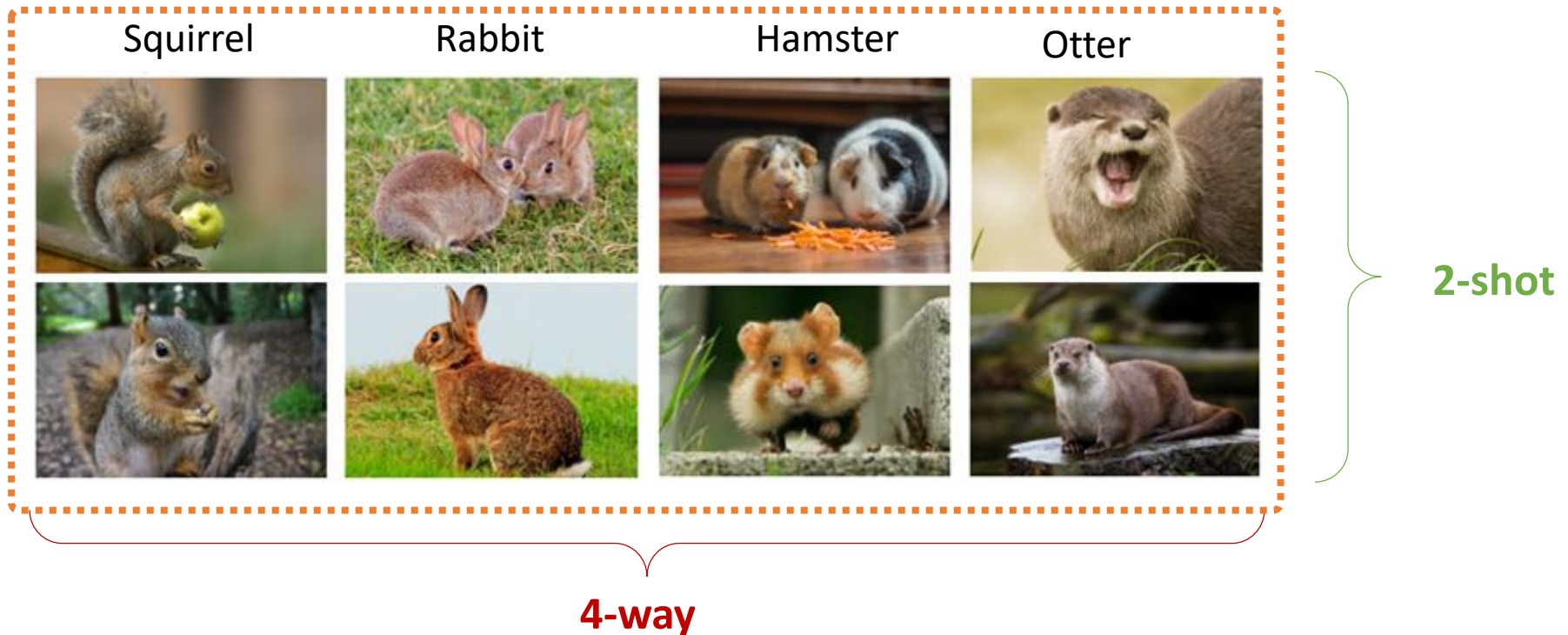
K-way n-shot Support Set

- K-way: the support set has k classes.
- n-shot: every class has n samples.

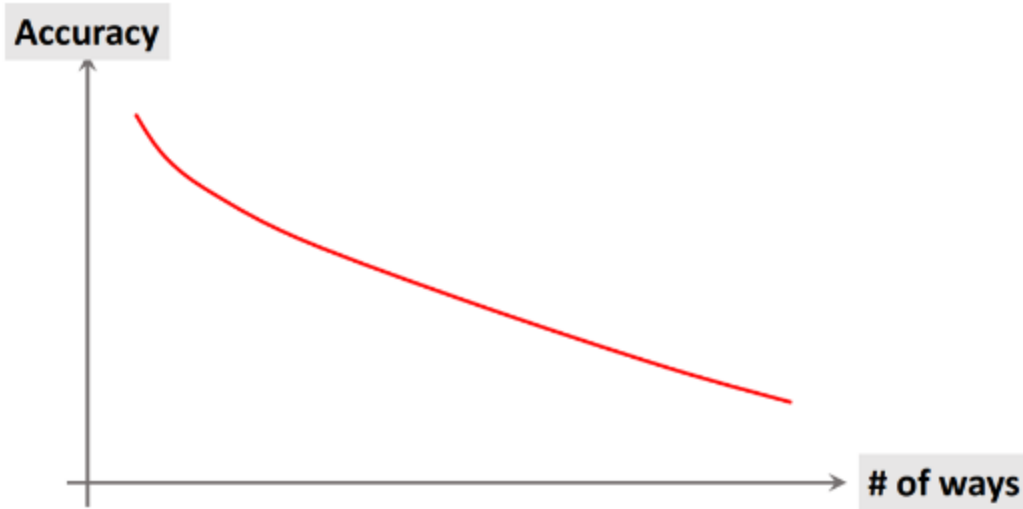
Support Set



Support Set



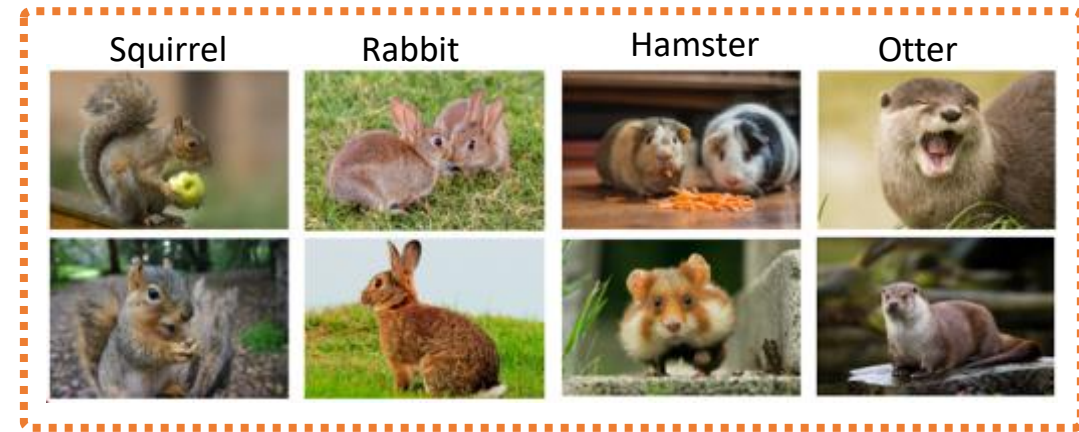
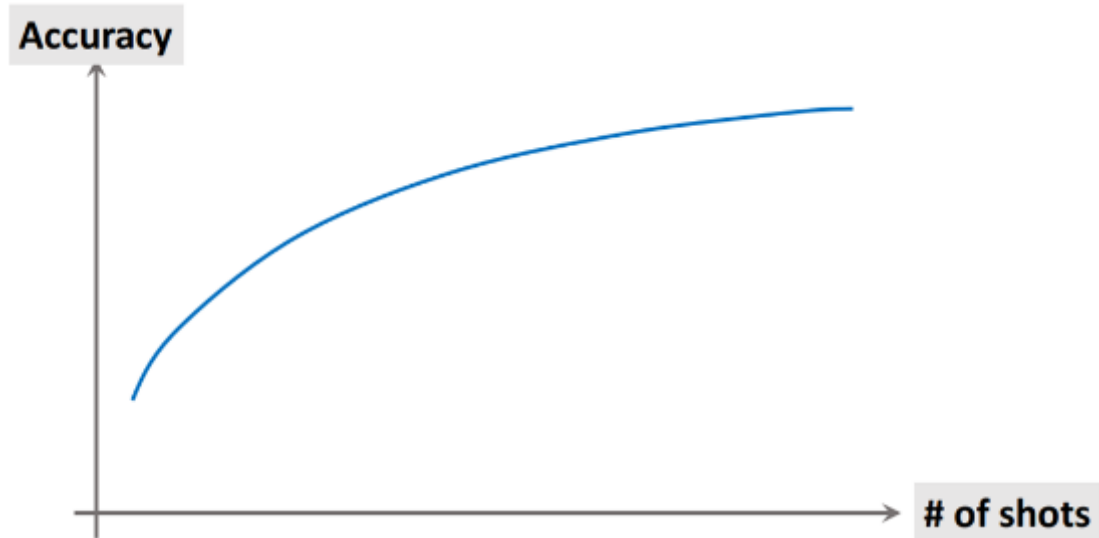
Prediction Accuracy VS. # of ways



3-way is easier than 6-way



Prediction Accuracy VS. # of shots



2-shot is easier than 1-shot



Are they the same kind of animal?



Are they the same kind of animal?



Basic Idea (Learn a Similarity Function)

- Learn a similarity function: $\text{sim}(\mathbf{x}, \mathbf{x}')$.
- Ideally, $\text{sim}(\mathbf{x}_1, \mathbf{x}_2) = 1$, $\text{sim}(\mathbf{x}_1, \mathbf{x}_3) = 0$, and $\text{sim}(\mathbf{x}_2, \mathbf{x}_3) = 0$.

Bulldog



\mathbf{x}_1

Bulldog



\mathbf{x}_2

Fox



\mathbf{x}_3

- Basic Idea (Learn a Similarity Function)
- Apply the similarity function for prediction.
 - Compare the **query** with every sample in the **support set**.
 - Find the sample with the highest similarity score.

Support Set



What is in the image?

Query:



sim = 0.2

sim = 0.1

sim = 0.03

sim = 0.05

sim = 0.7

sim = 0.5

Greyhound

Bulldog

Armadillo

Pangolin

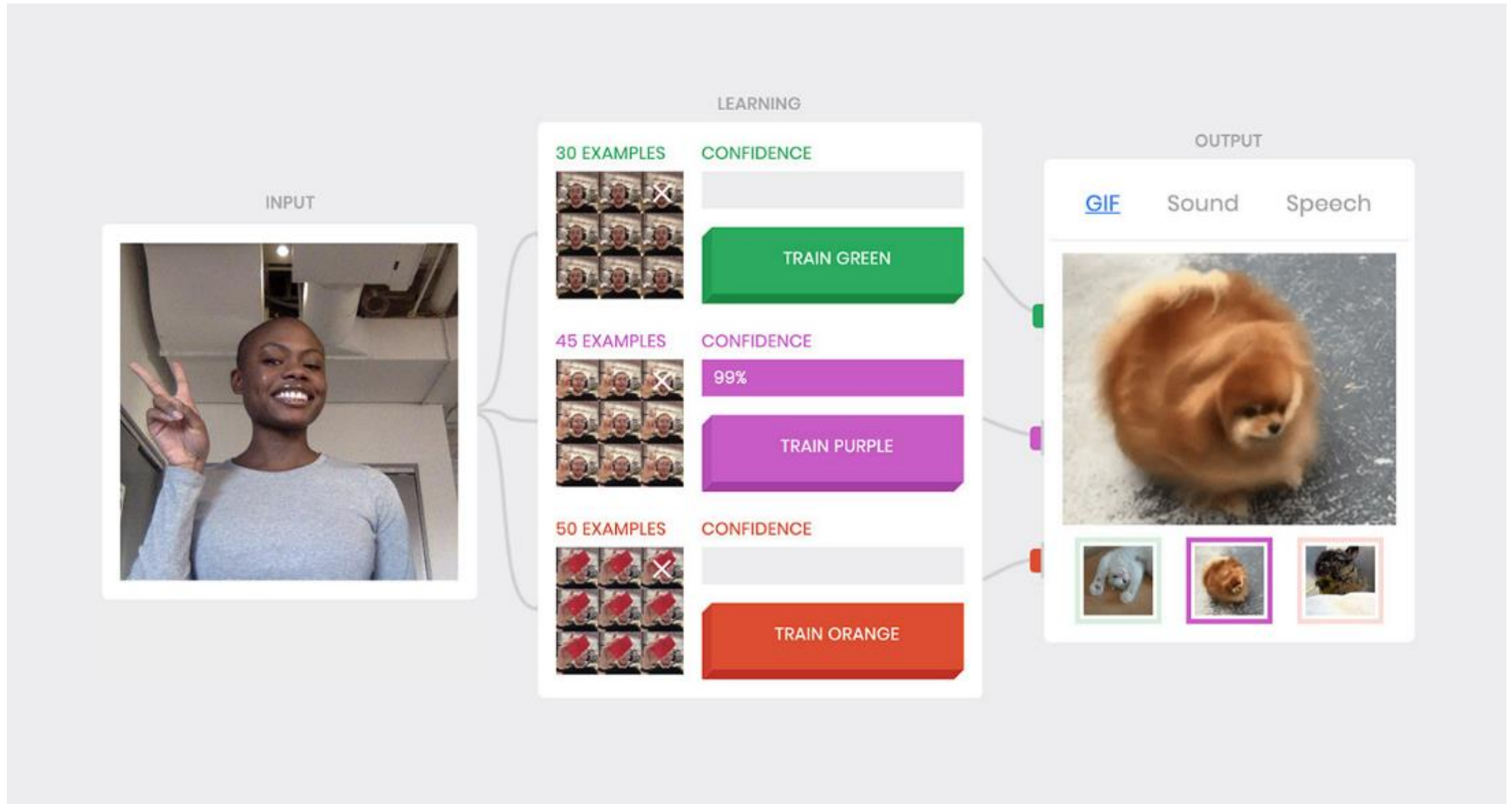
Otter

Beaver



- **Less data required to train:** Meta Learning helps to build **more general frameworks** that can transfer information from one context to another. This reduces the amount of data you need in the new context to solve the problems.
- **Speed:** Meta-learning ways and method help in producing **custom made models** which perform **better at a high speed**.
- **Scalable:** Meta-Learning models help to **increase the scalability** of an AI application by automating the process and improving algorithms.
- **Agile and adaptable** to **environmental changes** like Reinforcement learning.





Types of Meta-learning

- Metric-Based
- Model-Based
- Optimization-Based

- **Metric-Based**
 - **Siamese network**
 - **Prototypical Networks**

- In the **metric-based** meta learning setting, we will learn the appropriate metric space (i.e., learn the **similarity between two images**)
- The core idea in metric-based meta-learning **is similar to nearest neighbors algorithms** (i.e., k-NN classifier and k-means clustering)
- In the metric-based setting, we use a **simple neural network** that **extracts the features** from **two images** and **finds the similarity** by computing the **distance** between features of these two images.
- This approach is **widely used** in a **few-shot learning setting** where we **do not have many data points**.

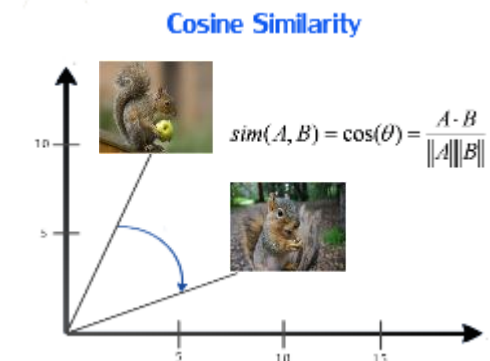
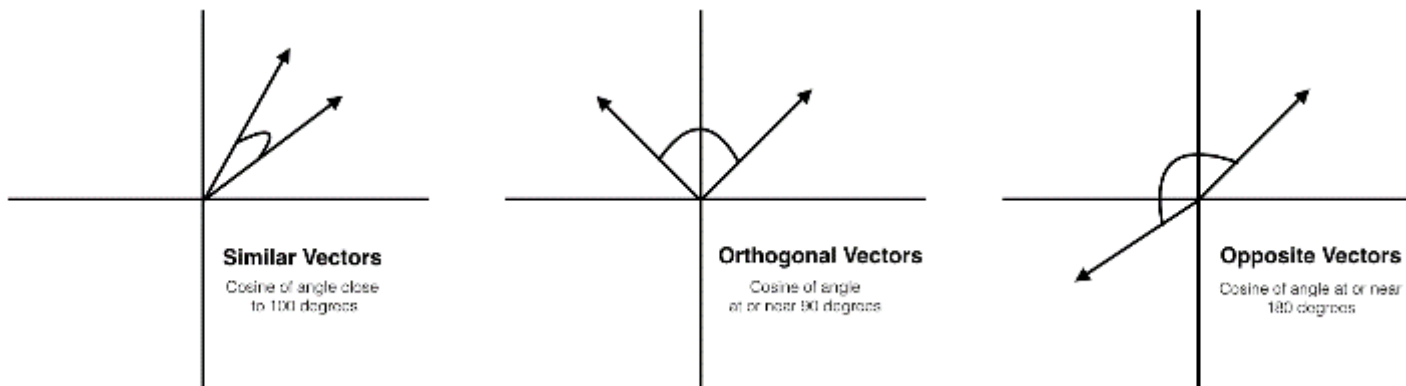
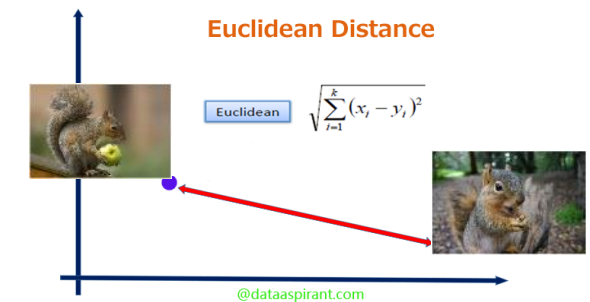
Euclidean distance

