Thirty-ninth International Conference on Machine Learning

Forget-free Continual Learning with Winning Subnetworks

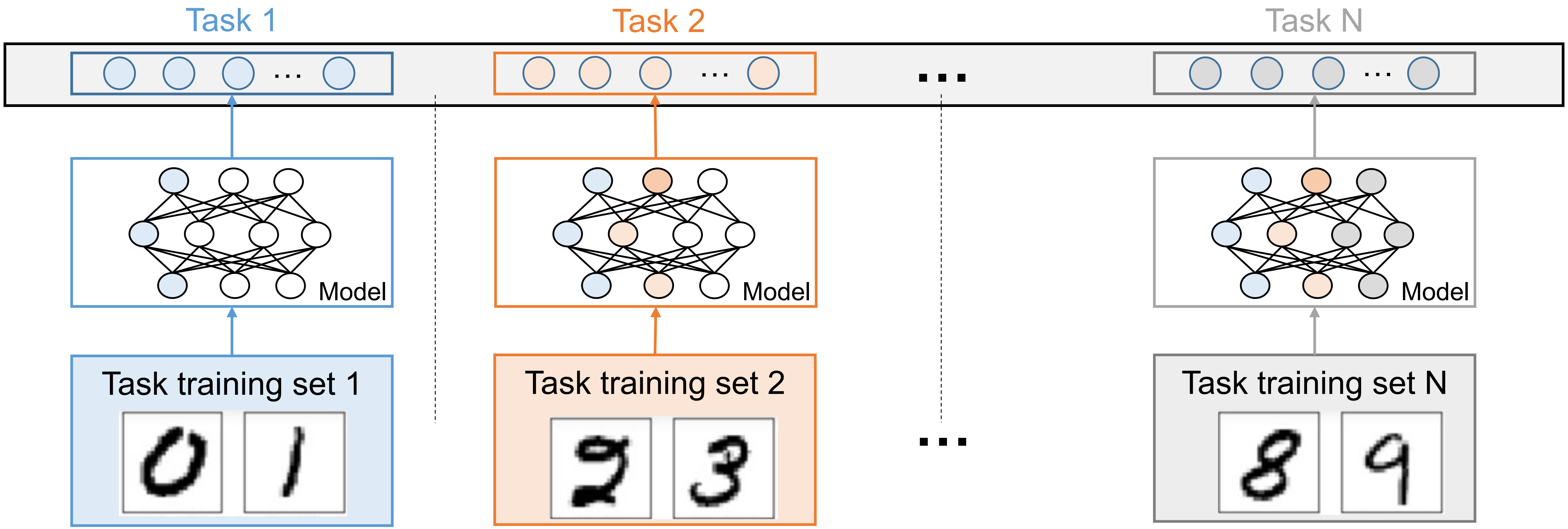
Haeyong Kang[*1], Rusty John Lloyd Mina[*1], Sultan Rizky Hikmawan Madjid[1], Jaehong Yoon[1]
Mark Hasegawa−Johnson[2], Sung Ju Hwang[13], Chang D. Yoo[1],

In Session 3 Track 9
Tue July 19 @ Room 327 – 329

[1]Korea Advanced Institute of Science and Technology (KAIST),
[2]University of Illinois at Urbana-Champaign
[3]AITRICS

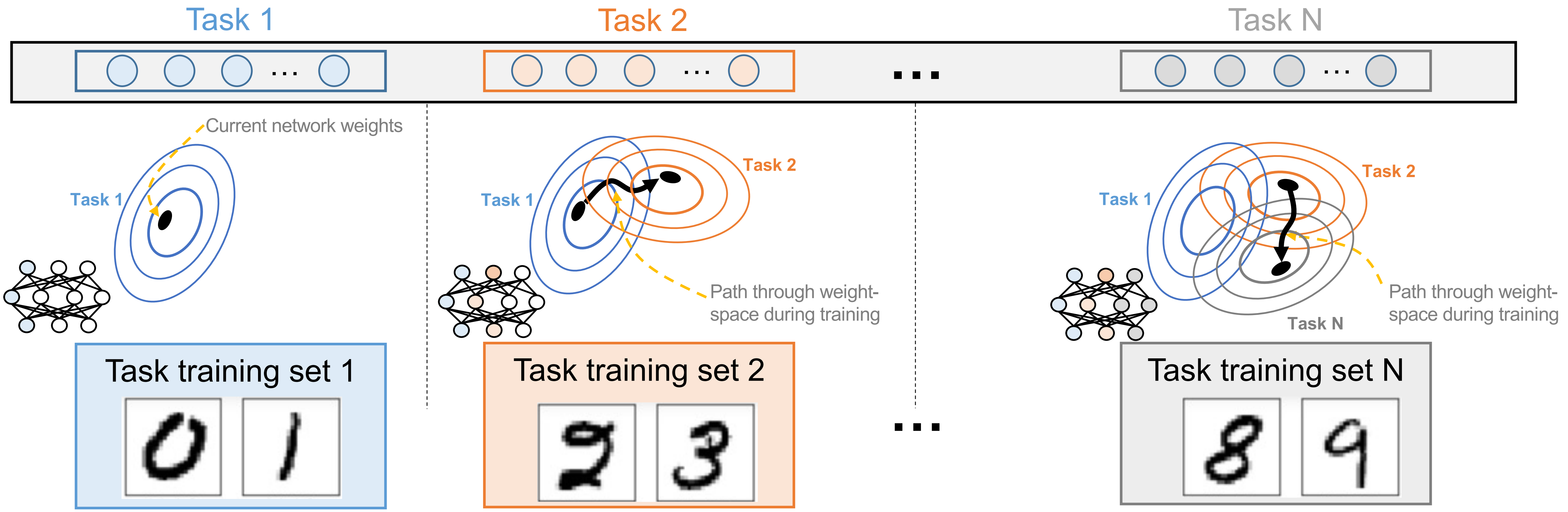
U-AIM

Concept of Continual Learning (CL)



- **Continual learning: a learning paradigm** that allows the model to **learn new tasks on sequence data.**

Catastrophic Forgetting (CF)?



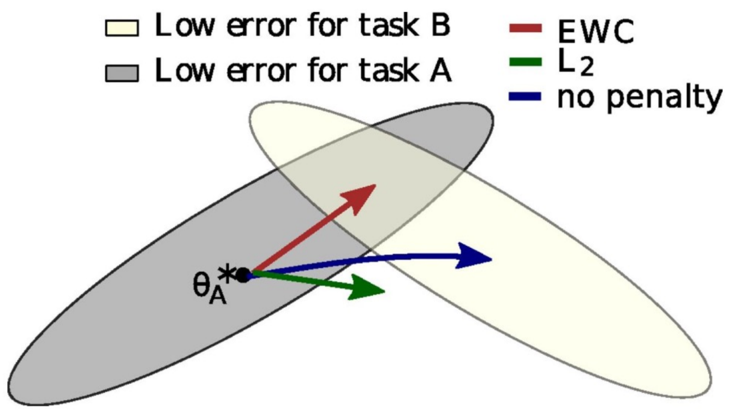
- Catastrophic Forgetting (CF) : **a degradation of performances on previous data.**
- The Objective: **To learn from the new incoming tasks** while retaining knowledge.

