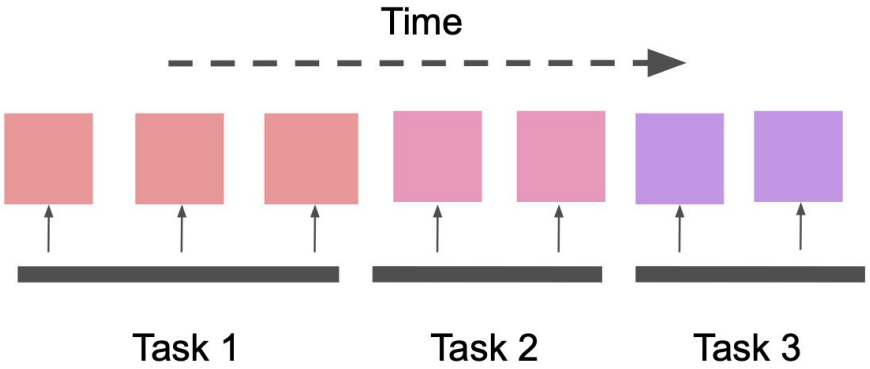
Supermasks in Superposition

Mitchell Wortsman*[1], Vivek Ramanujan*[2], Rosanne Liu[3], Aniruddha Kembhavi[2], Mohammad Rastegari[1], Jason Yosinski[3], Ali Farhadi[1]

[1]University of Washington [2]Allen Institute for AI [3]ML Collective

Presented by Akshata Bhat and Zifan Liu

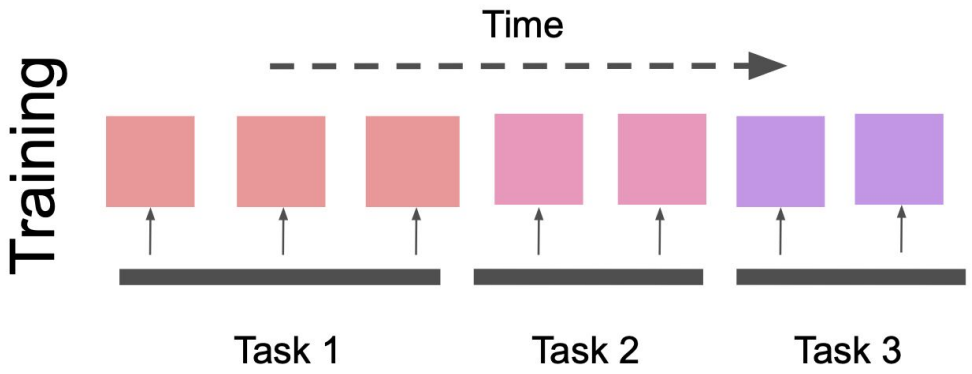
Background

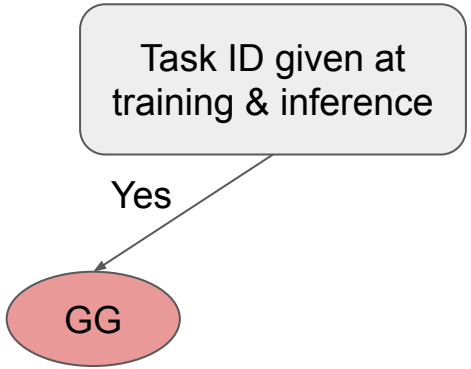


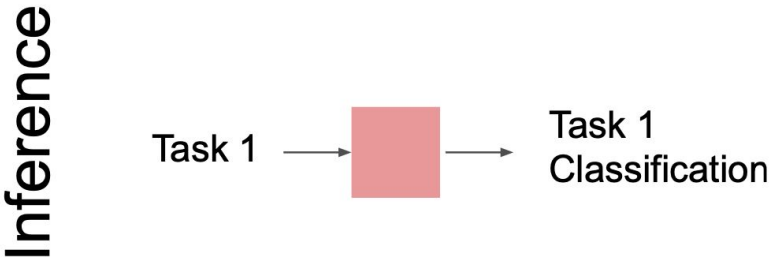
Three-letter taxonomy:

- First letter: If the task ID is given (G) or not (N) at training time.
- Second letter: If the task ID is given (G) or not (N) at inference time.
- Third letter: If the tasks share labels (s) or not (u).