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

\mathbf{p}, \mathbf{q} = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

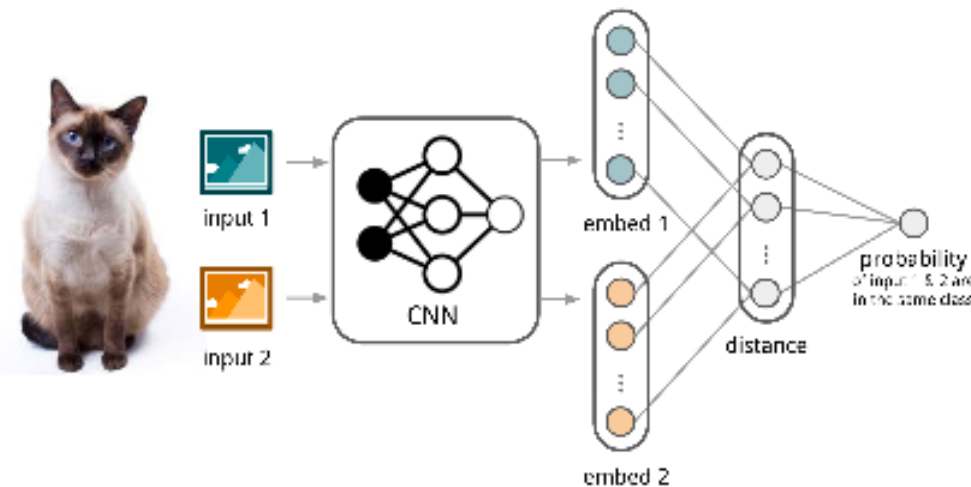
n = n-space

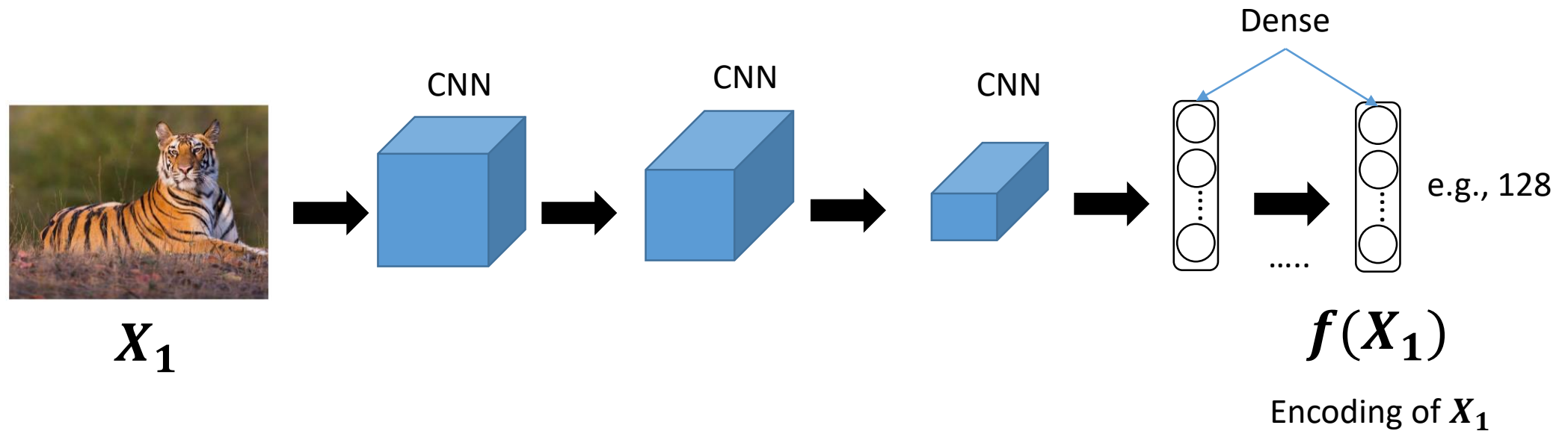


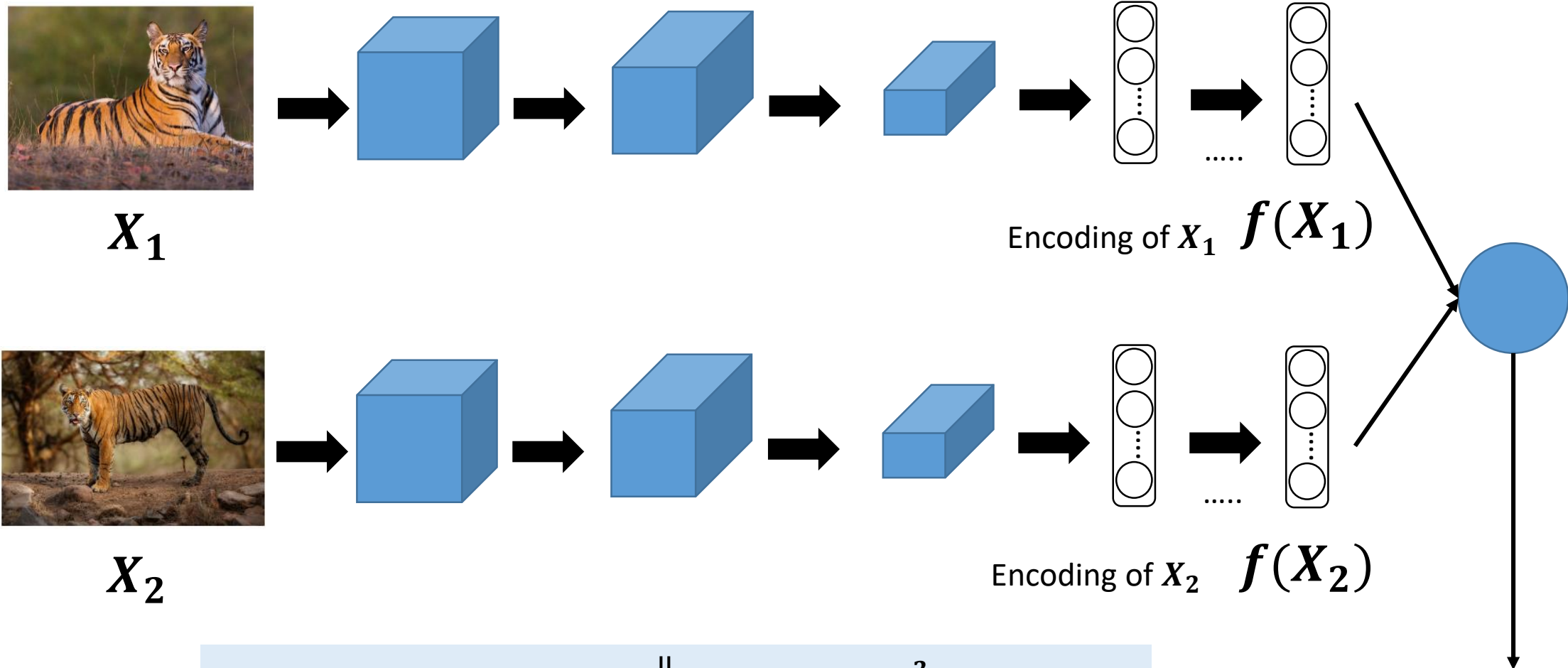
Various Types of **metric-based** learning networks

- Siamese networks
- Prototypical networks

- A **Siamese** neural network (sometimes called a **twin neural network**) is an artificial neural network that **uses the same weights** while working in **two different input vectors** to compute **comparable output vectors**. [1]
- Uses of **similarity measures** where a twin network might be used are such things as **recognizing handwritten, automatic detection of faces** in camera images, and matching queries with indexed documents.







X_1

Encoding of X_1 $f(X_1)$

X_2

Encoding of X_2 $f(X_2)$

If x_i, x_j are the same animal, $\|f(X_i) - f(X_j)\|_2^2$ is small.
 If x_i, x_j are the different animals, $\|f(X_i) - f(X_j)\|_2^2$ is large.

$$d(x_1, x_2) = \|f(X_1) - f(X_2)\|_2^2$$

Preparing Training Data

Data Set



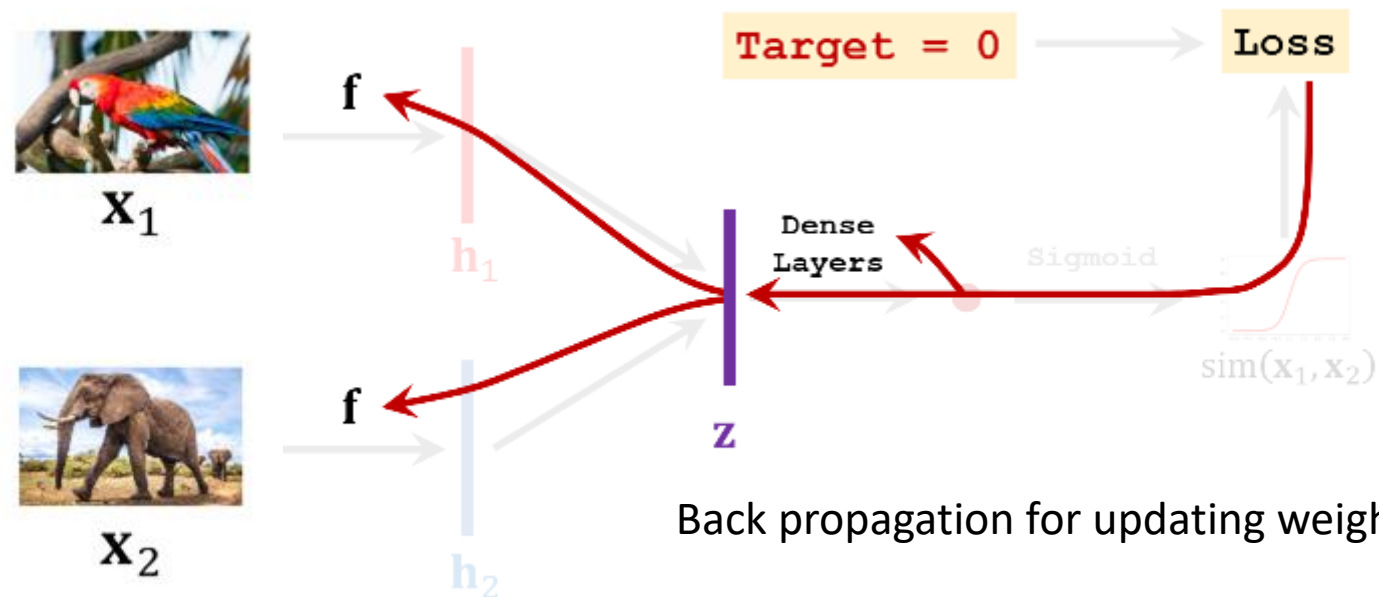
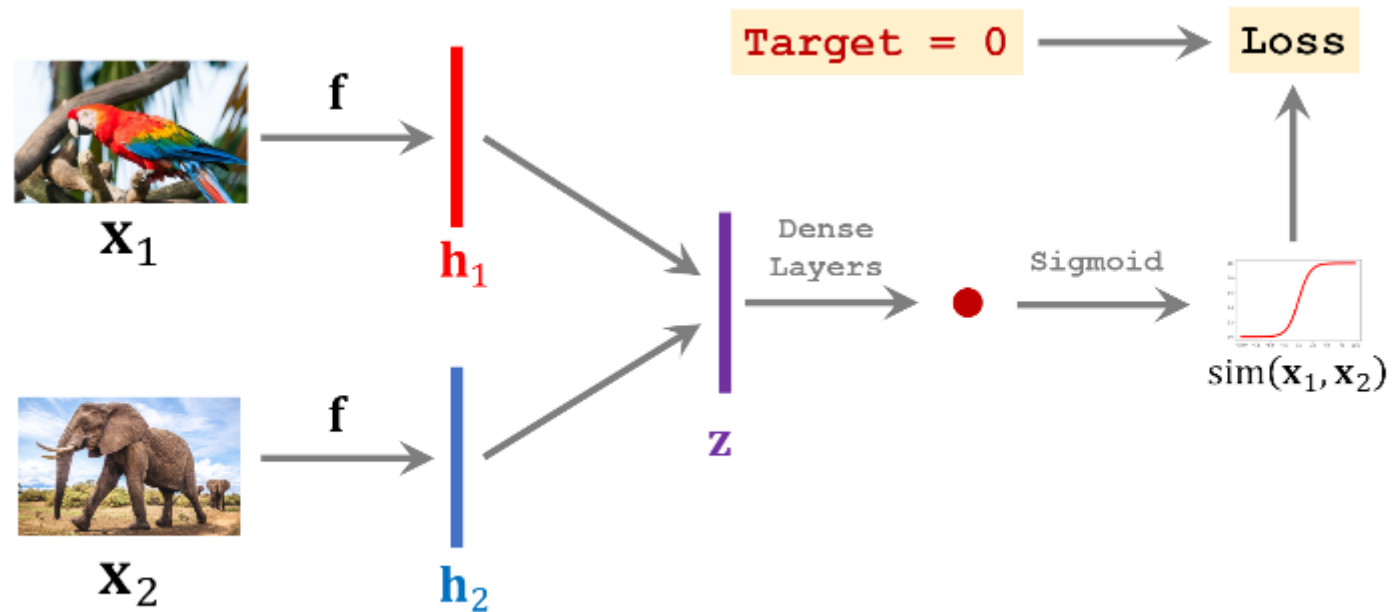
Training Data