Various Approaches for Solving Catastrophic Forgetting (CF)

Various approaches can be broadly categorized as follows:

Regularization-based methods

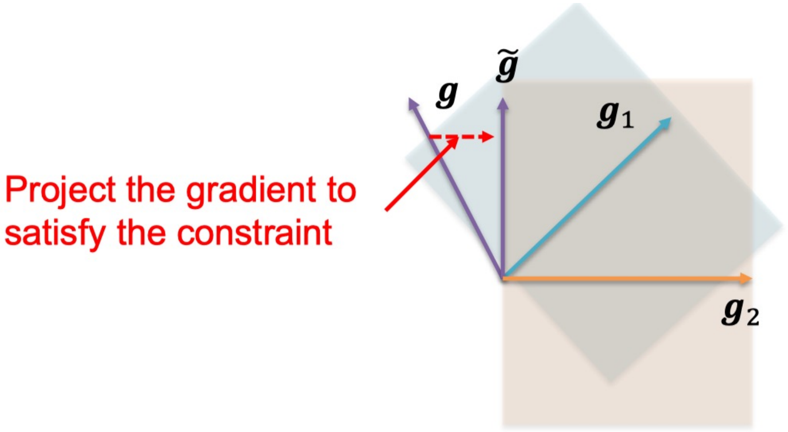
- **Some weights** are crucial for tasks.
- **Preserve task weights**



Elastic Weight Consolidation (EWC)

Rehearsal-based methods

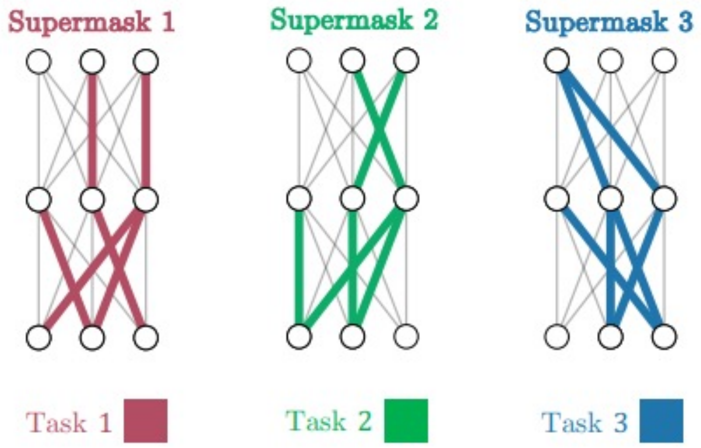
- **Reinforces a model's knowledge** by **replaying samples**.
- **Memory hungry** – Performance scales up with number of samples.



Gradient Episodic Memory (GEM)

Architecture-based methods

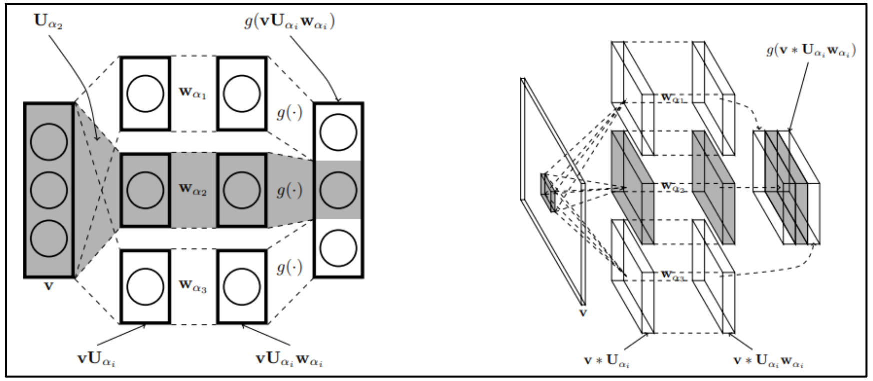
-**Finds task-subnetworks (supermasks) from a dense network**
- **Model capacity scales up** with number of tasks.



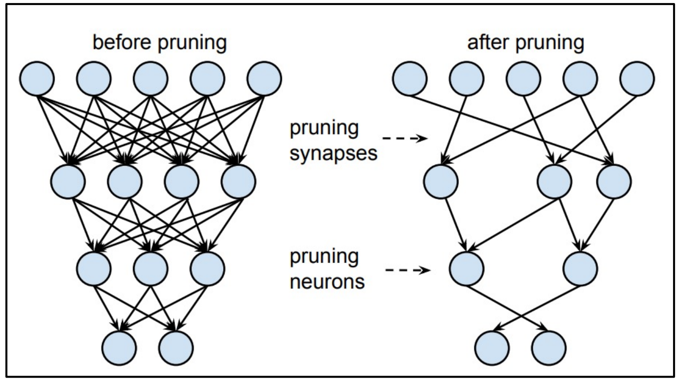
Supermasks in Superposition (SupSup)

How can we build a memory-efficient CL model?

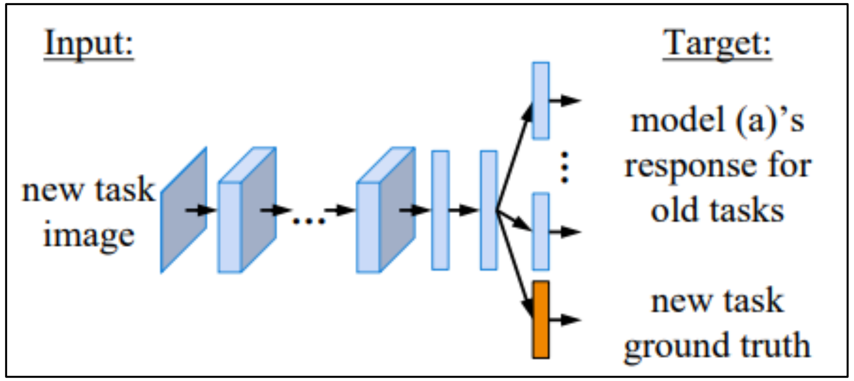
Dense neural networks:
- Over-parameterized (Denil et al.,2013; Han et al., 2016; Li et al., 2016)
- Removing redundant weights can achieve on-par or even better performance than NNs.