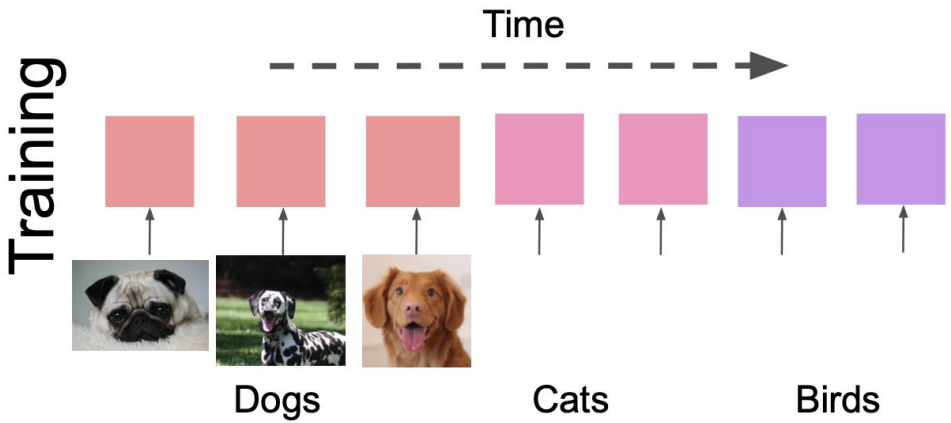
Background

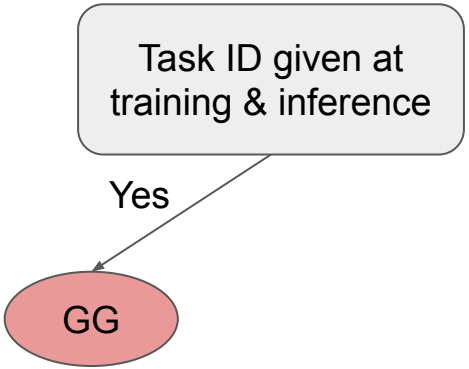

Training
Time
Task 1
Task 2
Task 3

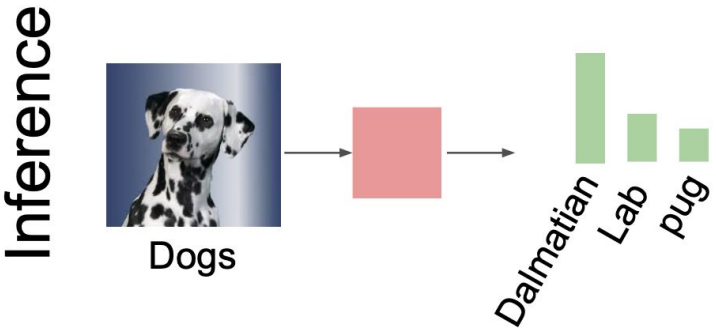

Task ID given at training & inference
Yes
GG