Positive Samples

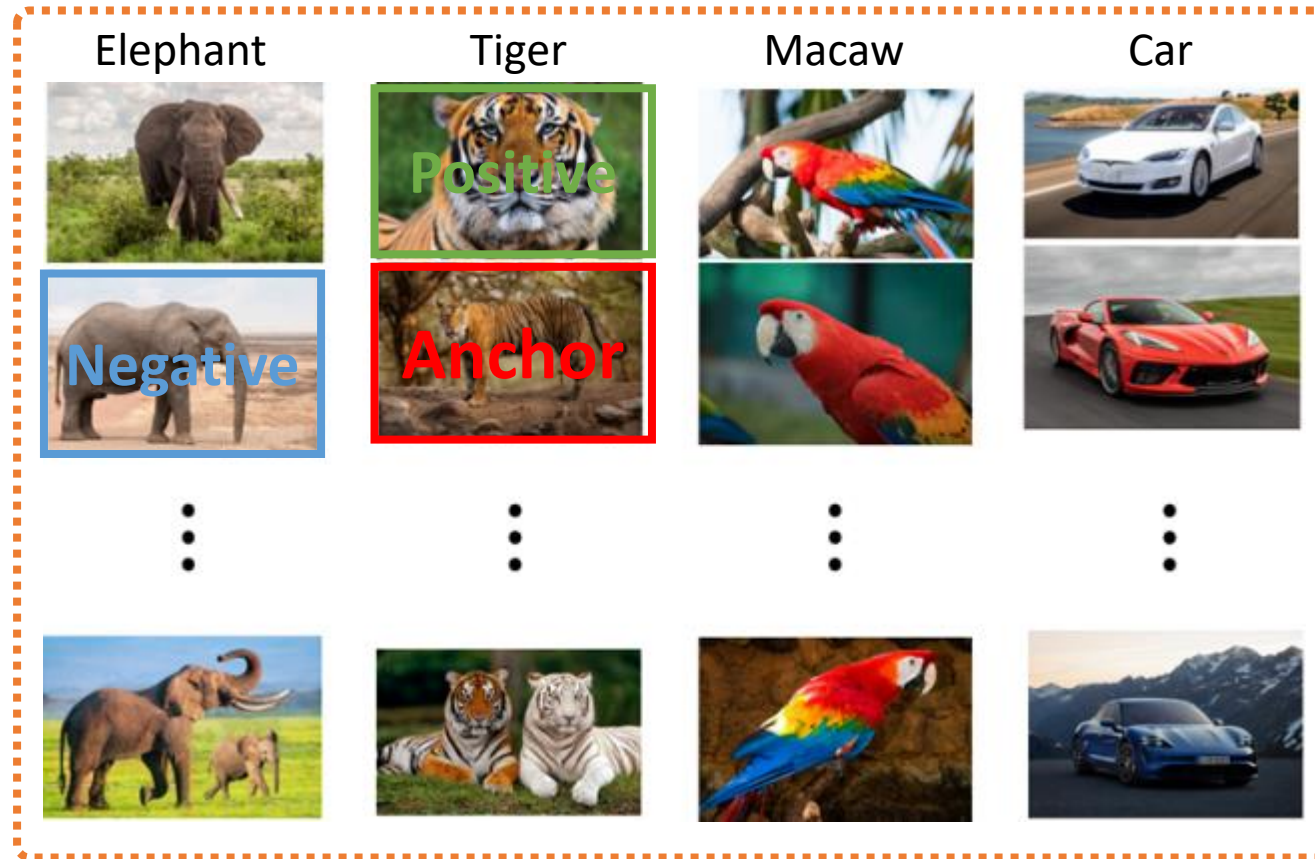


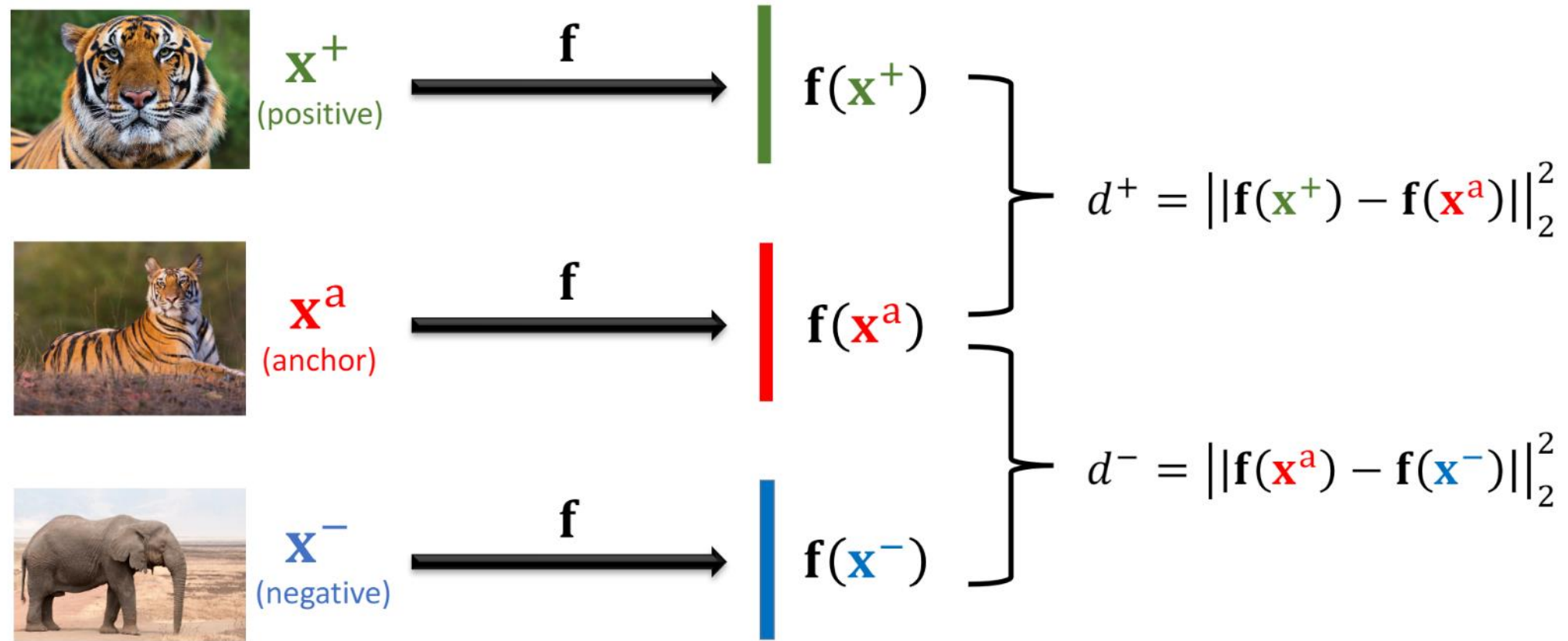
Negative Samples





Training Set







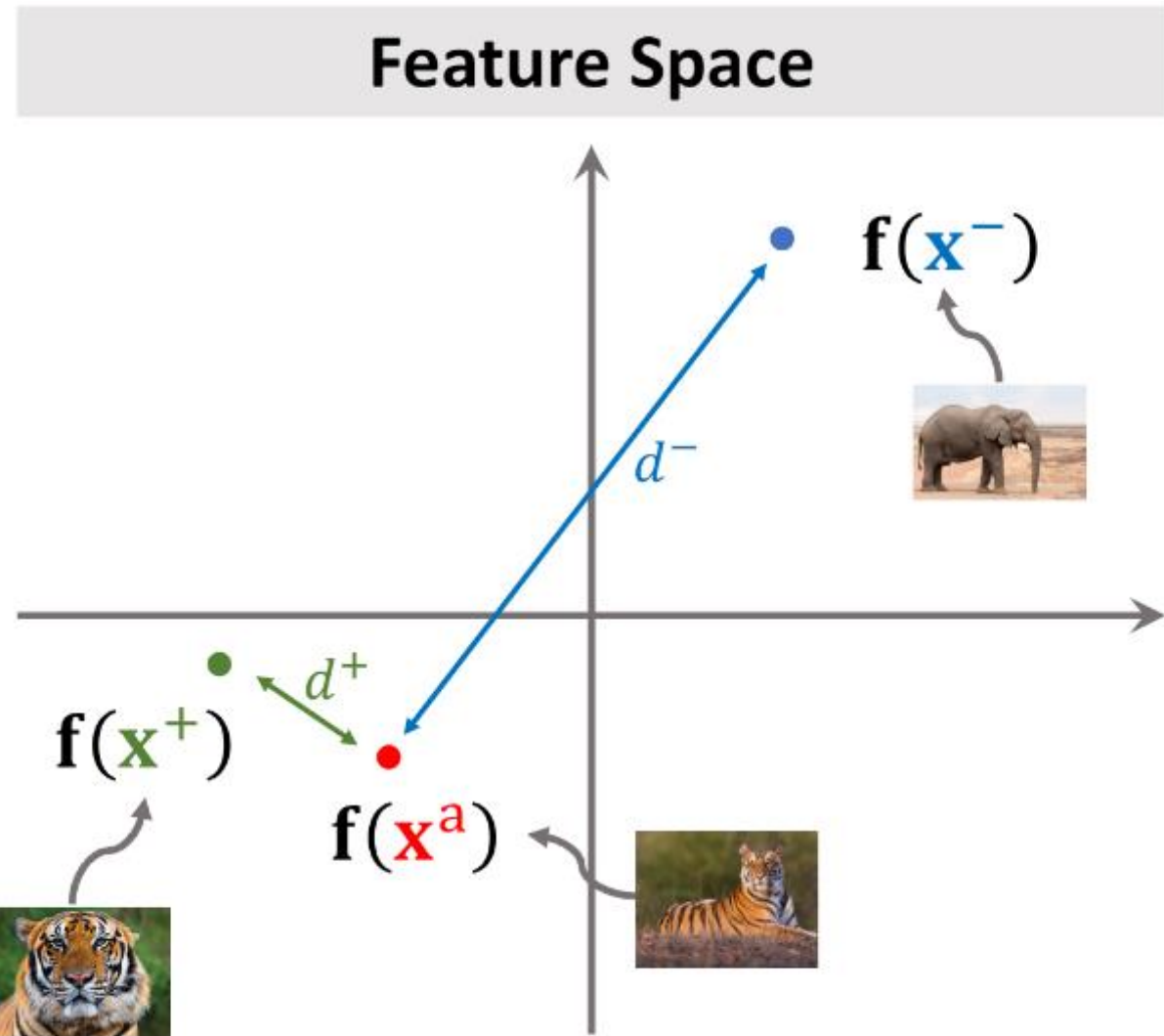
x^+
(positive)



x^a
(anchor)



x^-
(negative)





\mathbf{x}^+
(positive)



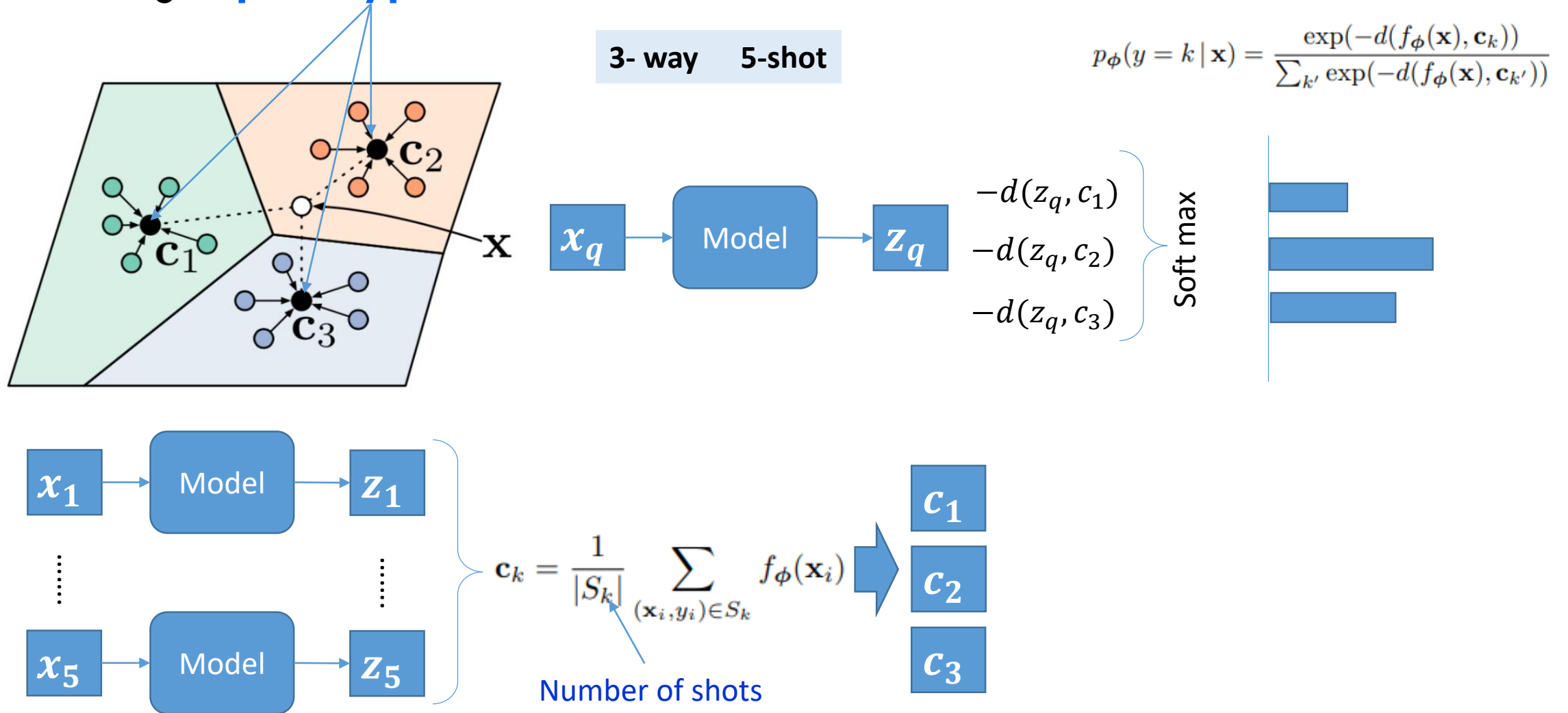
\mathbf{x}^a
(anchor)



\mathbf{x}^-
(negative)

- Encourage $d^+ = \|\mathbf{f}(\mathbf{x}^+) - \mathbf{f}(\mathbf{x}^a)\|_2^2$ to be small.
- Encourage $d^- = \|\mathbf{f}(\mathbf{x}^a) - \mathbf{f}(\mathbf{x}^-)\|_2^2$ to be big.
- If $d^- \geq d^+ + \alpha$, then no loss. ($\alpha > 0$ is margin.)
- Otherwise, the loss is $d^+ + \alpha - d^-$.
- $\text{Loss}(\mathbf{x}^a, \mathbf{x}^+, \mathbf{x}^-) = \max\{0, d^+ + \alpha - d^-\}$.
- Update the CNN (function \mathbf{f}) to decrease the loss.

- Training a **prototype extractor**



Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." *Advances in neural information processing systems*. 2017.

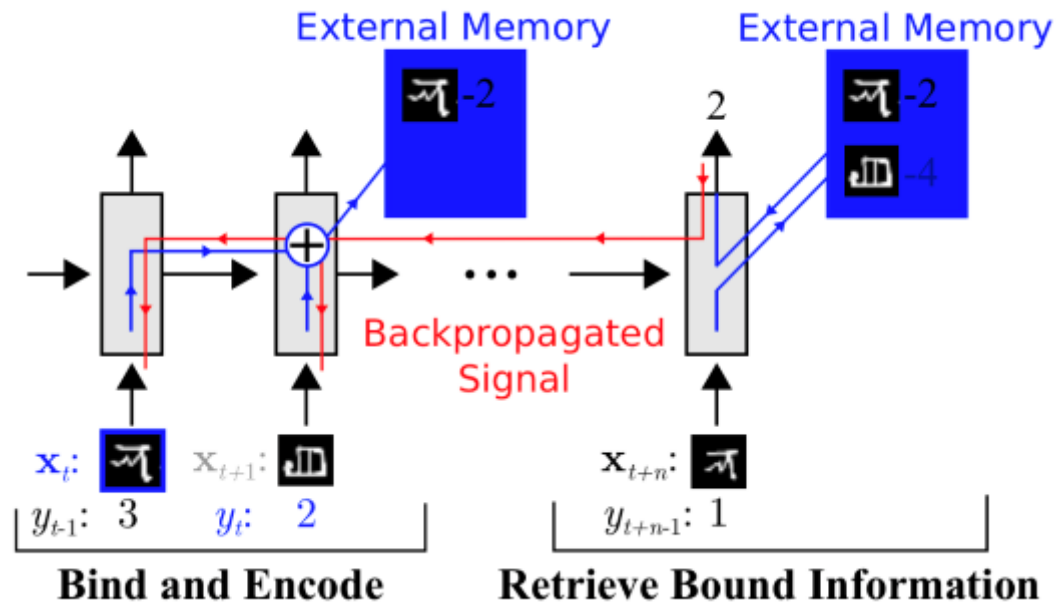
Types of Meta-learning

- Metric-Based
 - Model-Based
 - Optimization-Based
-
- Model-Based
 - Memory-Augmented Neural Networks

- Model-based meta-learning models updates its parameters rapidly with a few training steps, which can be achieved by its internal architecture or controlled by another meta-learner model[8].
- **Memory-Augmented Neural Networks**
 - **User Recurrent Networks**
 - **Add a memory**
 - **Example**
 - **Character Recognition**
 - **3 Labels**

Lilian Weng(2018). Meta-Learning: Learning to Learn Fast. OpenAI Blog . November 2018

- A family of model architectures use external memory storage to facilitate the learning process of neural networks, including Neural Turing Machines and Memory Networks.
- With an explicit storage buffer, it is easier for the network to rapidly incorporate new information and not to forget in the future.
- Such a model is known as MANN, short for “Memory-Augmented Neural Network”.
- Note that recurrent neural networks with only internal memory such as vanilla RNN or LSTM are not MANNs.

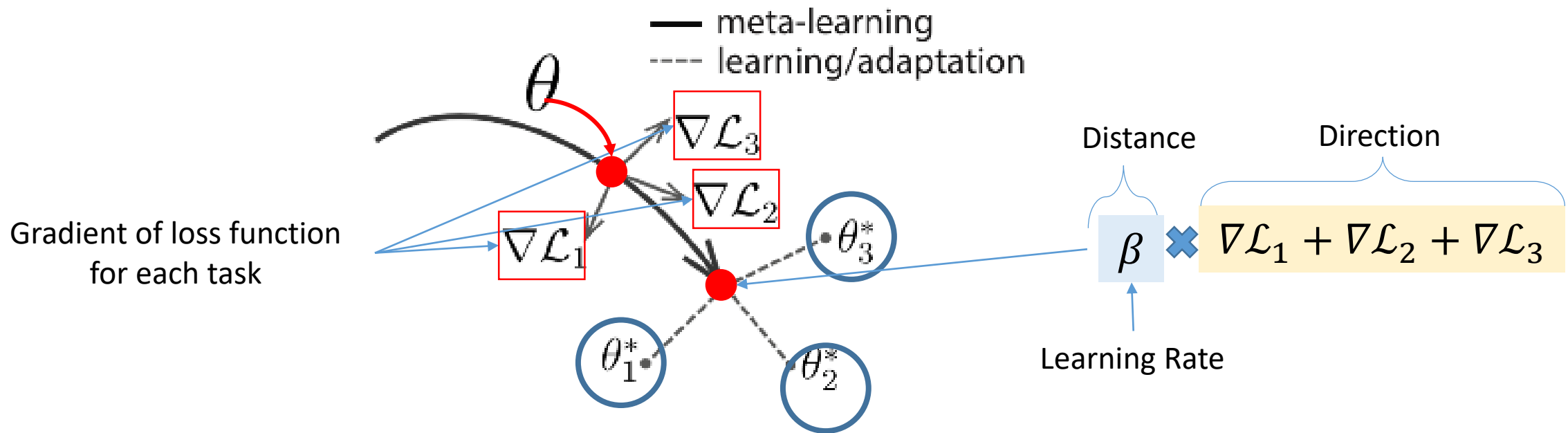


Types of Meta-learning

- Metric-Based
 - Model-Based
 - Optimization-Based
-
- Optimization-Based
 - Model Agnostic Meta Learning (MAML)

- Generally, we **optimize neural network** by training on a **large dataset** and **minimize the loss using gradient descent**.
- However, the **gradient-based optimization** is **neither designed to cope with a small number of training samples**, nor to **converge within a small number of optimization steps**.
- Is there a way to adjust the optimization algorithm so that the model can be good at **learning with a few examples**?

- **MAML is one of the recently introduced and most popularly used meta learning algorithms** and it has created a major breakthrough in meta learning research.
- The basic idea of MAML is **to find a better initial parameter** so that, with good initial parameters, the model can **learn quickly on new tasks with fewer gradient steps**.