(Denil et al.,2013)

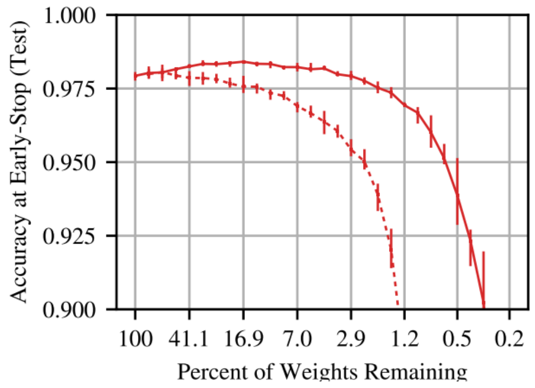


(Han et al., 2016)



(Li et al., 2016)

Lottery Ticket Hypothesis (LTH) (Frankle & Carbin, 2019) :
- **The existence of sparse subnetworks** that preserve the performance of a dense network.
- Searching for optimal winning tickets requires repetitive pruning and retraining.

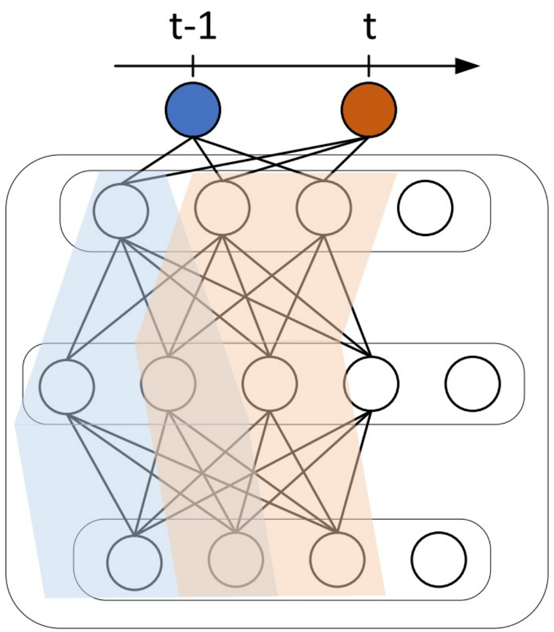


(Frankle & Carbin, 2019)

Architecture-based Continual Learning

ICML International Conference On Machine Learning

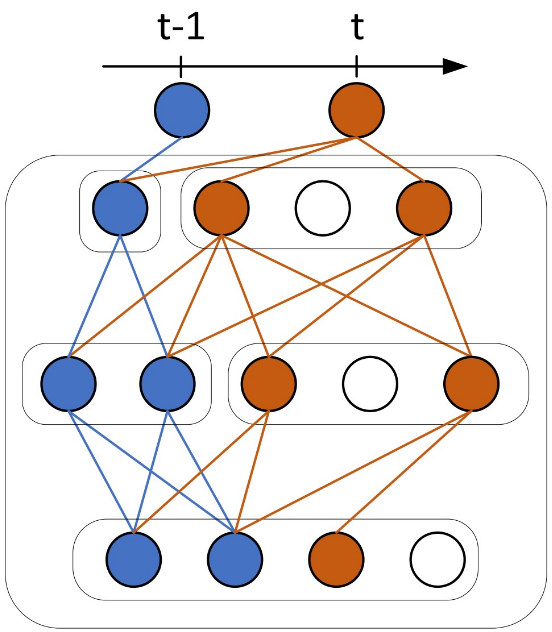
Fixed Backbone



- **Piggyback** (Mallya et al., 2018), and **SupSup** (Wortsman et al., 2020).

- Find the optimal binary mask on a fixed backbone network.

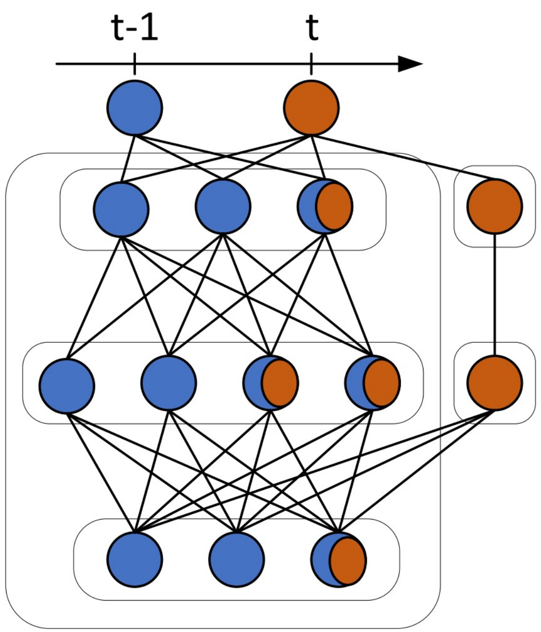
Biased Transfer



- **PackNet** (Mallya & Lazebnik, 2018) and **CLNP** (Golkar et al., 2019).

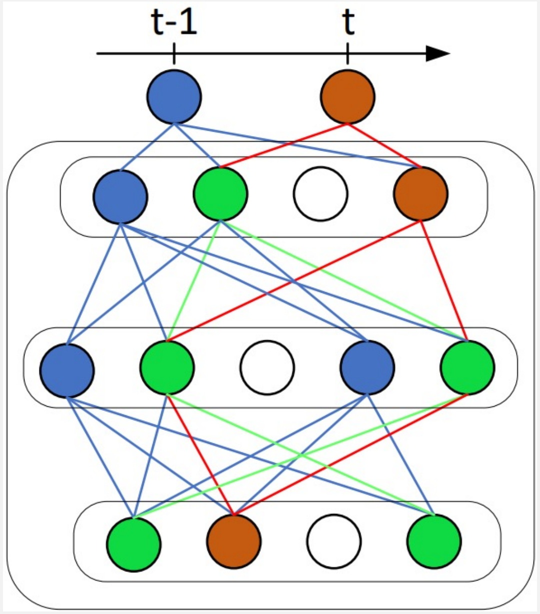
- **Reuse all features** and weights previous which causes biased transfer.

Selective Reuse Expansion beyond Dense Networks



-**APD** (Yoon et al., 2020) **selectively reuse / update** and dynamically expand the dense network.