Inference
Task 1
Task 1 Classification

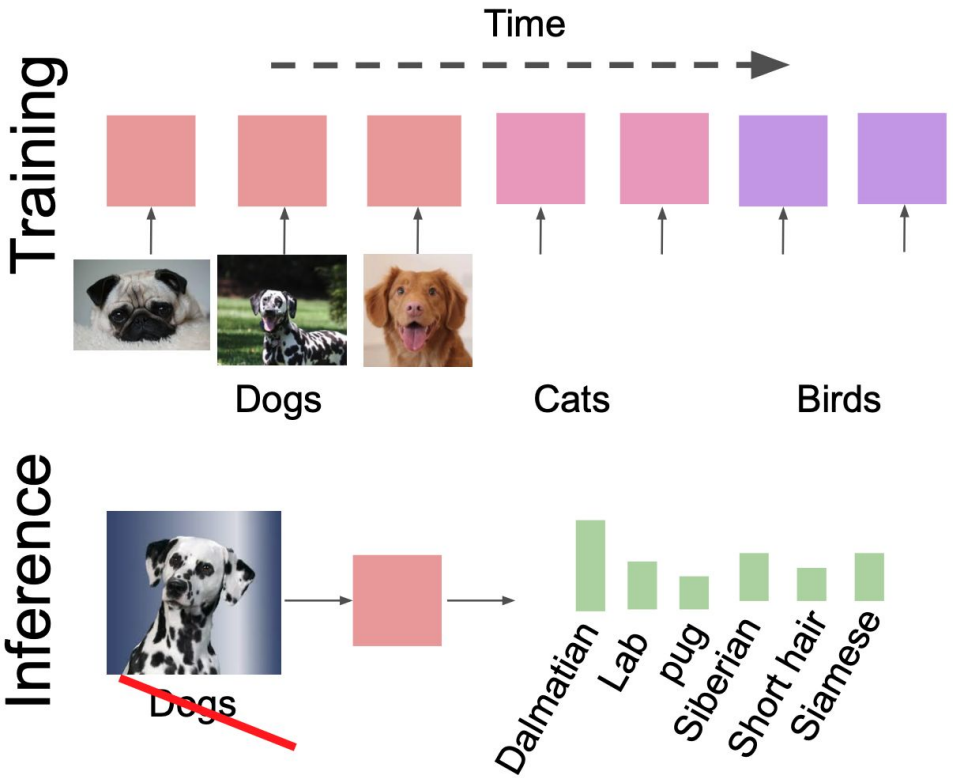
Background

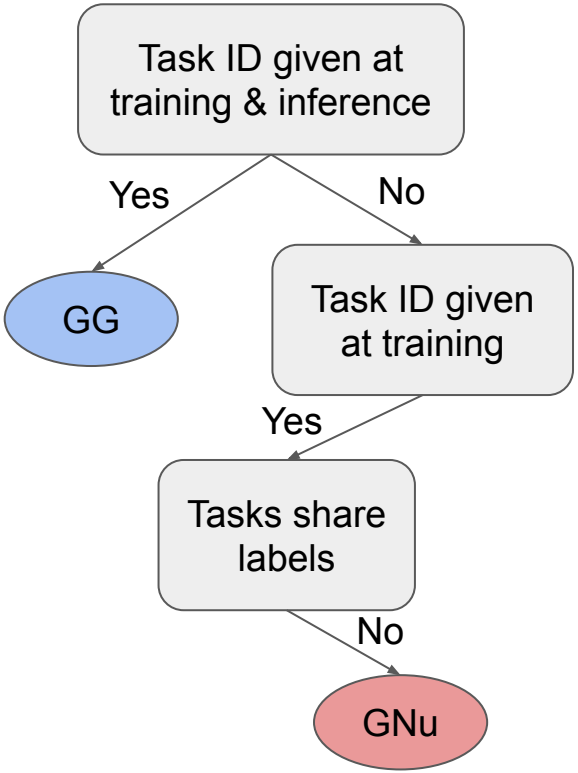

Time
Training
Dogs
Cats
Birds


Task ID given at training & inference
Yes
GG