Step 1

Trains from the initialization θ on D_i^{train} to get θ_i

$$\theta_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{D_i^{train}}(\theta)$$

Inner Loop Learning Rate

Train Data

Step 2

Evaluates each task T_i on D_i^{valid} with θ_i and updates θ

$$\theta = \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} \mathcal{L}_{D_i^{valid}}(\theta_i)$$

Outer Loop Learning Rate

Validation Data

Direction

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β : step size hyperparameters

1: randomly initialize θ

2: **while** not done **do**

3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

4: **for all** \mathcal{T}_i **do**

5: Evaluate $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$ with respect to K examples

6: Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$

7: **end for** **Note: the meta-update is using different set of data.**

8: Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$

9: **end while**

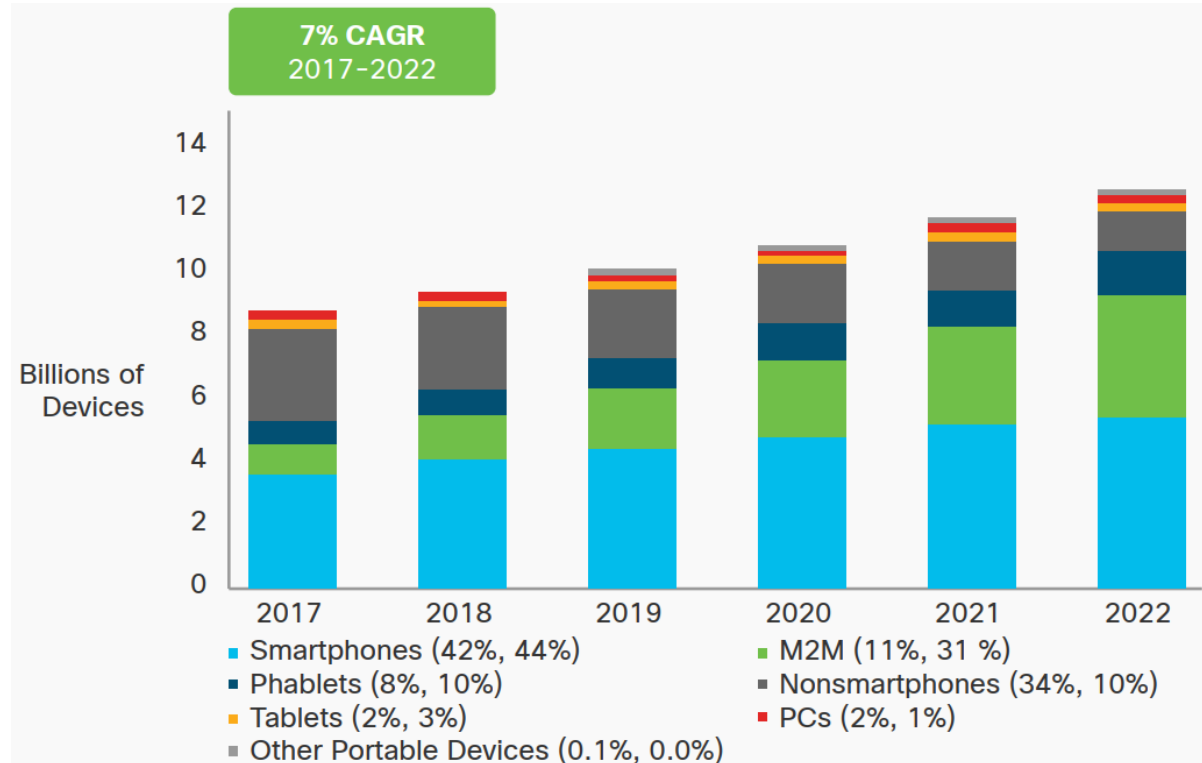
Outer Loop

Inner Loop

Use Case : Meta-Learning-Based Deep Learning Model Deployment Scheme for Edge Caching

- Main Components Needed for Architecture Search
- Search Strategy
- Meta-learning for Model Generation
- Type of meta learning for Model Generation
- System Model
- Problem Formulation
- Frame work
- Block Based Model Framework
- Reinforcement Meta Learning
- Performance Evaluation

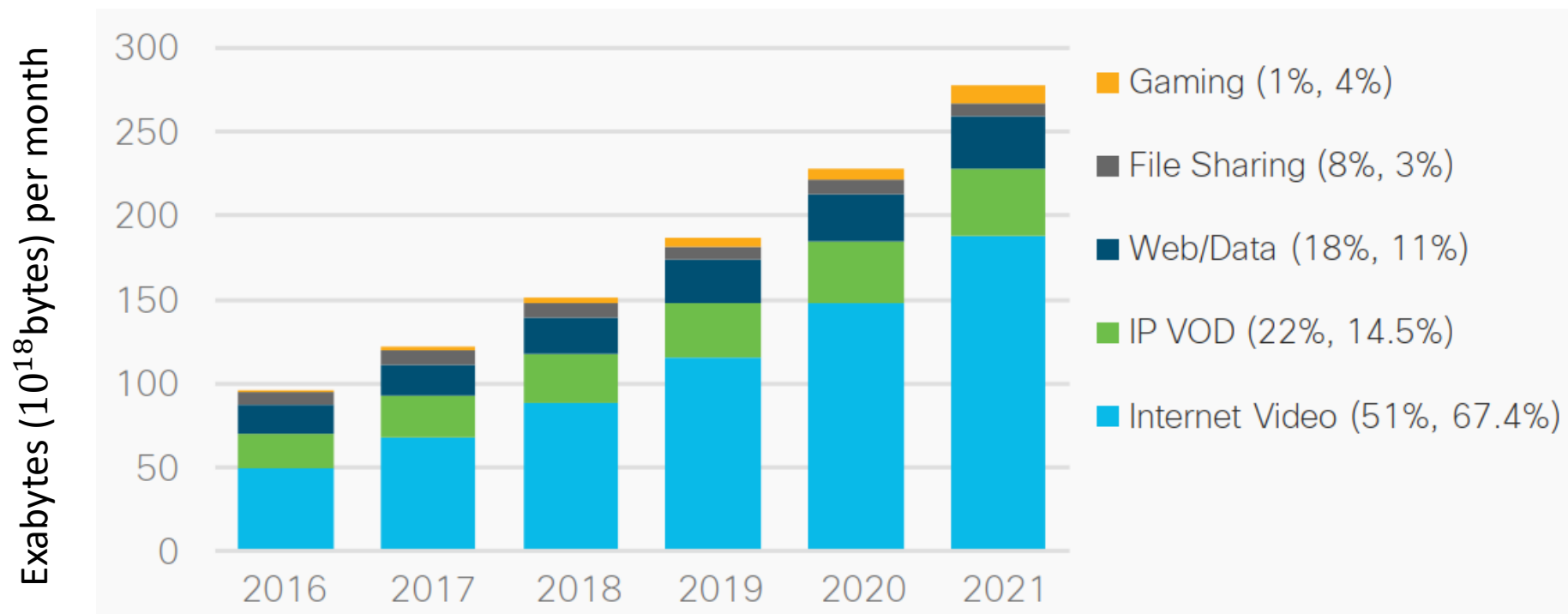
According to the Cisco Visual Networking Index, **watching videos from wireless devices has been generating most of the Internet traffic and is forecast to continue to grow exponentially.**



Source: Cisco Visual Networking Index (VNI), Feb. 2019.

Novel bandwidth hungry applications:

- Real time HD streaming
- Online Gaming
- Ultra-reliable and low-latency communication
- Virtual reality services
- Enhanced mobile broadband
- Mobile hologram
- And so on



Mobile data traffic growing (24% Compound Annual Growth Rate)

2/3 of the traffic is expected to be Internet Video

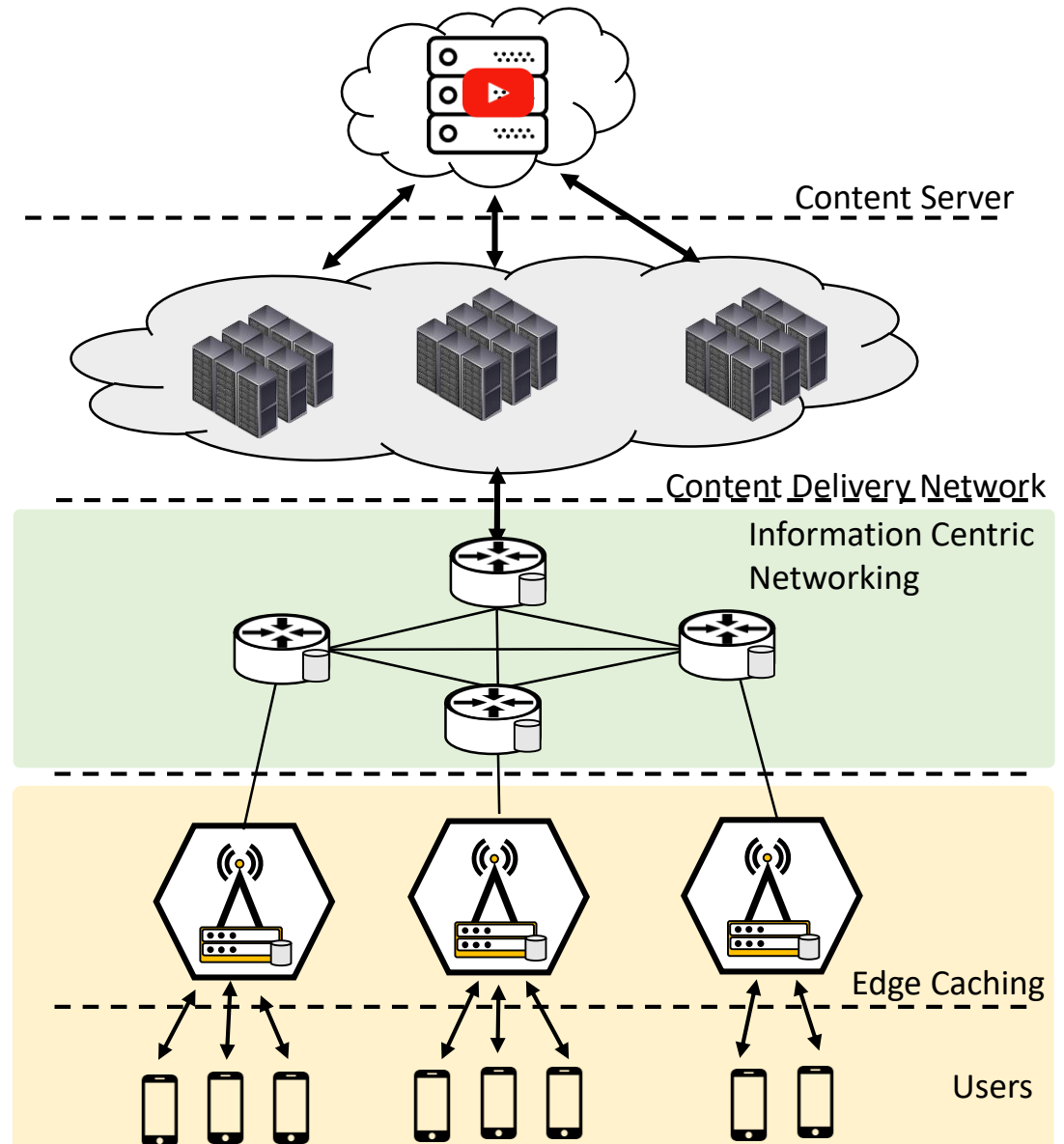
How to deal with increasing traffic demand, especially Video?

New architectures are proposed to **utilize caching in network nodes**, such as routers in the core network and base-stations in edge network [4].

Information Centric Networking (ICN) is one such architecture which shifts the network paradigm from **location-based to information-based network**.

Named Data Networking (NDN), the extension version of Content Centric Networking, is one of the most prominent ICN architecture [5].

Enabling caching at the edge nodes {the nodes located **nearest to the users** (i.e., MBS and SBSs) } can **significantly reduce the amount of re-downloading contents** from the original content servers, which leads to lower backhaul load [6], [7].



Cache decision can be classified into two main categories

Reactive

- The node makes the cache decision (to store the requested content or not) **only when the request for a particular content arrives** [8] and [9].

Proactive

- The node proactively predicts a content's popularity and makes the cache decision **before requested by any user** [10] and [11].

Content popularity can be defined as the ratio of the number of requests for a particular content to the total number of requests, usually obtained for a certain region during a given period of time.

Node: i.e., Router, Base Station, Small-cell Base Station

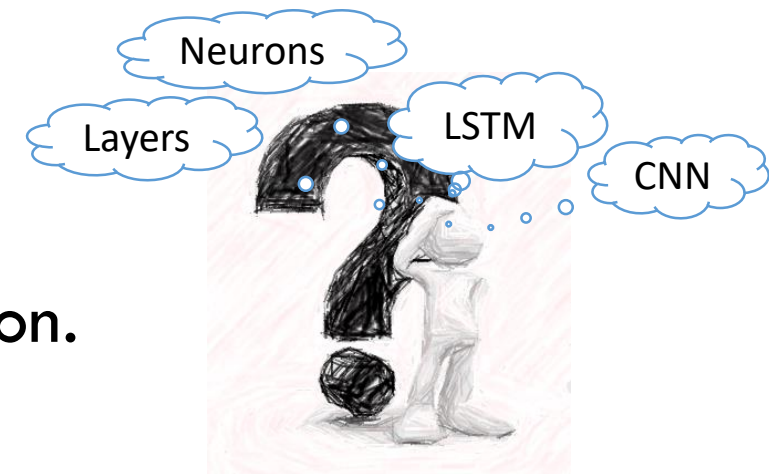
[8] K. Thar, N. H. Tran, S. Ullah, T. Z. Oo, and C. S. Hong, "Online caching and cooperative forwarding in information centric networking," IEEE Access, vol. 6, pp. 59679–59694, Oct. 2018.

[9] A. Ndikumana, K. Thar, T. M. Ho, Nguyen H. Tran, Phuong L. Vo, Dusit Niyato, and Choong Seon Hong, "In-network caching for paid contents in content centric networking," in Proc. GLOBECOM-IEEE Global Commun. Conf., Dec. 2017, pp. 1–6.

[10] E. Zeydan et al., "Big data caching for networking: Moving from cloud to edge," IEEE Commun. Mag., vol. 54, no. 9, pp. 36–42, Sep. 2016.

[11] M. Chen, W. Saad, C. Yin, and M. Debbah, "Echo state networks for proactive caching in cloud-based radio access networks with mobile users," IEEE Trans. Wireless Commun., vol. 16, no. 6, pp. 3520–3535, Jun. 2017.

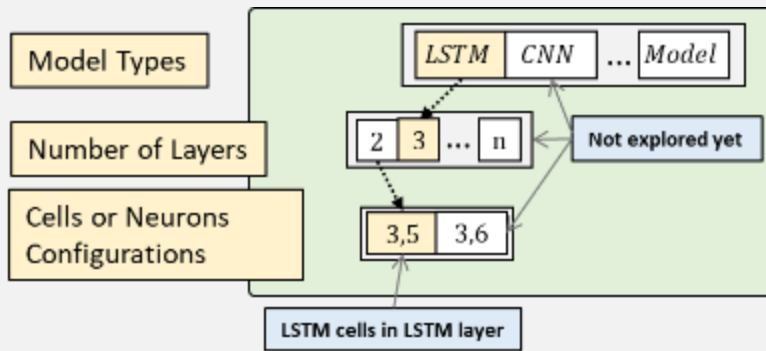
- Constructing an appropriate deep learning model is **complicated and time-consuming process even for the human experts.**
- This is because human experts need to find out followings to solve the domain-specific problem with satisfactory performance.
 - **Relevant deep learning architectures**
 - Training procedures, and
 - **Hyperparameters**
- This process has to be repeated for every application.
- So, we proposed the **Model Architecture search** to construct the Deep Learning Models.



Search Space

Search Strategy

Performance Estimation Strategy



LSTM = Long-Short Term Memory
CNN = Convolutional Neural Network

Search strategies **find a deep learning architecture** to improve the performance such as validation accuracy.

- Grid search
- Randomized search
- Reinforcement learning based search
- Meta-learning + Reinforcement

Performance estimation strategy **estimates the performance** of an architecture **to guide the search process**.

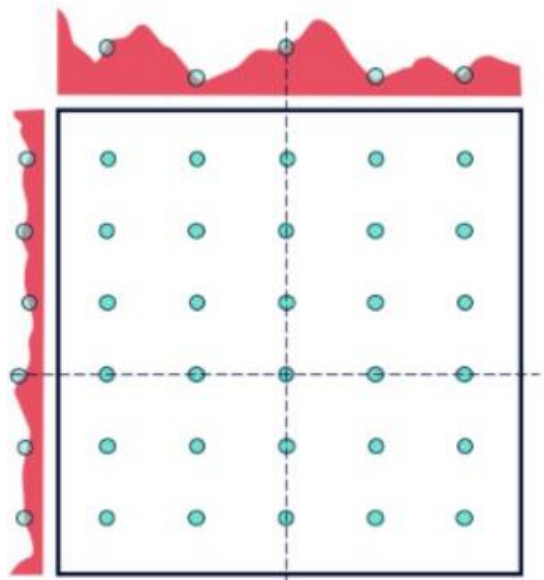
Reduce the computational burden

Performance can be estimated based on **lower fidelities** of the actual performance after full training [21]

- Shorter training times
- **Training on a subset of the data**
- Training on lower-resolution images
- Training with less filters per layer

Grid-search

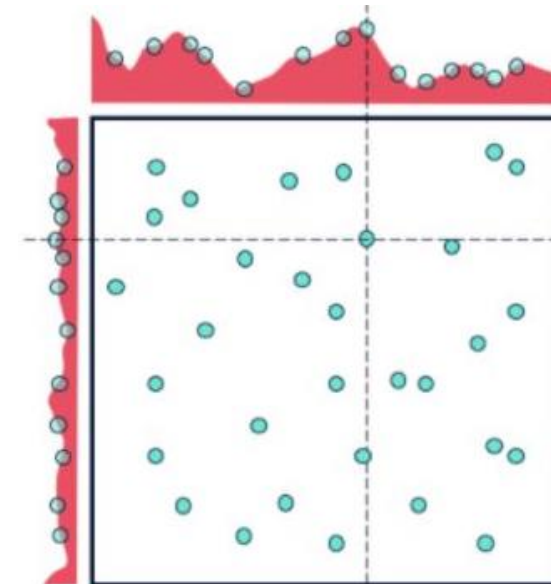
- Finds the best model by increasing the hyperparameter values in **sequential combinations**.



Grid Search

Random search

- Finds the best model among random configurations in order to **reduce search space**.