Winning Sub-Network (WSN)

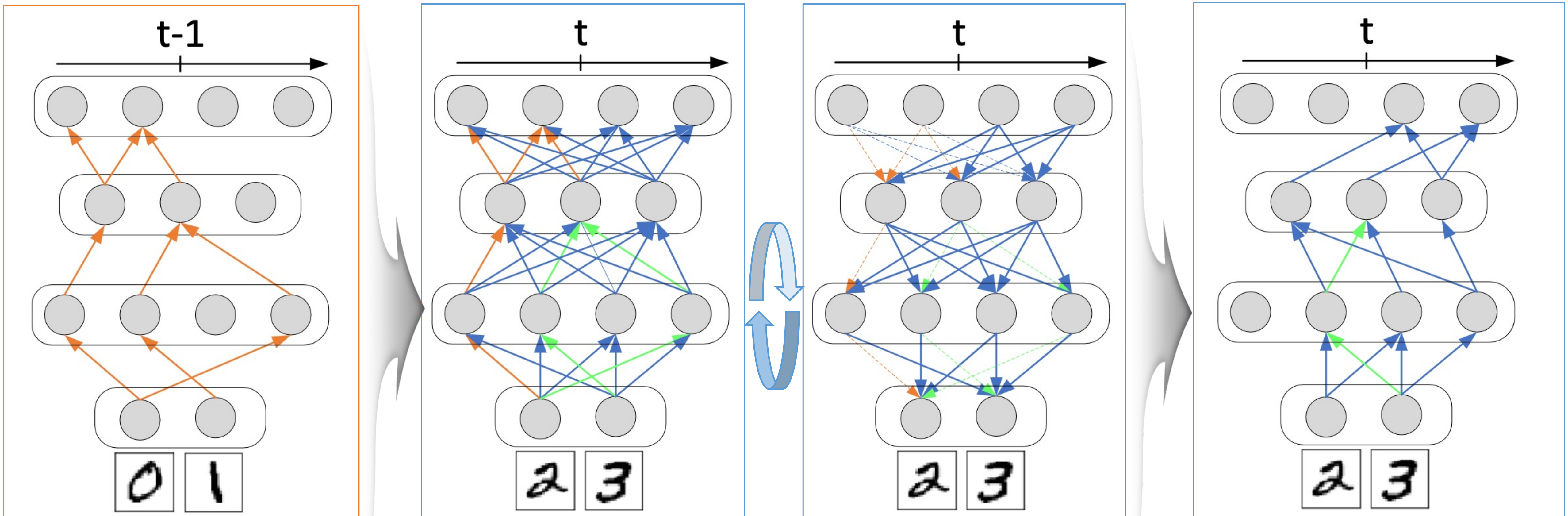


- **Selectively reuse** and **dynamically expand** subnetworks within a dense network.

- Green edges are reused weights.

Illustration of Winning Sub-Networks (WSN) for CL

An illustration of Winning Sub-Networks (WSN):



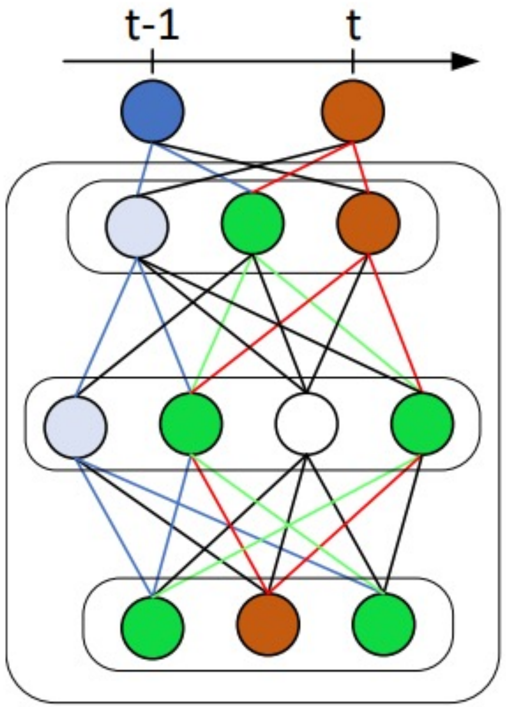
(a) selected weights $\hat{\boldsymbol{\theta}}_{t-1}$ at prior task

(b) forward pass using reused weights

(c) backward pass only on non-used weights

(d) selected weights $\hat{\boldsymbol{\theta}}_{t}$ with subsets of reused weights

Our WSN’s Benefits of reused weights for learning sequence tasks



(d) Selective Reuse Expansion within Network (Our WSN)

(+) Transfer Learning:

To reuse some of the weights from previously chosen weights

(+) Finetuning:

To select new weights from the set of not-yet-chosen weights

(+) Computation Efficiency:

With reused weights learned at $t-1$, WSN selects a few new weights for learning new task t and **learns faster** than others.

Winning Sub-Networks (WSN) Algorithm

Algorithm 1 Winning Subnetworks (WSN)

input $\{\mathcal{D}_t\}_{t=1}^{\mathcal{T}}$, model weights $\boldsymbol{\theta}$, score weights $\mathbf{s}$, binary mask $\mathbf{M}_0 = \mathbf{0}^{|\boldsymbol{\theta}|}$, layer-wise capacity c

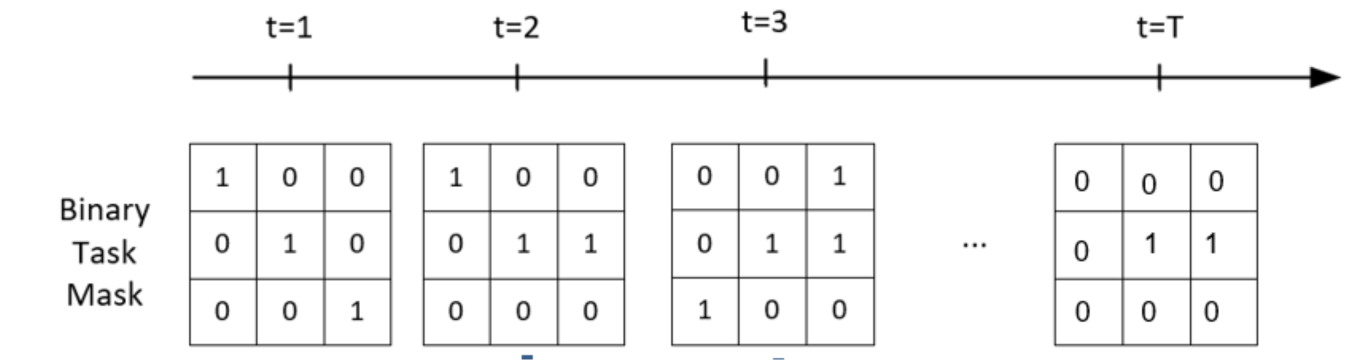
1: Randomly initialize $\boldsymbol{\theta}$ and $\mathbf{s}$.
2: **for** task $t = 1, ..., \mathcal{T}$ **do**
3:     **for** batch $\mathbf{b}_t \sim \mathcal{D}_t$ **do**
4:         Obtain mask $\mathbf{m}_t$ of the top-$c\%$ weights at each layer
5:         Compute $\mathcal{L}\left(\boldsymbol{\theta} \odot \mathbf{m}_t; \mathbf{b}_t\right)$
6:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \odot (\mathbf{1} - \mathbf{M}_{t-1})\right)$ ▷ Weight update
7:         $\mathbf{s} \leftarrow \mathbf{s} - \eta(\frac{\partial \mathcal{L}}{\partial \mathbf{s}})$ ▷ Weight score update
8:     **end for**
9:     $\mathbf{M}_t \leftarrow \mathbf{M}_{t-1} \vee \mathbf{m}_t$ ▷ Accumulate binary mask
10: **end for**