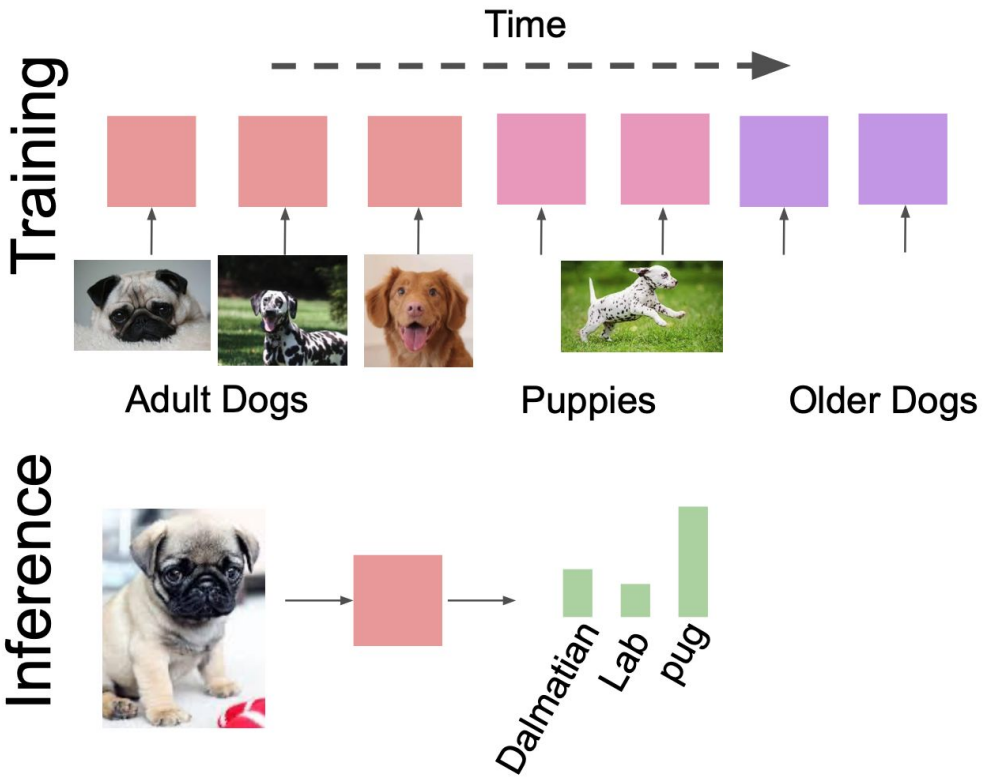

Inference
Dogs
Dalmatian
Lab
pug

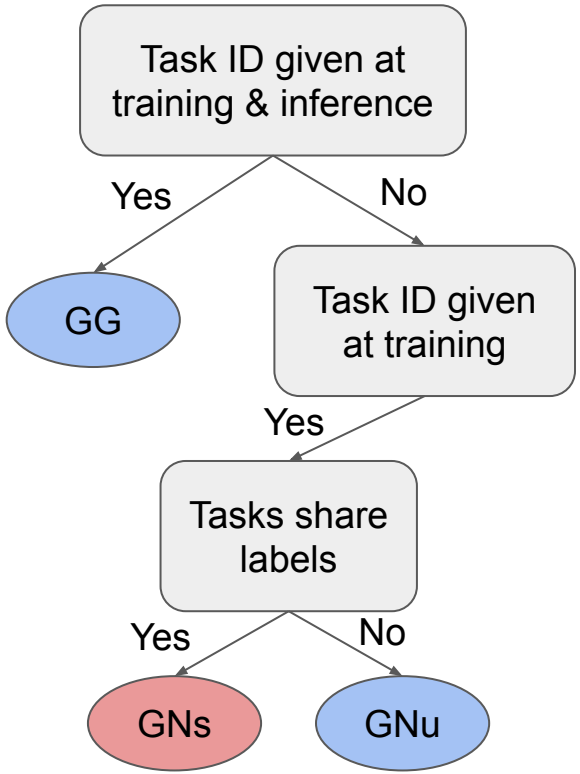
Background


Time
Training
Dogs
Cats
Birds
Inference
Dogs
Dalmatian
Lab
pug
Siberian
Short hair
Siamese