Random Search

Reinforcement

- **Guide the random exploration** in the likely direction of learning models with better performance
- **Lack of feedback** mechanism from the Edge nodes to search and construct the better model than the currently used models such as LSTM, CNN, etc.

Meta-learning + Reinforcement

- Meta-learning (learning to learn) is the approach to create learning models that can **learn new abilities or acclimate to new circumstances rapidly from experience** (meta-data).

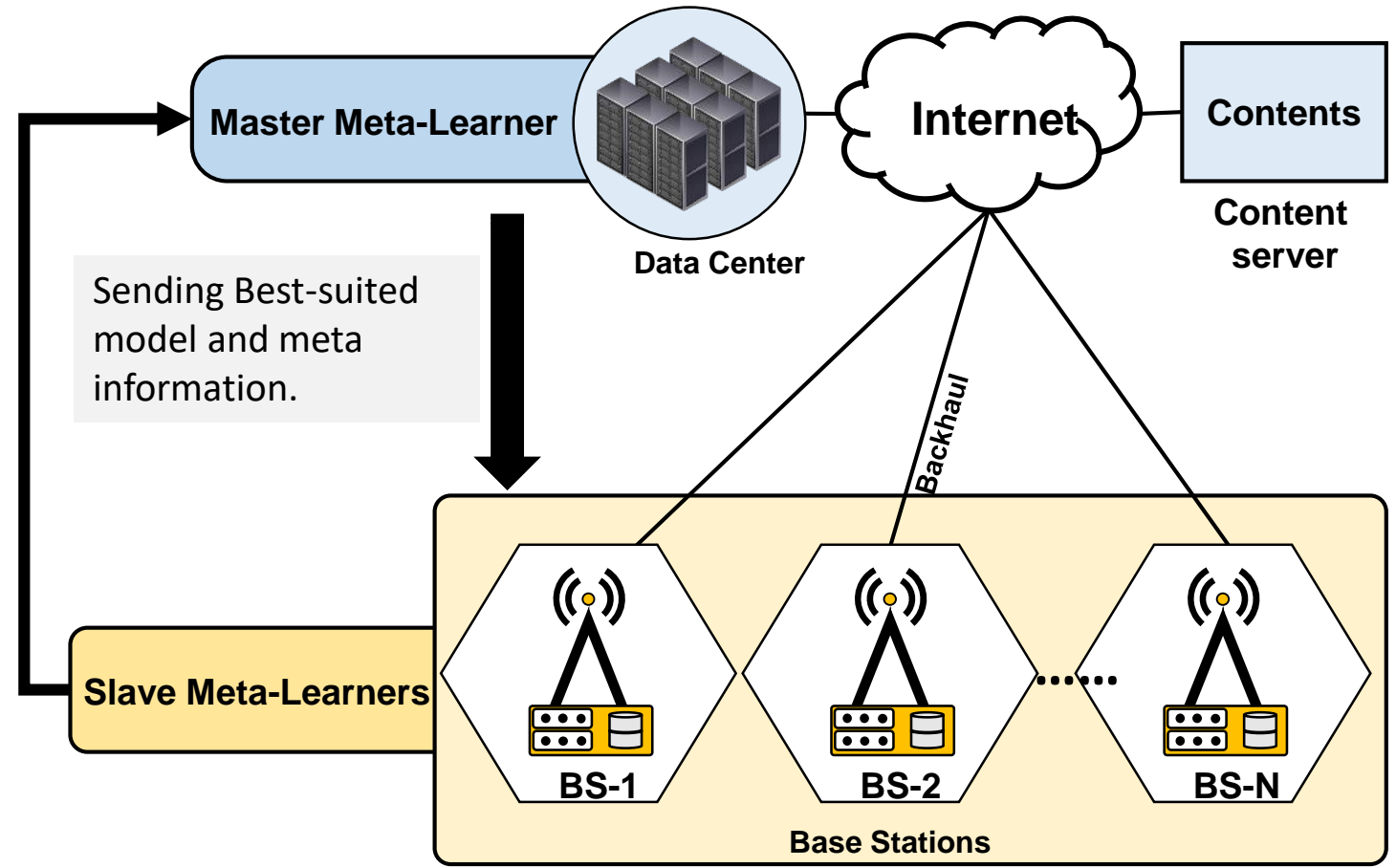
Learning from **Experience (meta-data)** to learn new tasks much faster

What are Meta data for Model Generation?

Meta data comprise the exact algorithm configurations used to train the models

- ❖ Hyper parameters
- ❖ Deep Neural Network architectures,
- ❖ Model evaluations (i.e., accuracy and training time)
- ❖ Etc..

- **Goal:** Maximize the cache hit, in order to reduce access latency
- **Input:** Features information such as number of request
- **Output:** Content's popularity score
- **Dataset:** MovieLens (<https://grouplens.org/datasets/movielens/>)



System Model

Content retrieval cost from Cloud to Edge node

Cache decision for content f at base station b

$$C_{b,cache}^{(t+1)} = \sum_{f \in \mathcal{F}} (1 - x_{b,f}^{(t+1)}) \phi_{b,f}^{(t+1)} \kappa_f p_b^{\text{retrieve}}, \quad \forall b \in \mathcal{B},$$

Number of requests for content f at base station b

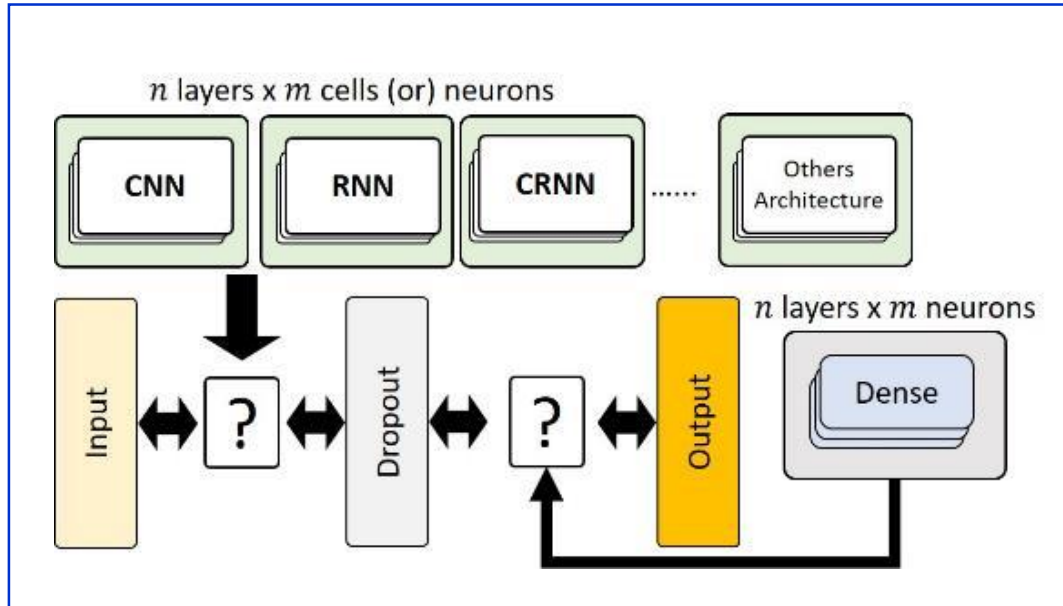
Size of content f

Content retrieving cost at base station b

minimize: $\frac{1}{T} \sum_{t \in \mathcal{T}} \sum_{b \in \mathcal{B}} C_{b,cache}^{(t+1)}$ T: 24 hours

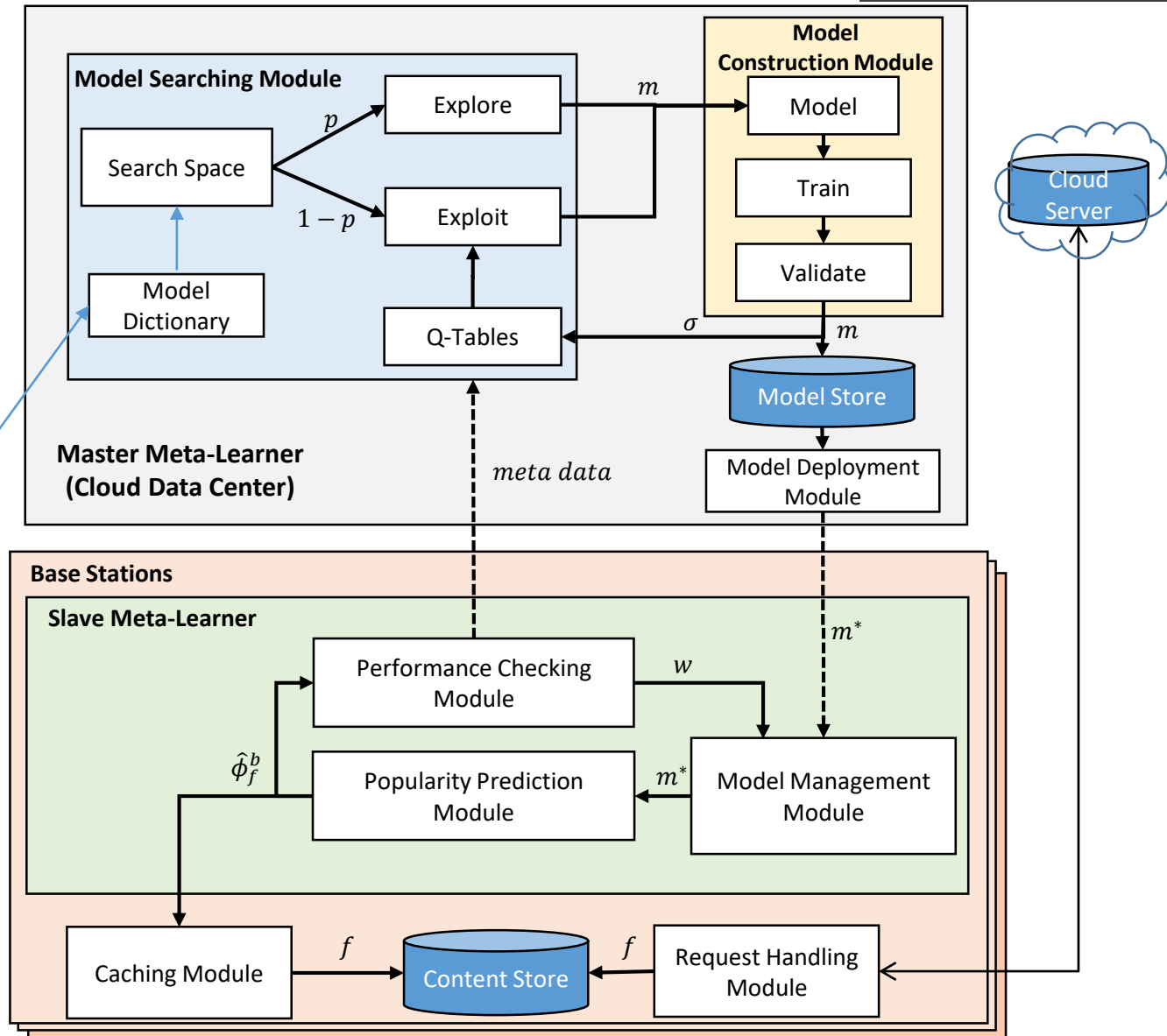
subject to: $\sum_{f \in \mathcal{F}} x_{b,f}^{(t)} \kappa_f \leq S_b, \quad x_{b,f}^{(t)} \in \{0, 1\}, \quad \forall b \in \mathcal{B}, \quad \forall t.$

Cache size of base station b

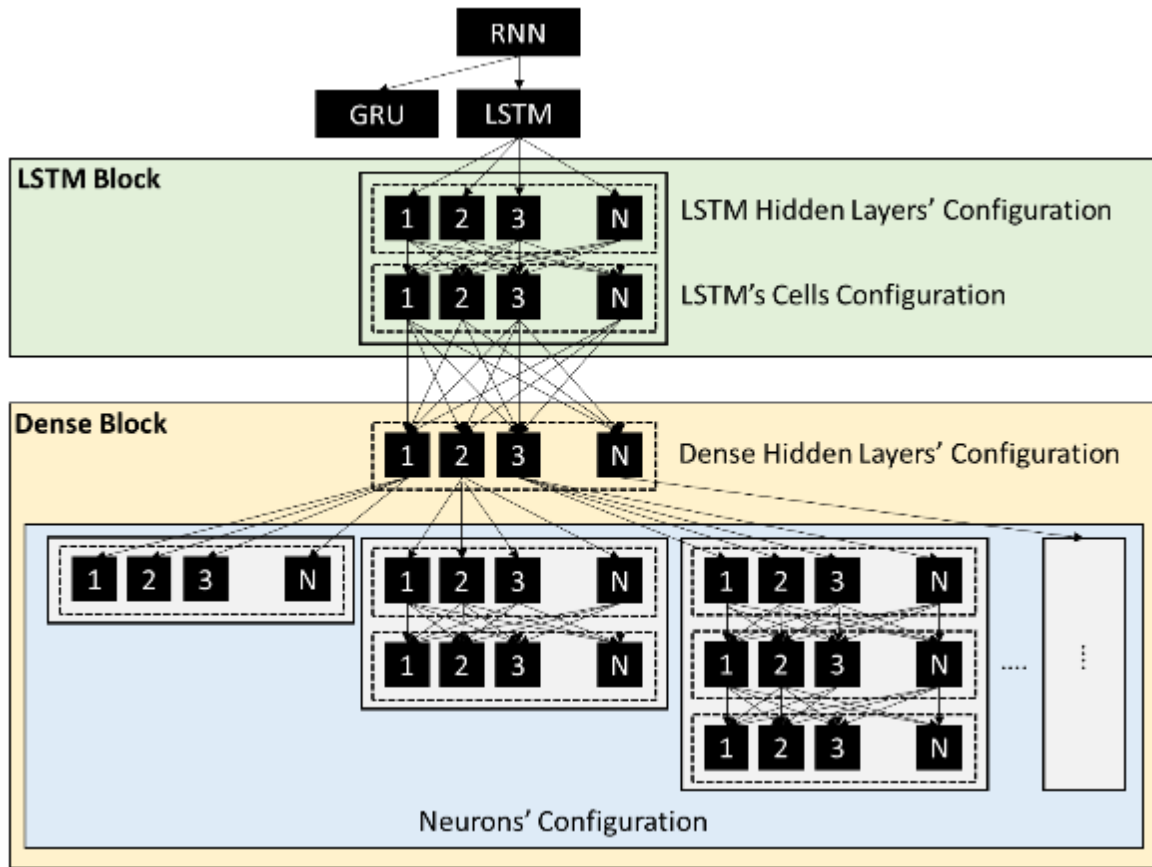


CNN = Convolutional Neural Network
 RNN = Recurrent Neural Network
 CRNN = Convolutional Recurrent Neural Network

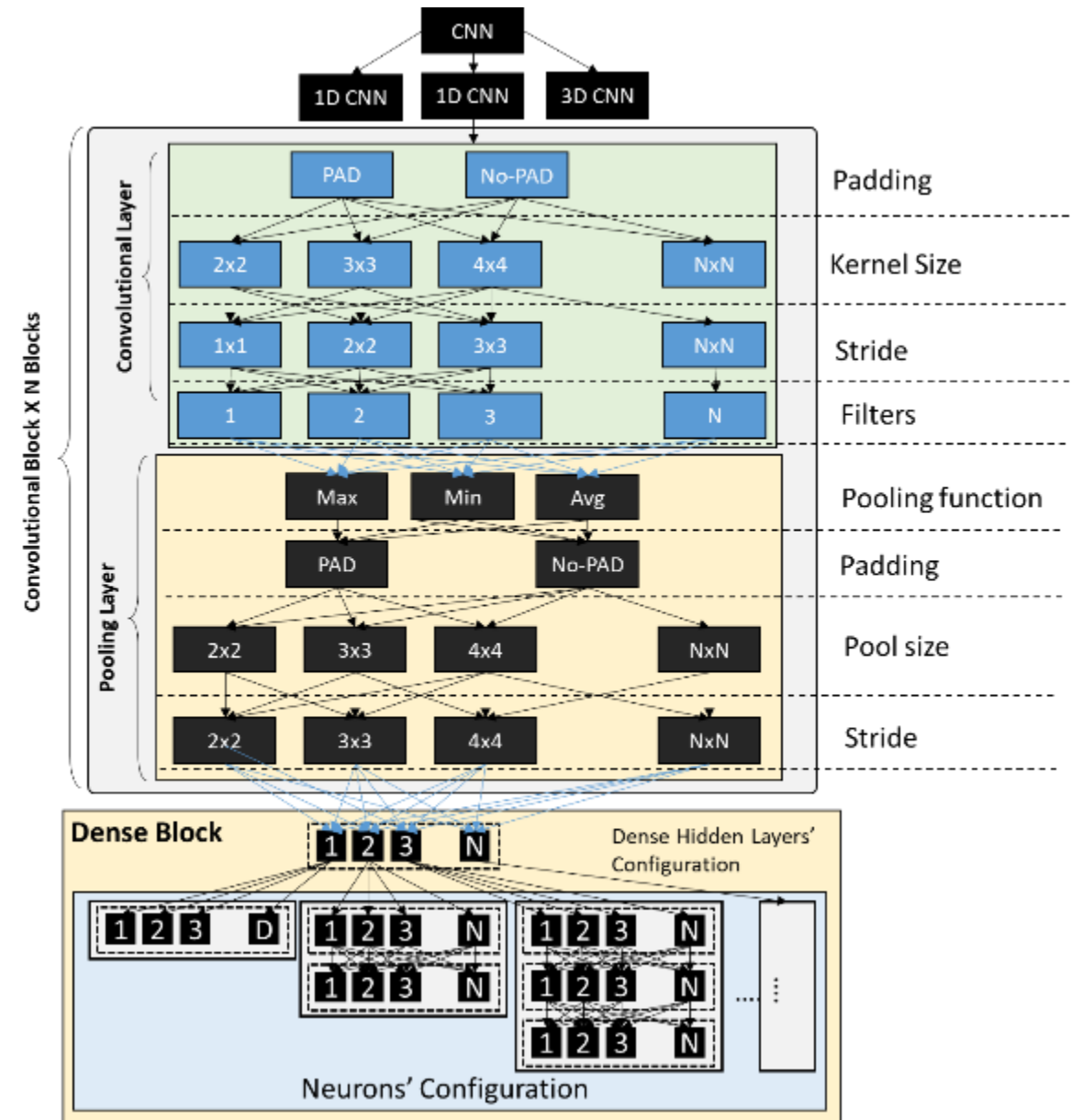
m^* (meta data): Learning Model Configurations (i.e., model serial number, number of layers, weight, etc.)
 σ : reward value
 w : weights parameter
 $\hat{\phi}_f^b$: number of predicted requests
 f : kind of contents