Issues:

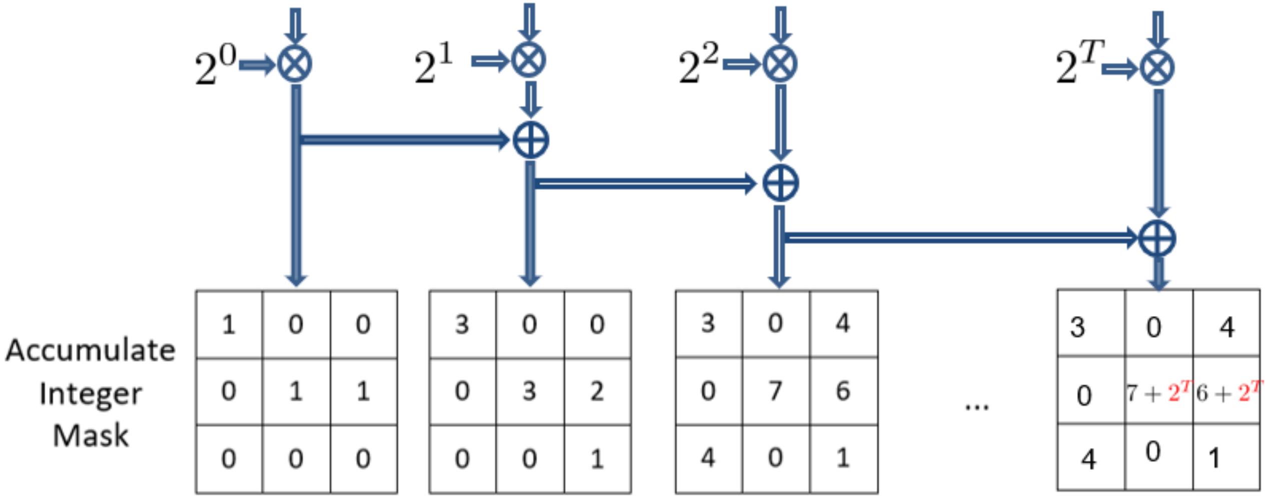
As the number of tasks increases,
→ the number of binary masks to save also increase

Huffman Encoding of Accumulated Integer Mask

- An acquired per task mask (subnetwork)



- STEP 1: Encoding bit stream masks into integer masks.



- STEP2: Convert the integer into ASCII code symbols
- STEP3: N-bit-wise Huffman coding

※ ASCII code symbol

DEC	OCT	HEX	BIN	Symbol
32	040	20	00100000	
33	041	21	00100001	!
34	042	22	00100010	"
35	043	23	00100011	#
36	044	24	00100100	$
37	045	25	00100101	%
38	046	26	00100110	&

U-AIM

Experiments : Datasets with Task Info. and Architectures

ICML International Conference On Machine Learning

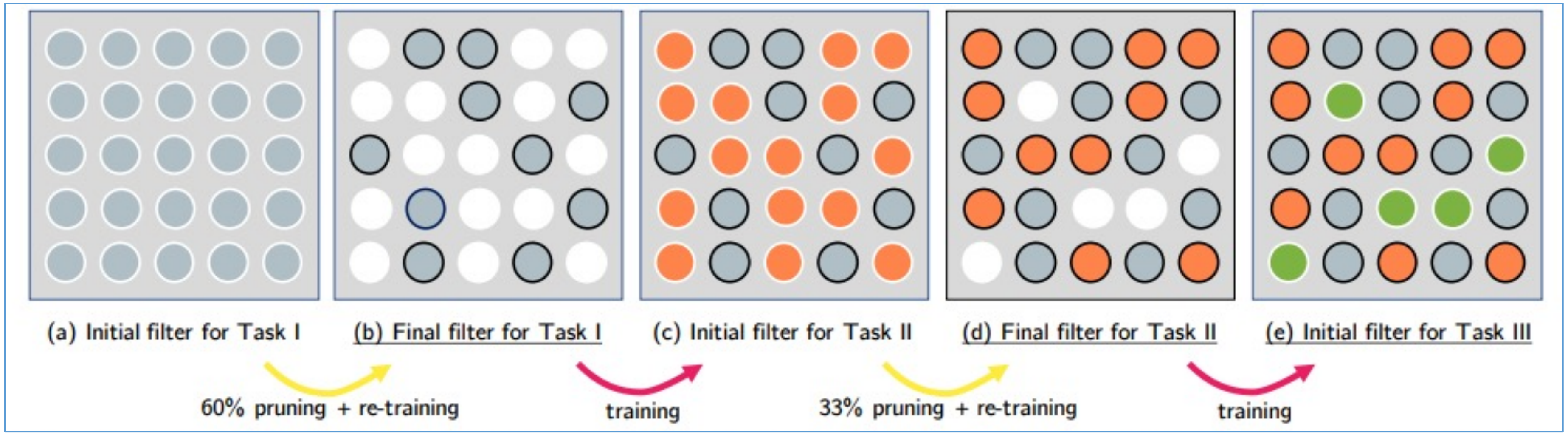
Datasets	CL Task Information	Tasks / classes	Architectures
Permuted MNIST(PMNIST)	A variant of MNIST (LeCun, 1998) where each task has a deterministic permutation to the input image pixels.	**10 / 10**	**Two-layered MLP with 100-100 neurons**
5-Datasets	A mixture of 5 different vision datasets (Saha et al., 2021): CIFAR-10 (Krizhevsky, 2009), MNIST (LeCun, 1998), SVHN (Netzer et al., 2011), FashionMNIST (Xiao et al., 2017), and notMNIST (Bulatov, 2011).	**5 / 10**	**Reduced ResNet18**
Omniglot Rotation	An OCR images datasets, composed of 100 tasks as of each includes 12 classes. We further preprocess and the raw images by generating their rotated version in 90◦ , 180◦ , and 270◦ , followed by Yoon et al. (2020).	**100 / 12**	**LeNet with 64-128-2500-1500 neurons**
CIFAR-100 Split	A visual object dataset, constructed by randomly dividing 100 classes of CIFAR-100 into 10 tasks with 10 classes per task.	**10 / 10**	**AlexNet**
CIFAR-100 Superclass	We follow the setting from Yoon et al. (2020) that divides CIFAR-100 dataset into 20 tasks according to the 20 superclasses, and each superclass contains 5 different but semantically related classes.	**20 / 20**	**LeNet with 64-128-2500-1500 neurons**
TinyImageNet	A variant of ImageNet (Krizhevsky et al., 2012) containing 40 of 5-way classification tasks with the image sized by 64 × 64 × 3.	**40 / 5**	**4 Conv layers and 3 Fully connected layers**