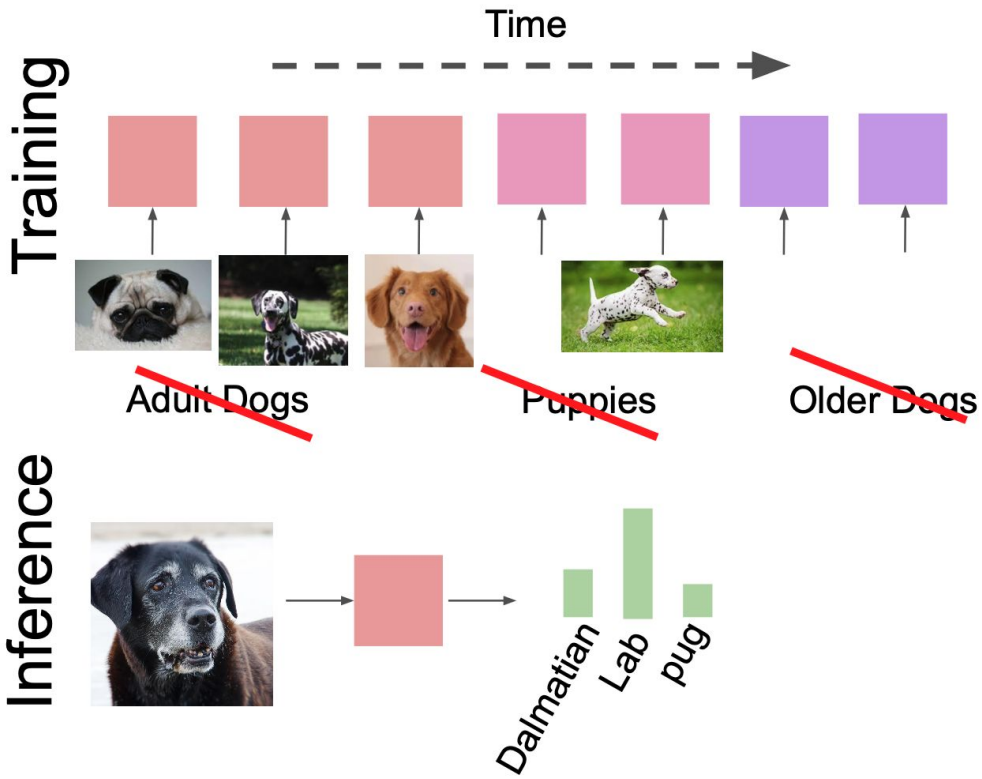

Task ID given at training & inference
Yes
No
GG
Task ID given at training
Yes
Tasks share labels
No
GNu

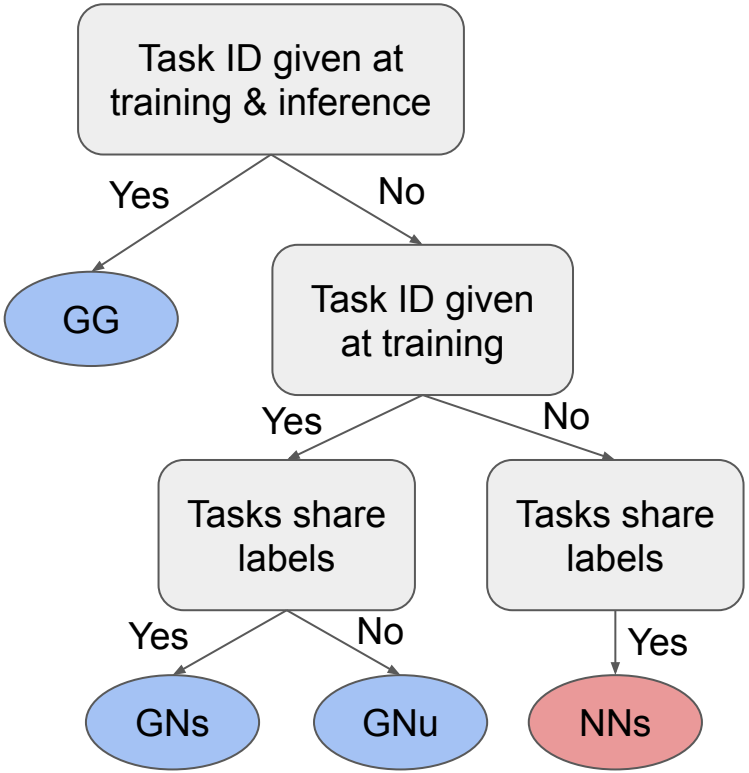
Background


Time
Training
Adult Dogs
Puppies
Older Dogs
Inference
Dalmatian
Lab
pug


Task ID given at training & inference
Yes
No
GG
Task ID given at training
Yes
Tasks share labels
Yes
No
GNs
GNu

Background


Time
Training
Adult Dogs
Puppies
Older Dogs
Inference
Dalmatian
Lab
pug


Task ID given at training & inference
Yes
No
GG
Task ID given at training
Yes
No
Tasks share labels
Tasks share labels
Yes
No
Yes
GNs
GNu
NNs

Previous works

Previous works on continual learning lie in the following three categories:

- **Regularization based methods** penalize the movement of parameters that are important for solving previous task.
- **Exemplars/replay based methods** explicitly or implicitly memorize data from previous tasks.
- **Task-specific components based methods** use different components of a network for different tasks.