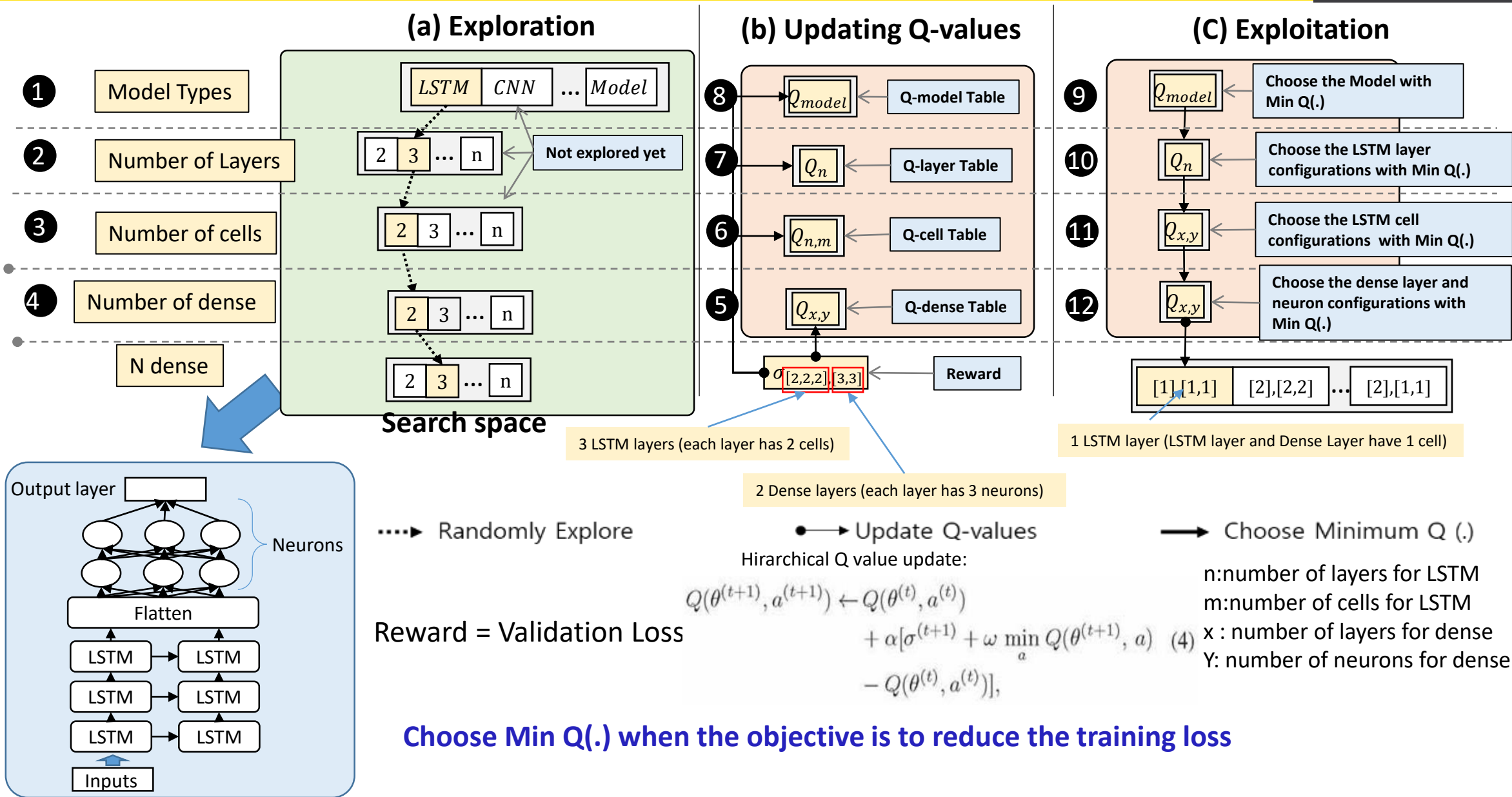


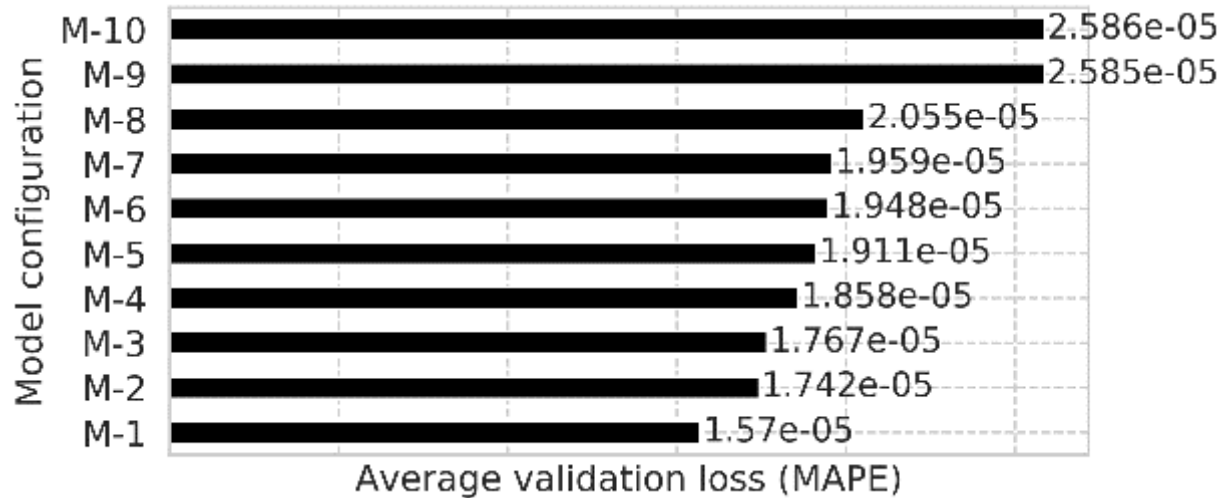
Block Based Model Framework



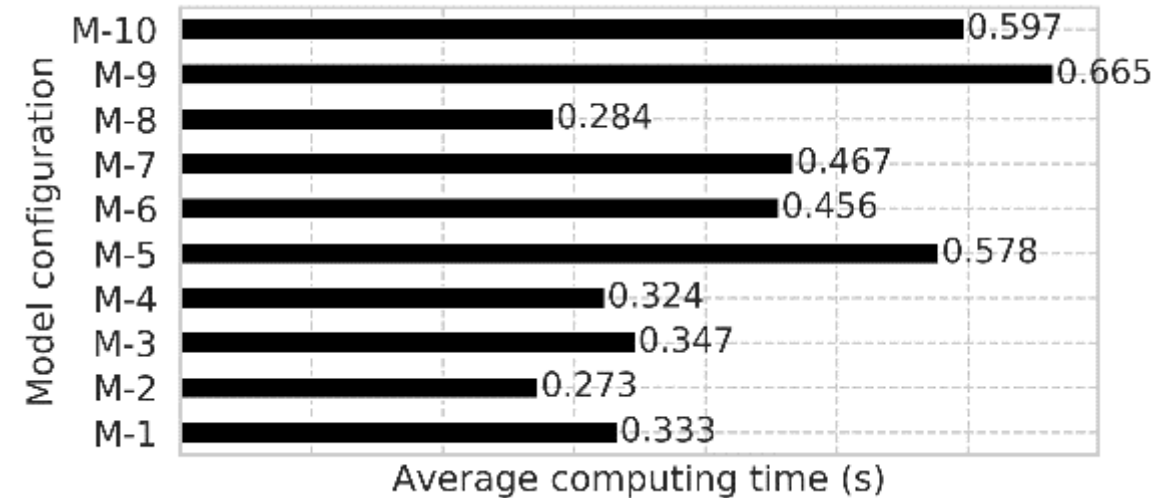
RNN = Recurrent Neural Network
 GRU = Gated Recurrent Unit
 LSTM = Long-Short Term Memory
 CNN = Convolutional Neural Network





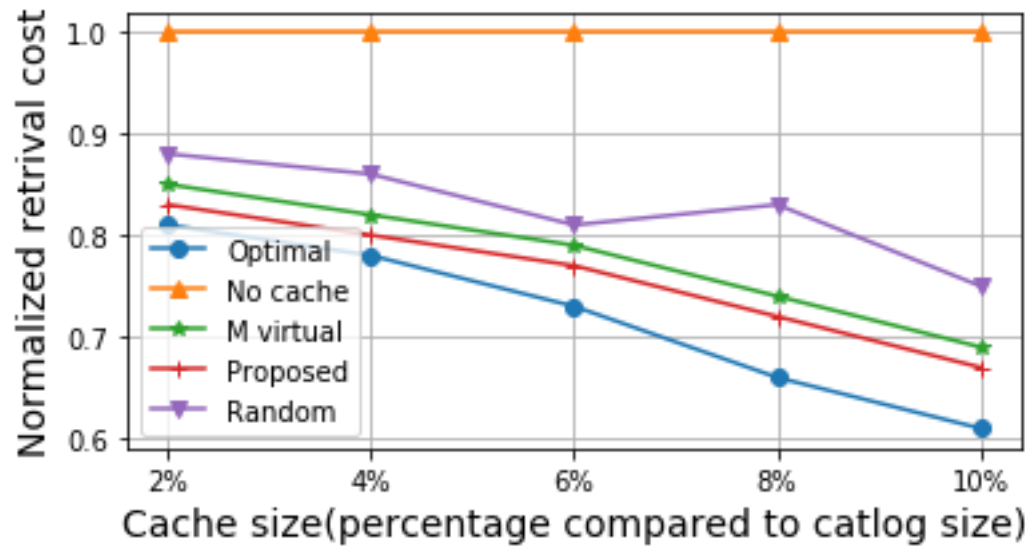


Average validation loss

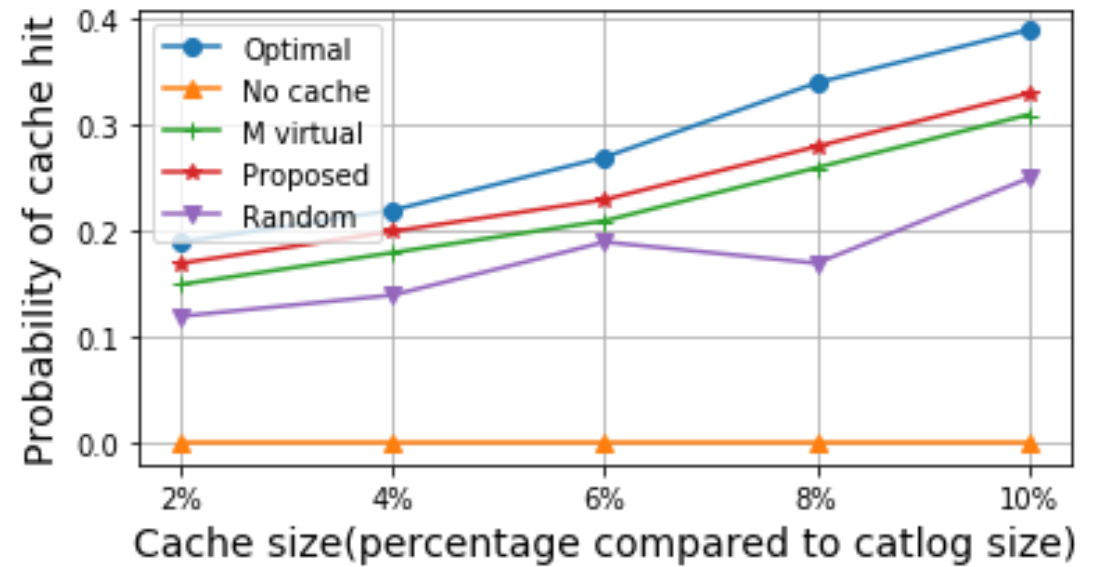


Average computing time

Model	Model Type	Configuration	Validation loss	Computing Time (s)
M 1	LSTM	LSTM <73> <73>, Dense <171, 102, 36, 188>	1.57e-05	0.333
M 2	LSTM	LSTM <68>, Dense <63, 64, 67, 64>	1.742e-05	0.273
M 3	LSTM	LSTM <26> <26> <26>, Dense <152, 25, 112, 162>	1.767e-05	0.347



Retrieval cost



Probability of cache hit

Kyi Thar, Thant Zin Oo, Zhu Han, and Choong Seon Hong, "Meta-Learning-Based Deep Learning Model Deployment Scheme for Edge Caching," 15th International Conference on Network and Service Management (CNSM 2019), Oct 21-25, 2019, Halifax, Canada

Use Case : Edge Computing Resources Reservation in Vehicular Networks: A Meta-Learning Approach

- Introduction
- System Model
- Problem Formulation
- The proposed meta-learning framework
- Simulation Results

Goal: To accurately predict the resource consumption in edge nodes.

- With the development of **autonomous vehicular technologies**, the **execution tasks** become **more memory-consuming** and **computation-intensive**.
- Simultaneously, a certain portion of **tasks are latency-sensitive**, such as collaborative perception, path planning, collaborative simultaneous localization and mapping, real-time pedestrian detection, etc.
- Because of the **limited computation resources inside vehicles** and **restricted transmission bandwidth**, **edge computing** can be **an effective way to assist** with the **tasks execution**.

- Therefore, to predict the resource consumption in edge nodes accurately in different scenarios
 - We propose a **two-stage meta-learning** based approach to **adaptively choose the appropriate machine learning algorithms** based on the **meta-features** extracted on database.

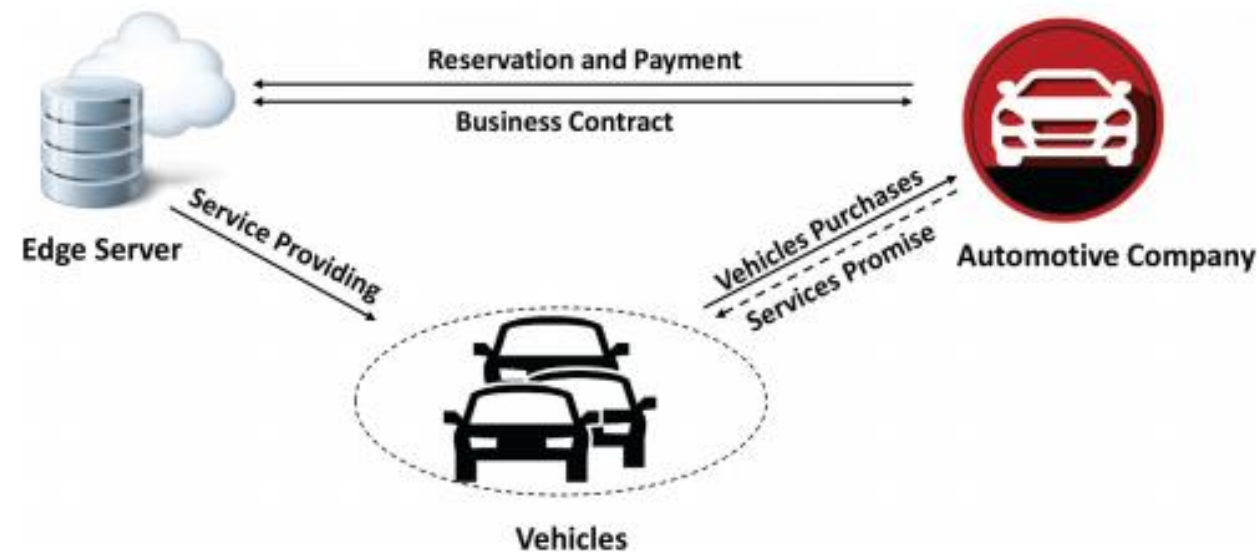


Fig. 1. The scenario and procedures description.