Baselines

Baselines	Information
STL	Single-task learning, not a CL method
FINETUNE	Naïve sequential training
EWC	Regularization-based methods
HAT	Regularization-based methods
GPM	Rehearsal-based methods
FS-DGPM	Rehearsal-based methods
PackNet	Architecture-based methods
SupSup	Architecture-based methods
Multitask	Trains on multiple tasks simultaneously, not a CL method

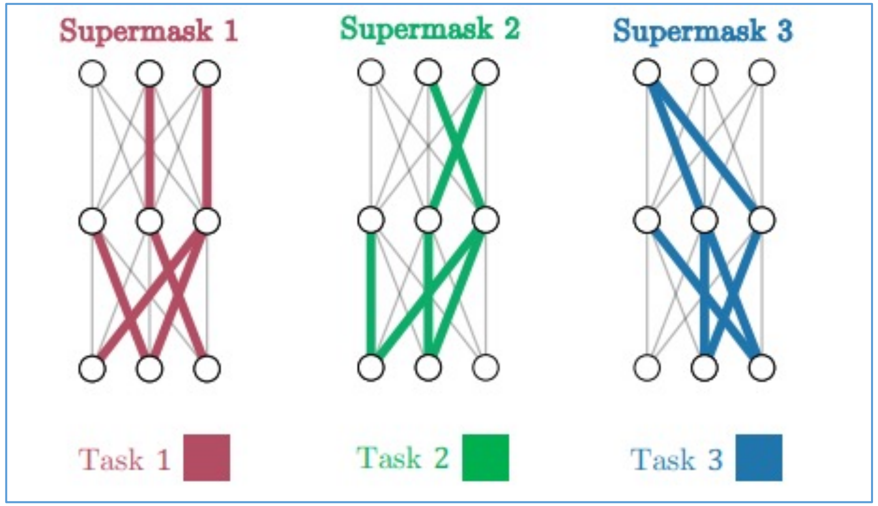
Baselines: PackNet, SupSup, and WSN (ours)

PackNet



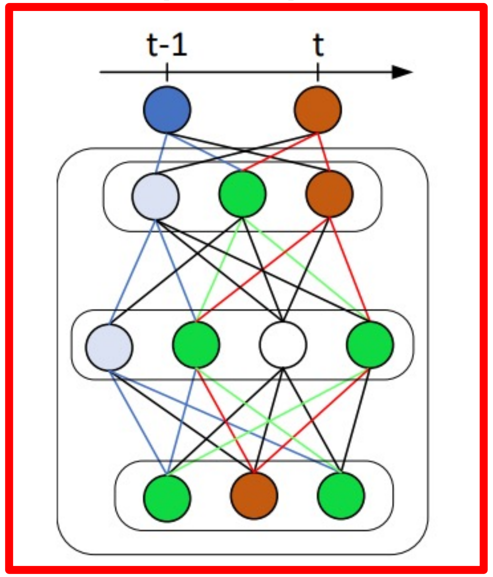
- Reused all weights
- Select weights on the absolute of mask values.

SupSup



- Each weight score (subnetwork) for each task

WSN (ours)



- Selective reused weights
- Single weight score

Performance comparisons of WSN and Baselines (1)

Table 1. **Performance comparison of the WSN and baselines** on various benchmark datasets.
- Accuracy (ACC),
- Average capacity (CAP),
- Average backward transfer (BWT)

Values with † and $*$ denote reported performances from (Saha et al., 2021) and (Yoon et al., 2020).

Method	Permuted MNIST			5 Datasets			Omniglot Rotation		
	ACC (%)	CAP (%)	BWT	ACC (%)	CAP (%)	BWT	ACC (%)	CAP (%)	BWT
STL	97.37 (± 0.01)	1,000.0	-	93.44 (± 0.12)	500.0	-	82.13 (± 0.08)*	10,000.0	-
FINETUNE	78.22 (± 0.84)	100.0	-0.21 (± 0.01)	80.06 (± 0.74)	100.0	-0.17 (± 0.01)	44.48 (± 1.68)	100.0	-0.45 (± 0.02)
EWC (Kirkpatrick et al., 2017)	92.01 (± 0.56)	100.0	-0.03 (± 0.00)	88.64 (± 0.26)†	100.0†	-0.04 (± 0.01)†	68.66 (± 1.92)*	100.0*	-
HAT (Serrà et al., 2018)	-	-	-	91.32 (± 0.18)†	100.0†	-0.03 (± 0.00)†	-	-	-
GPM (Saha et al., 2021)	94.96 (± 0.07)	100.0	-0.02 (± 0.01)	91.22 (± 0.20)†	100.0	-0.01 (± 0.00)†	85.24 (± 0.37)	100.0	-0.01 (± 0.00)
PackNet (Mallya & Lazebnik, 2018)	96.37 (± 0.04)	96.38	**0.0**	92.81 (± 0.12)	82.86	**0.0**	30.70 (± 1.50)	399.2	**0.0**
SupSup (Wortsman et al., 2020)	96.31 (± 0.09)	122.89 (± 0.07)	**0.0**	93.28 (± 0.21)	104.27 (± 0.21)	**0.0**	58.14 (± 2.42)	407.12 (± 0.17)	**0.0**
WSN, $c = 0.03$	94.84 (± 0.11)	**19.87 (± 0.16)**	**0.0**	90.57 (± 0.65)	**12.11 (± 0.06)**	**0.0**	80.68 (± 2.60)	**75.87 (± 1.24)**	**0.0**
WSN, $c = 0.05$	95.65 (± 0.03)	26.49 (± 0.16)	**0.0**	91.61 (± 0.21)	17.26 (± 0.25)	**0.0**	**87.28 (± 0.72)**	79.85 (± 1.19)	**0.0**
WSN, $c = 0.1$	96.14 (± 0.03)	40.41 (± 0.54)	**0.0**	92.67 (± 0.12)	28.01 (± 0.28)	**0.0**	83.10 (± 1.56)	83.08 (± 1.61)	**0.0**
WSN, $c = 0.3$	**96.41 (± 0.07)**	77.73 (± 0.36)	**0.0**	93.22 (± 0.32)	62.30 (± 0.69)	**0.0**	81.89 (± 1.15)	102.2 (± 0.89)	**0.0**
WSN, $c = 0.5$	96.24 (± 0.11)	98.10 (± 0.25)	**0.0**	**93.41 (± 0.13)**	86.10 (± 0.57)	**0.0**	79.80 (± 2.16)	121.2 (± 0.50)	**0.0**
MTL	96.70 (± 0.02)†	100.0	-	91.54 (± 0.28)†	100.0	-	81.23 (± 0.52)	100.0	-

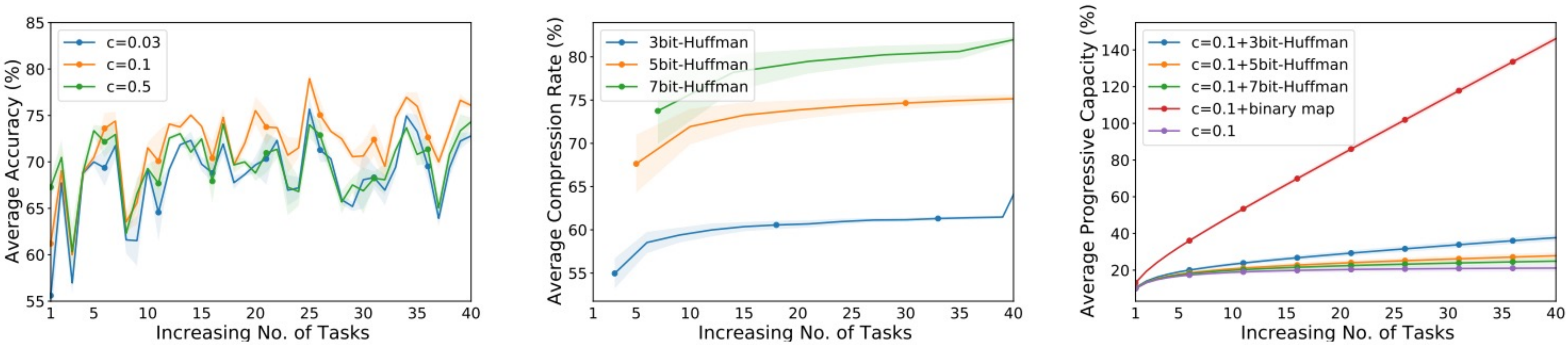
Performance comparisons of WSN and Baselines (2)

Table 2. **Performance comparisons of the WSN and other state-of-the-art** including baselines:
- Average accuracy (ACC)
- Average capacity (CAP),
- Average backward transfer (BWT)

† denotes results reported from Deng et al. (2021).

Method	CIFAR-100 Split			CIFAR-100 Superclass			TinyImageNet		
	ACC (%)	CAP (%)	BWT (%)	ACC (%)	CAP (%)	BWT (%)	ACC (%)	CAP (%)	BWT (%)
EWC (Kirkpatrick et al., 2017)	72.77 (± 0.45)†	100.0	-3.59 (± 0.55)†	50.26 (± 1.48)†	100.0	-7.87 (± 1.63)†	-	-	-
GEM (Lopez-Paz & Ranzato, 2017)	70.15 (± 0.34)†	100.0	-8.61 (± 0.42)†	50.35 (± 0.80)†	100.0	-9.50 (± 0.85)†	50.57 (± 0.61)*	100.0	-20.50 (± 0.10)*
ICARL (Rebuffi et al., 2017)	53.50 (± 0.81)†	100.0	-20.44 (± 0.82)†	49.05 (± 0.51)†	100.0	-11.24 (± 0.27)†	54.77 (± 0.32)*	100.0	-3.93 (± 0.55)*
ER (Chaudhry et al., 2019b)	70.07 (± 0.35)†	100.0	-7.70 (± 0.59)†	51.64 (± 1.09)†	100.0	-7.86 (± 0.89)†	48.32 (± 1.51)*	100.0	-19.86 (± 0.70)*
La-MaML (Gupta et al., 2020)	71.37 (± 0.67)†	100.0	-5.39 (± 0.53)†	54.44 (± 1.36)†	100.0	-6.65 (± 0.85)†	66.90 (± 1.65)†	100.0	-9.13 (± 0.90)†
GPM (Saha et al., 2021)	73.18 (± 0.52)†	100.0	-1.17 (± 0.27)†	57.33 (± 0.37)†	100.0	-0.37 (± 0.12)†	67.39 (± 0.47)†	100.0	**1.45 (± 0.22)**†
FS-DGPM (Deng et al., 2021)	74.33 (± 0.31)†	100.0	-2.71 (± 0.17)†	58.81 (± 0.34)†	100.0	-2.97 (± 0.35)†	70.41 (± 1.30)†	100.0	-2.11 (± 0.84)†
PackNet (Mallya & Lazebnik, 2018)	72.39 (± 0.37)	96.38 (± 0.00)	**0.0**	58.78 (± 0.52)	126.65 (± 0.00)	**0.0**	55.46 (± 1.22)	188.67 (± 0.00)	**0.0**
SupSup (Wortsman et al., 2020)	75.47 (± 0.30)	129.00 (± 0.03)	**0.0**	61.70 (± 0.31)	162.49 (± 0.00)	**0.0**	59.60 (± 1.05)	214.52 (± 0.89)	**0.0**
WSN, $c = 0.03$	70.65 (± 0.36)	**18.56 (± 0.25)**	**0.0**	54.99 (± 0.71)	**22.30 (± 0.22)**	**0.0**	68.72 (± 1.63)	**37.19 (± 0.21)**	**0.0**
WSN, $c = 0.05$	72.44 (± 0.27)	25.09 (± 0.42)	**0.0**	57.99 (± 1.34)	27.37 (± 0.33)	**0.0**	71.22 (± 0.94)	41.98 (± 0.52)	**0.0**
WSN, $c = 0.1$	74.55 (± 0.47)	39.87 (± 0.62)	**0.0**	60.45 (± 0.37)	38.55 (± 0.20)	**0.0**	**71.96 (± 1.41)**	48.65 (± 3.03)	**0.0**
WSN, $c = 0.3$	75.98 (± 0.68)	80.26 (± 1.53)	**0.0**	61.47 (± 0.30)	63.47 (± 1.33)	**0.0**	70.92 (± 1.37)	73.44 (± 2.35)	**0.0**
WSN, $c = 0.5$	**76.38 (± 0.34)**	99.13 (± 0.48)	**0.0**	**61.79 (± 0.23)**	80.93 (± 1.58)	**0.0**	69.06 (± 0.82)	92.03 (± 1.80)	**0.0**
Multitask	79.75 (± 0.38)†	100.0	-	61.00 (± 0.20)†	100.0	-	77.10 (± 1.06)†	100.0	-