Total cost of consuming edge services

$$S(t) = \begin{cases} m^{(t)} \times P_{rv}, & n^{(t)} \leq m^{(t)}, \\ m^{(t)} \times P_{rv} + (n^{(t)} - m^{(t)}) \times P_{rt}, & m^{(t)} < n^{(t)}. \end{cases} \quad (1)$$

Total amount of computing resources reserved by automotive company
 price for reservation
 price for real-time request
 Practical needed amount of computing resources for edge resources

Total waste

$$W^{(t)} = \begin{cases} (m^{(t)} - n^{(t)}) \times P_{rv}, & n^{(t)} \leq m^{(t)}, \\ (n^{(t)} - m^{(t)}) \times (P_{rt} - P_{rv}), & m^{(t)} < n^{(t)}. \end{cases} \quad (2)$$

Objective function can be defined by mean-square-error (MSE)

To minimize the waste $\min \left\| m^{(t)} - n^{(t)} \right\|_2^2. \quad (3)$

- On the first stage, we utilize a deep neural network (named as Decider) to figure out **which algorithm** should be selected according to the experiences.

meta-feature vector

$$\min_i \mathcal{L}_1(f_{\theta_i}) = \|f_{\theta_i}(\mathbf{c}) - r\|_2^2, \quad (4)$$

s.t. $f_{\theta_i} \in \mathcal{M},$
 $\mathbf{c} \in \mathcal{C},$

suggested algorithm
 (the best performance one according to previous experiences)

Basically, each algorithm in \mathcal{M} can be denoted as f_{θ_i} , where f is the function that can represent the i^{th} algorithm in \mathcal{M} and θ describes the corresponding parameters determined by the machine learning model configurations.

- On the second stage, the chosen machine learning model (named as Prognosticator) is implemented to perform the inference for edge resource consumption.

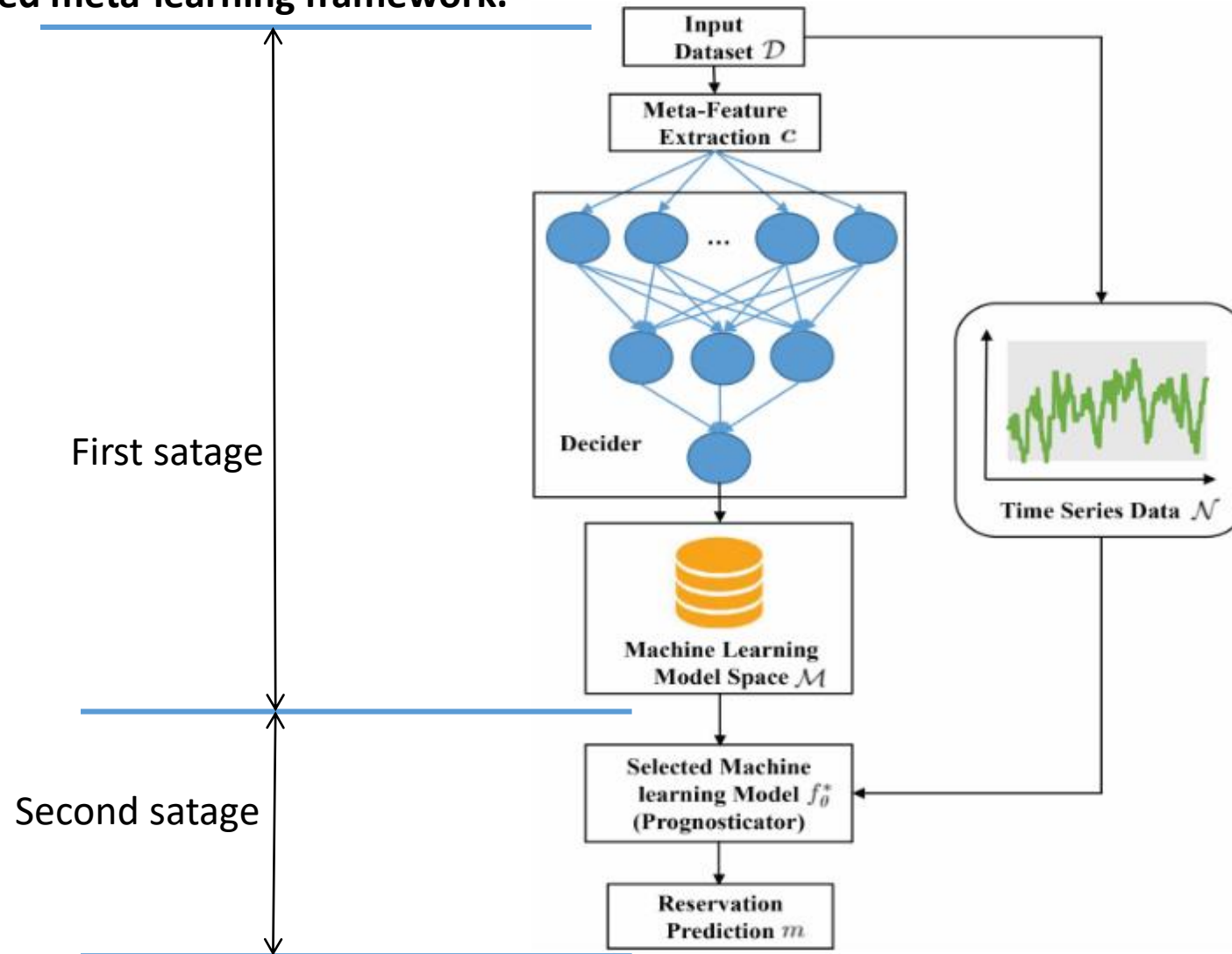
Training

Testing

$$\min_{\phi} \mathcal{L}_2(f_{\theta}^*(\phi)) = \sum_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{T}_i} \|f_{\theta}^*(\phi; \mathbf{x}) - \mathbf{y}\|_2^2, \quad (7)$$

Parameters

The proposed meta-learning framework.



Dawei Chen, Yin-Chen Liu, BaekGyu Kim, Jiang Xie, Choong Seon Hong, Zhu Han, "Edge Computing Resources Reservation in Vehicular Networks: A Meta-Learning Approach," IEEE Transactions on Vehicular Technology, Vol.69, Issue 5, pp.5634-5646, May. 2020

Algorithm 1: The First Stage Meta-Learning.

- 1: Input: experience data \mathcal{D} ; meta-feature space \mathcal{C} ; machine learning model space \mathcal{M} ; new task \mathcal{T}_i ; learning rate α ; maximum iteration steps t_{\max} .
 - 2: Randomly initialize the parameter δ . ← weights
 - 3: **for** $t = 1: t_{\max}$ **do**
 - 4: Feed forward propagate all the samples in \mathcal{D} and calculate the MSE by (4);
 - 5: Back propagate MSE through the network by applying (5) and update the values of parameter set δ' via (6);
 - 6: **end for**
 - 7: Output: the trained Decider with optimal parameters δ^* ;
-

$$\frac{\partial \mathcal{L}_1}{\partial \delta^{(l)}} = \sum_l \left(\frac{\partial \mathcal{L}_1}{\partial o^{(l)}} \right) \frac{\partial o^{(l)}}{\partial o^{(l-1)}} \frac{\partial o^{(l-1)}}{\delta^{(l)}}, \quad (5)$$

$$\delta' = \delta - \alpha \nabla_{\delta} \mathcal{L}_1, \quad (6)$$

$$\min_{\phi} \mathcal{L}_2(f_{\theta}^*(\phi)) = \sum_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{T}_i} \|f_{\theta}^*(\phi; \mathbf{x}) - \mathbf{y}\|_2^2, \quad (7)$$

$$\frac{\partial \mathcal{L}_2}{\partial \phi^{(p)}} = \sum_p \left(\frac{\partial \mathcal{L}_2}{\partial o^{(p)}} \right) \frac{\partial o^{(p)}}{\partial o^{(p-1)}} \frac{\partial o^{(p-1)}}{\phi^{(p)}}, \quad (8)$$

$$\phi' = \phi - \beta \nabla_{\phi} \mathcal{L}_2, \quad (9)$$

Algorithm 2: The Second Stage Meta-Learning.

- 1: Input: meta-feature space \mathcal{C} ; machine learning model space \mathcal{M} ; new task \mathcal{T}_i ; new task dataset \mathcal{N} ; learning rate β ; maximum iteration steps k_{\max} .
 - 2: Feed the meta-features \mathbf{c}_i from \mathcal{T}_i into the Decider and the specific Prognosticator, i.e., f_{θ}^* , can be obtained;
 - 3: Divide \mathcal{N} into training set \mathbf{x} and testing set \mathbf{y} ;
 - 4: Feed \mathbf{x} into f_{θ}^* with randomly initialized parameters ϕ ;
 - 5: **for** $k = 1: k_{\max}$ **do**
 - 6: Feed forward propagate all the samples through \mathbf{x} and get the MSE by (7);
 - 7: Back propagate MSE throughout the network f_{θ}^* by applying (8) and update the parameter set ϕ' via (9);
 - 8: **end for**
 - 9: Output: the Prognosticator f_{θ}^* with optimal parameters ϕ^* ;
 - 10: Feed the testing data \mathbf{y} into $f_{\theta}^*(\phi^*)$ and infer the amount of edge resource consumption;
-

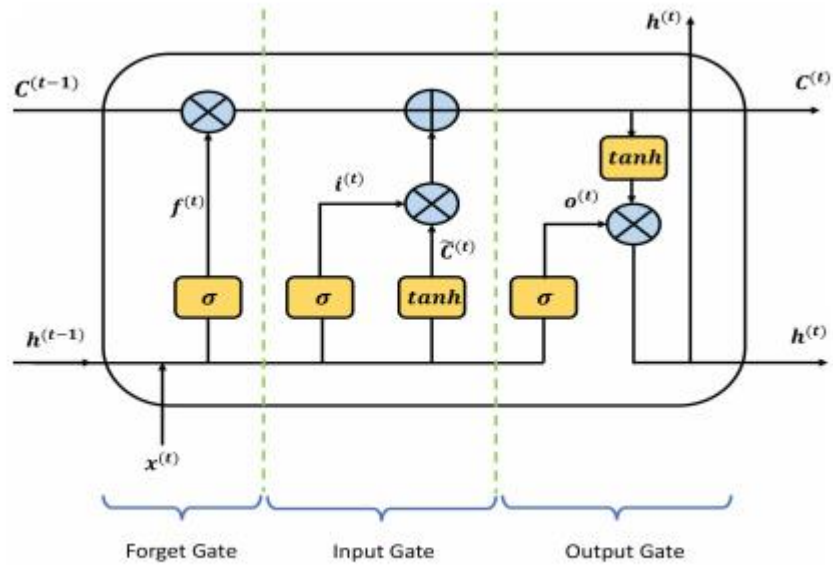


Fig. 3. The structure of a LSTM cell.

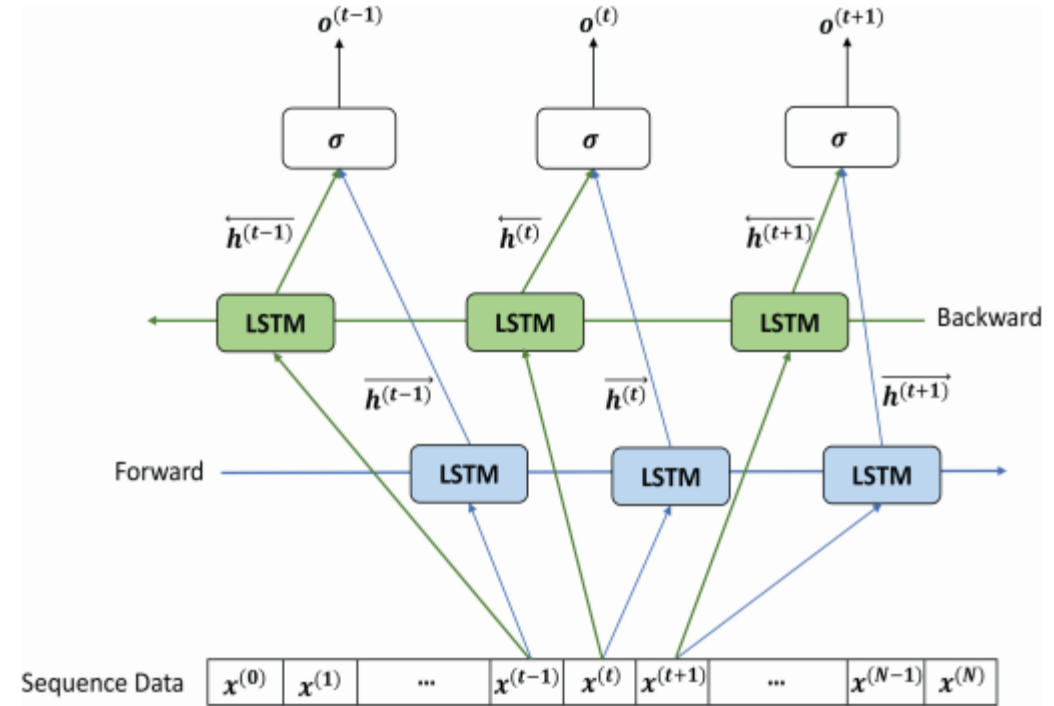


Fig. 4. The architecture of BiLSTM network.

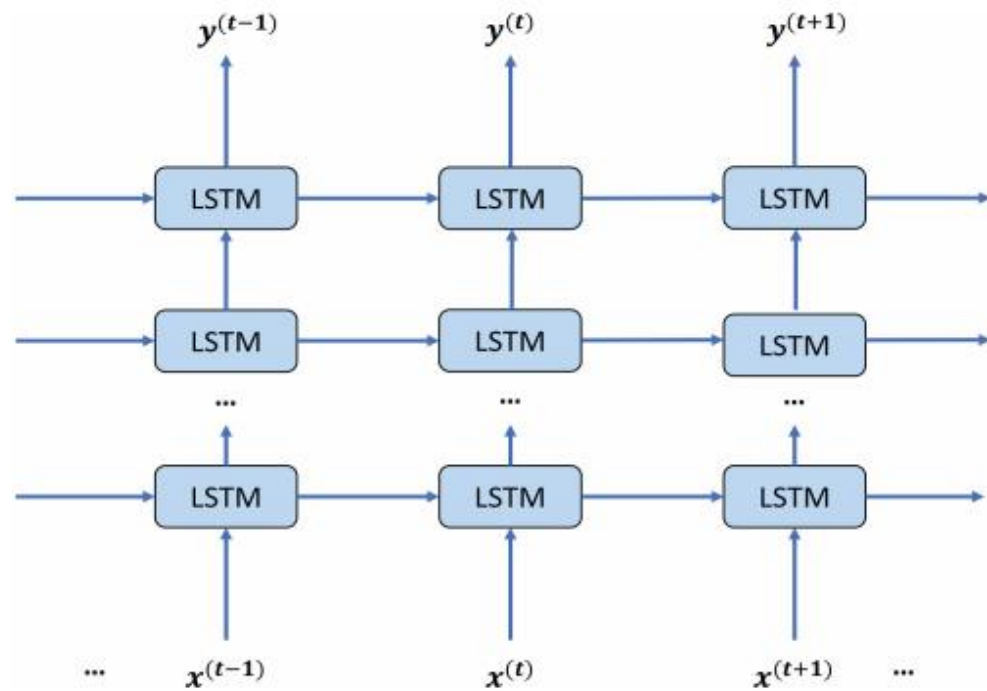


Fig. 5. The structure of a stacked LSTM network.

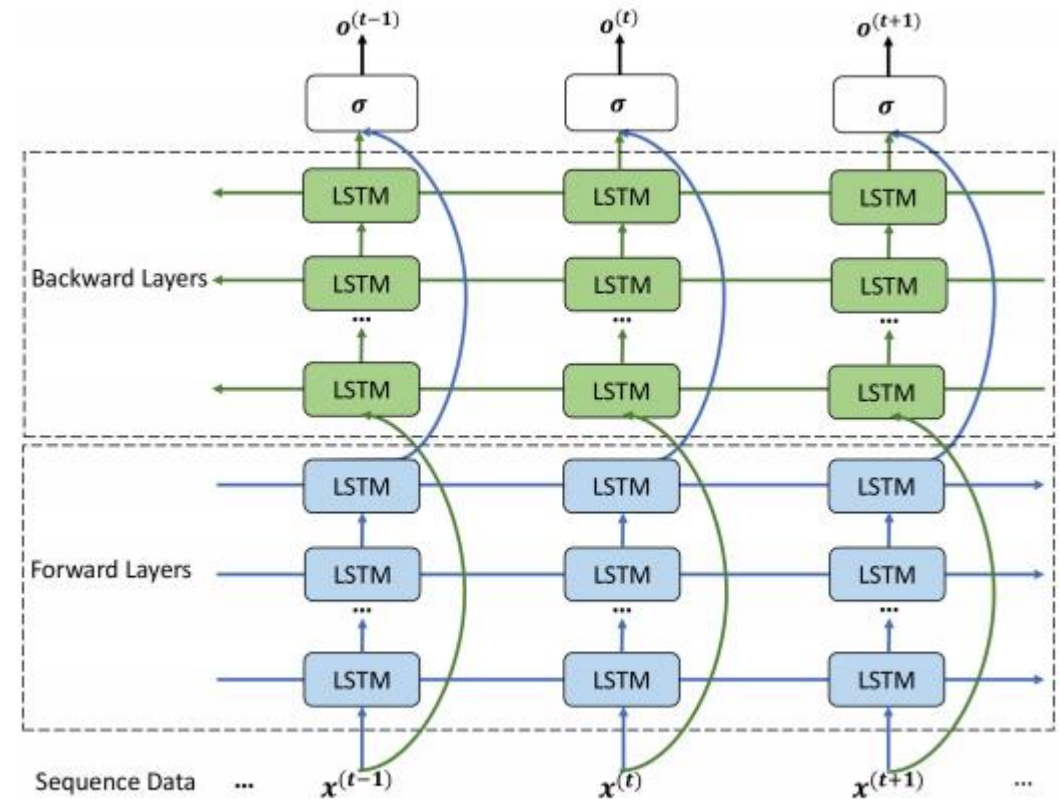
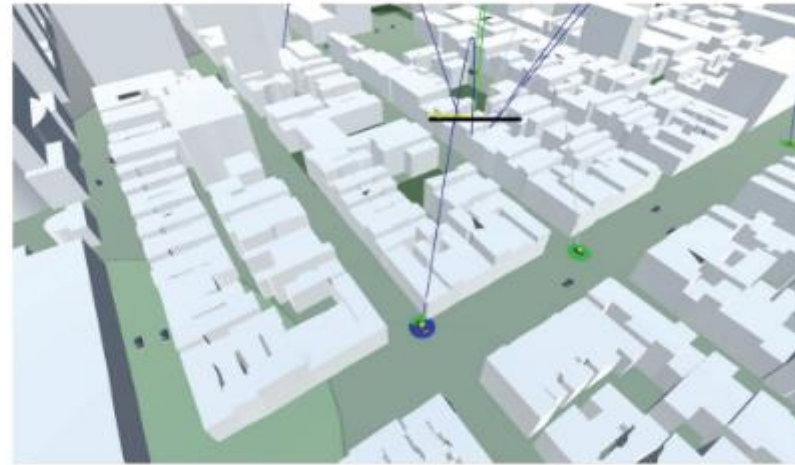


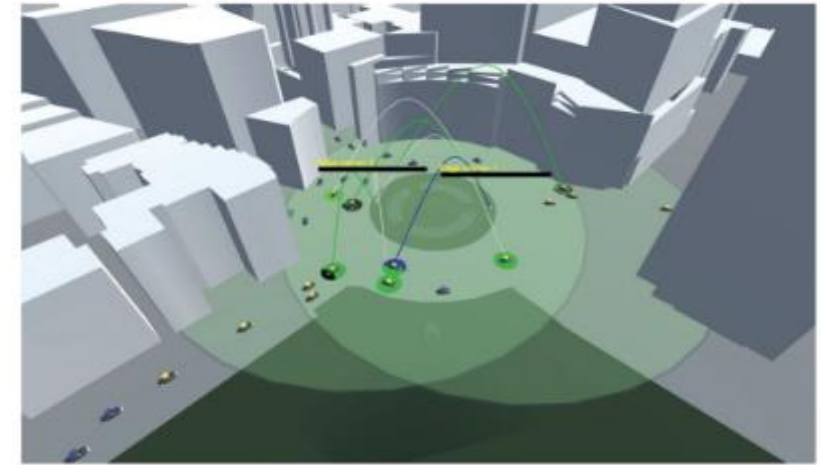
Fig. 6. The structure of a Stacked BiLSTM network.

PARAMETERS SETTINGS FOR \mathcal{M}

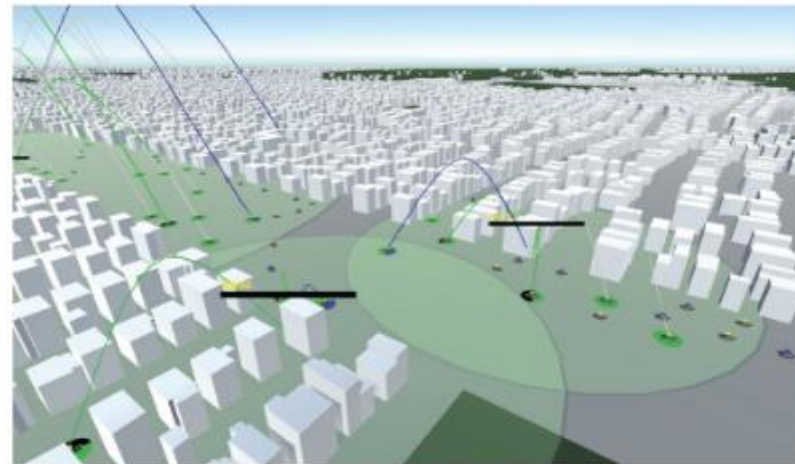
Models	Hidden Layer	Units	Dropout Percentage
LSTM	1	64	-
BiLSTM	1	64	-
S-LSTM	3	32-32-32	-
S-BiLSTM	2	32-32	-
S-LSTM-D	3	32-64-32	10%
S-BiLSTM-D	2	64-64	10%



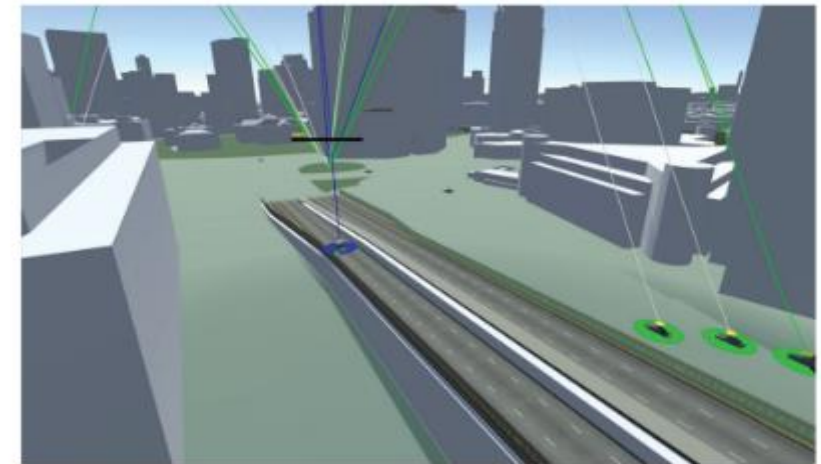
(a)



(b)



(c)



(d)

Fig. 8. The different scenarios built in Unity. (a) The multi-intersection scenario. (b) The roundabout scenario. (c) The highway scenario. (d) The bridge scenario.

SCORES OBTAINED BY PROPOSED META METHOD AND NON-META METHODS

Roadmap	Multi-Intersections			Roundabout			Highway			Bridge		
Model	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH	RMSE	MAPE	MEGH
Meta-learning	37.91	9.13	1.51	19.42	8.56	1.35	37.23	11.64	1.63	46.63	9.31	1.49
LSTM	44.26	9.56	1.68	19.76	9.03	1.37	43.92	12.65	2.72	52.63	10.03	1.72
BiLSTM	69.71	11.78	2.41	38.42	17.14	2.79	44.50	13.57	2.45	74.45	13.67	2.36
S-LSTM	111.28	17.15	3.79	38.71	17.40	2.82	60.79	20.08	3.57	78.98	22.96	2.73
S-BiLSTM	64.16	14.21	2.49	40.70	26.28	3.15	59.26	22.41	2.84	75.99	14.63	2.42
S-LSTM-D	87.57	16.20	2.89	37.87	22.64	2.72	79.20	25.55	3.58	95.27	16.08	3.06
S-BiLSTM-D	62.05	13.14	2.16	38.98	22.79	2.99	64.55	16.32	2.74	93.53	16.56	2.76

RMSE: Root Mean Squared Error

MEPE: Mean Absolute Percentage Error

MEGH : indicating the effective measurement for variety of traffic analysis purpose. The smaller, the better regression of observed flows

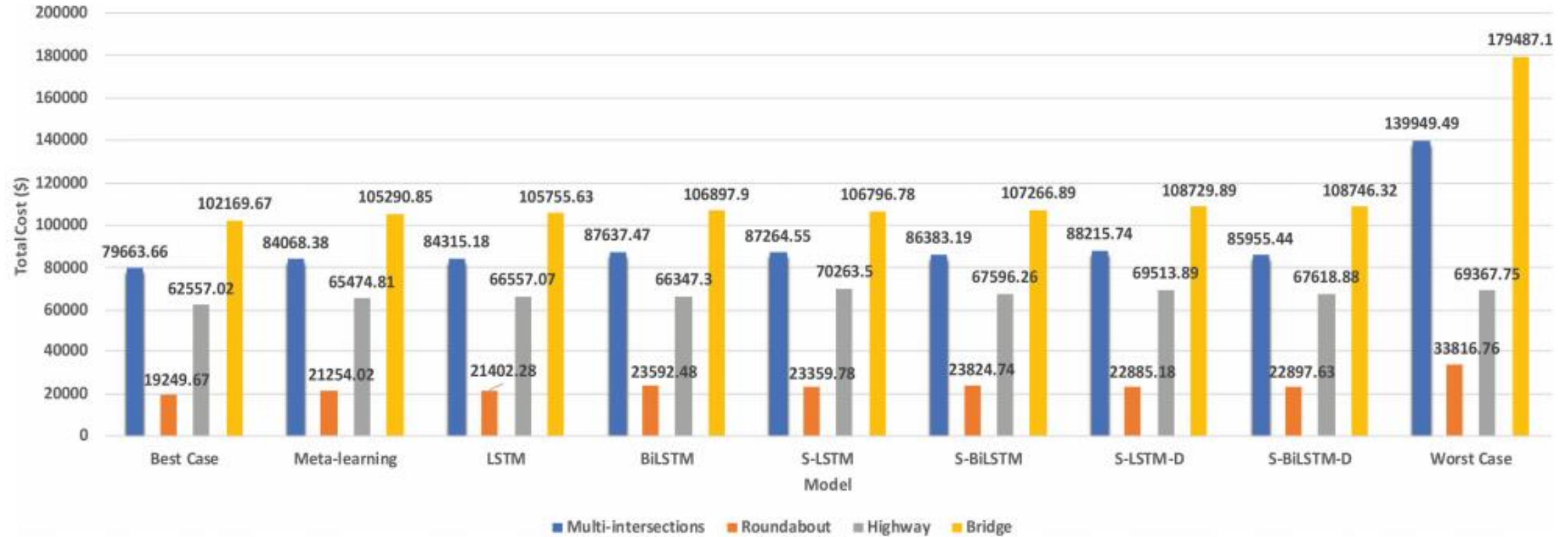
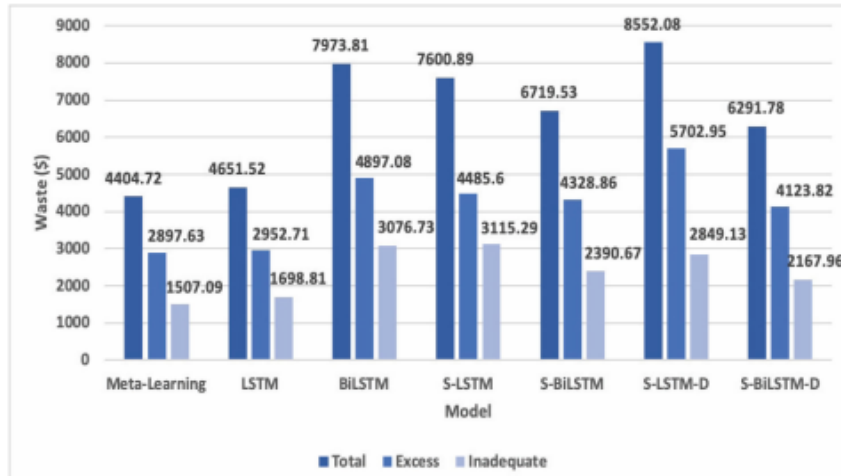
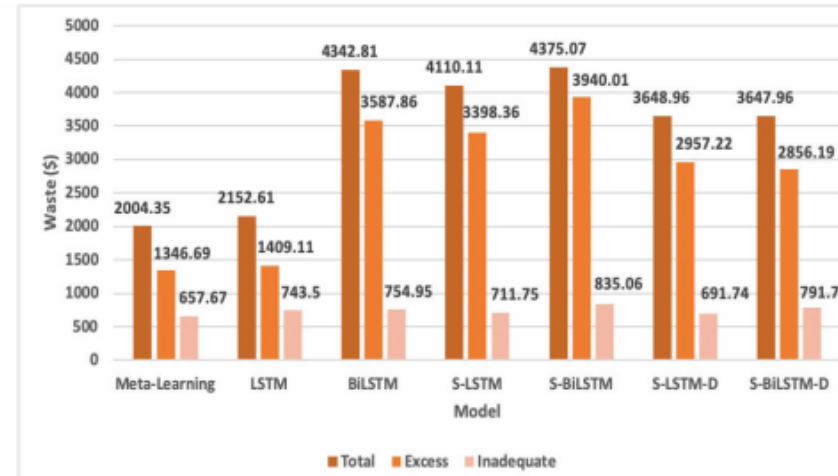


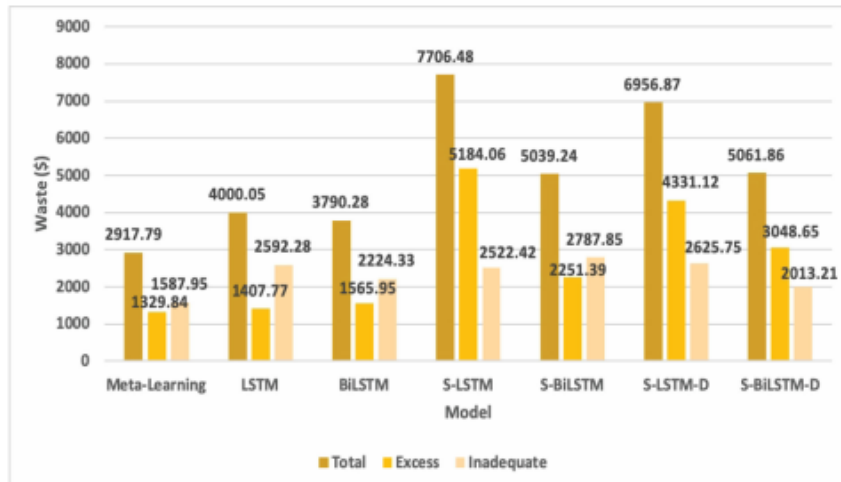
Fig. 9. Total cost for different methods.



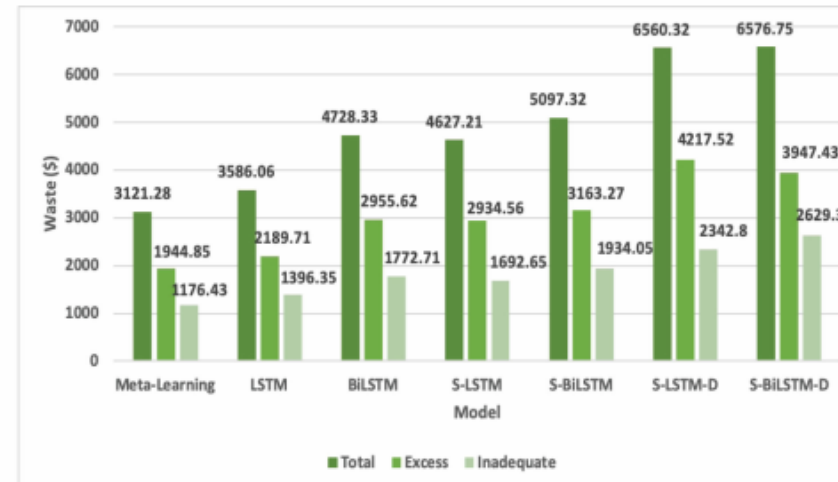
(a)



(b)



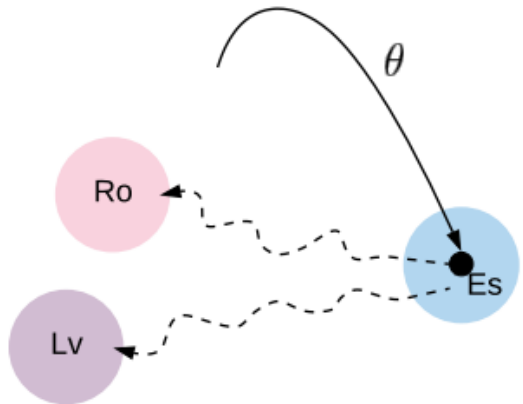
(c)



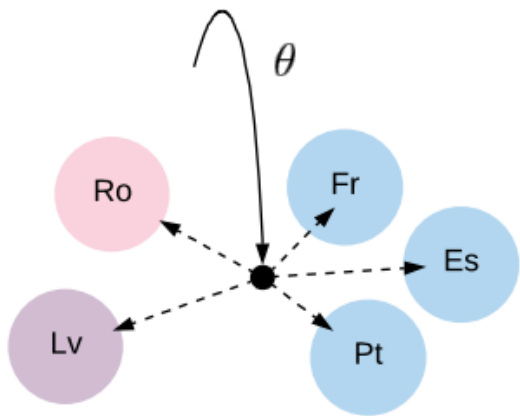
(d)

Fig. 10. Waste for different methods. (a) Waste in multi-intersection scenario. (b) Waste in roundabout scenario. (c) Waste in highway scenario. (d) Waste in bridge scenario.

Conclusion



Transfer Learning



Meta Learning

	Transfer Learning	Meta Learning
Training	Works by per training on large datasets at once	Meta-learning is trained as a multi-task problem . Meta-Learning deals with creating models which either learn and optimize fast or models which generalize and adapt to different tasks easily. Model-based Meta-Learning aims at designing architectures which are inherently capable of generalizing and learning with less data.
Fine-tuning	Fine-tuning requires medium-size datasets (hundreds to thousands) and a lot of training steps	Fine-tuning works for small datasets (tens) and requires only a few training steps (usually 1-3)
Objective	Pretrained weights are usually used as initial parameters for feature extraction during fine-tuning	Meta-training finds initial parameters which requires less training steps for fine-tuning. Memory Networks are mainly used for NLP based tasks. But it can be tuned for any sequence-based tasks. Memory Networks are shallow and compute friendly.

- Meta-learning is still a relatively young area of research and its applications
- Meta-learning can solve AI problem when dataset is small, non-i.i.d., and uncertain
 - ✓ For example,
 - Content caching in edge and mobility-based edge (e.g., vehicle, UAV, etc) network
 - Adaptive computational and communication resource management in wireless network
 - AI-based mobility management for high speed vehicles (e.g., high-speed train, car etc.)
 - Energy management for smart-grid and microgrid domain

Thank You!

Q & A