Huffman Encoder Compression Rate & Progressive Capacities



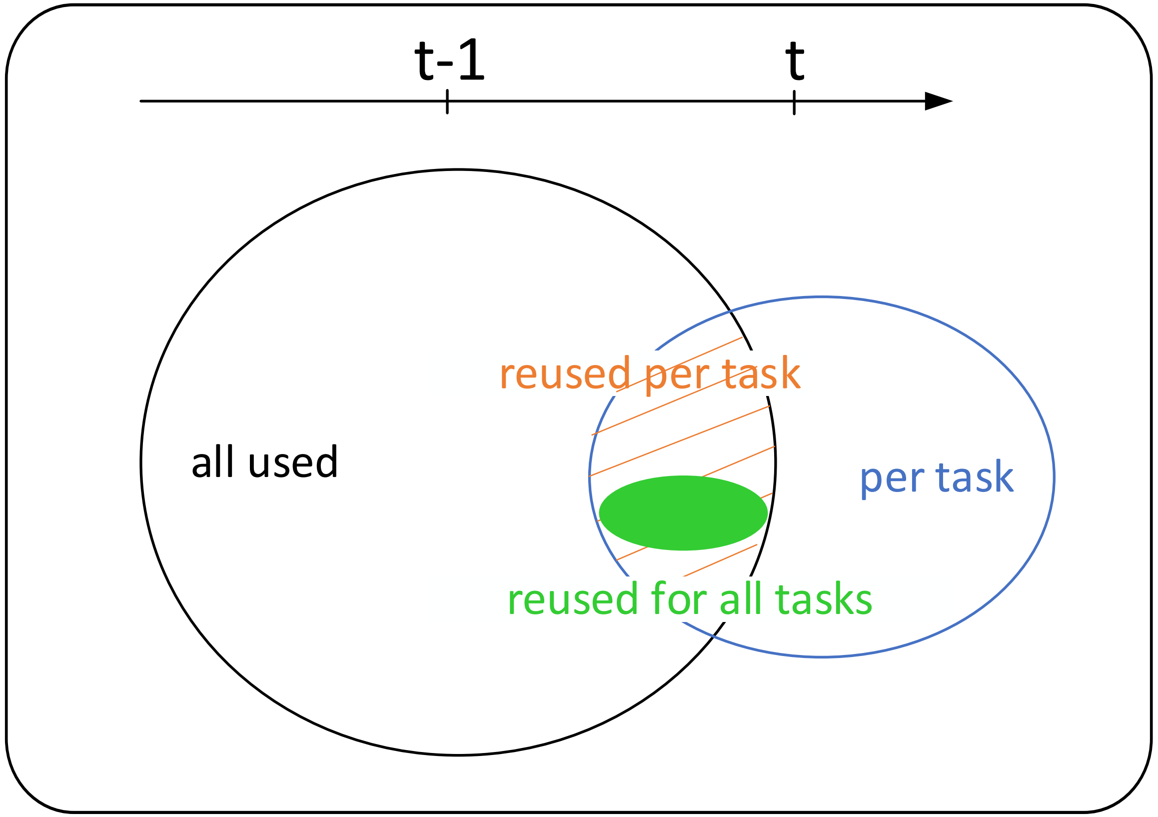
(a) Per Task Accuracy over c (b) N-Bit-wise-Compression Rate (c) Progressive Capacities of Models

Performances and Compressed Capacities - Sequence of TinyImageNet Dataset Experiments.

(a) The c = 0.1 shows generalized performances over others.
(b) With fixed c = 0.1, the bit-wise Huffman compression rate.
(c) The model capacity with the model capacity + the compressed binary masks over varying bits.

→ Within the 40-tasks, the 7-bits compressed capacities are the least increasing along with the c = 0.1 model capacity.

Catastrophic Forgetting From WSN's Viewpoint (1)



A Dense Network

- **All used weights** represents all activated sets of weights up to task t − 1.

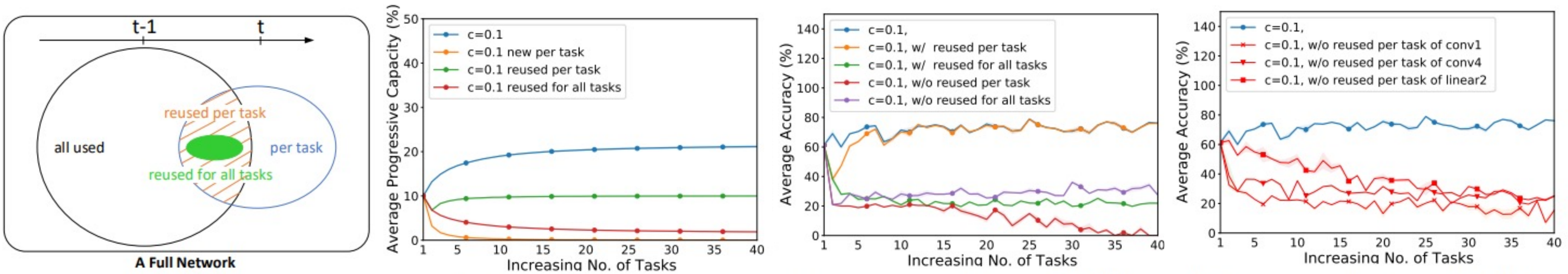
- **Per task** represents an activated set of weights at task t.

- **Reused per task** represents an intersection set of weights per task and reused weights.

- **New per task = Per task - reused per task** represents a new activated set of weights at task t.

- **Reused for all tasks** represents an intersection set of weights reused from task 1 up to task t.

Catastrophic Forgetting From WSN's Viewpoint (2)



(a) A Diagram of Capacities (b) Capacities over c (c) Accuracy of Reused Weights (d) Accuracy w/o Reused Weights

Figure 5. **Layer-wise Analysis on TinyImageNet Dataset Experiments**:

(a) Weights reusability within a dense network,

(b) Capacities except to binary maps are determined by c = 0.1,

(c) The most significant forgetting occurs from weights without reused per task

(d) Performance drops significantly at Conv1 layer.

Conclusions

- **Winning SubNetworks sequentially learns and selects an optimal subnetwork for each task.**

- Specifically, **WSN jointly learns the model weights and task-adaptive binary masks, attempting to select a small set of weights to be activated (winning ticket) by reusing weights.**

- The proposed method is inherently **immune to catastrophic forgetting.**

- **Binary masks were compressed using Huffman coding for a sub-linear increase in network capacity** with respect to the number of tasks.