SupSup belongs to the third category.

Previous works

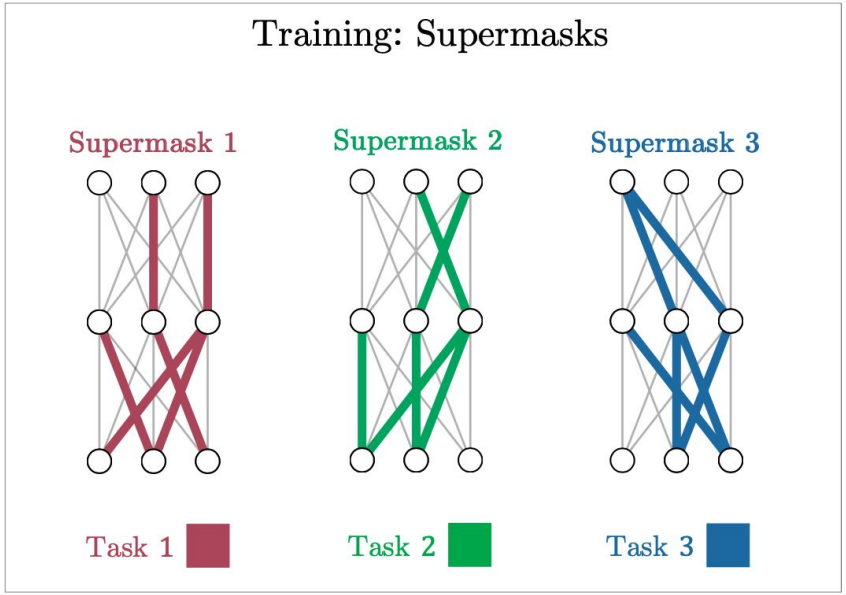
The authors consider the following two baseline methods:

- **Batchensemble (BatchE)** learns a shared weight matrix on the first task and learns only a rank-one scaling matrix for each subsequent task. The final weight for each task is the elementwise product of the shared matrix and the scaling matrix.
- **Parameter Superposition (PSP)** combines the parameter matrices of different tasks into a single matrix based on the observation that weights for different tasks are not in the same subspace.

Wen et al., 2020; Cheung et al., 2019

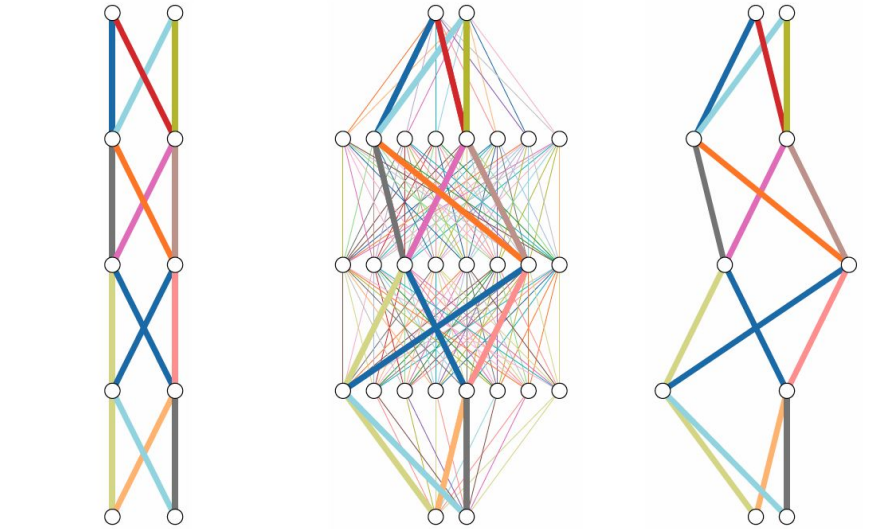
SupSup Overview - “SuperMask in Superposition”

Expressive power of subnetworks



Supermask

Assumption: If a neural network with random weights is sufficiently overparameterized, it will contain a subnetwork that perform as well as a trained neural network with the same number of parameters.



A neural network τ which achieves good performance

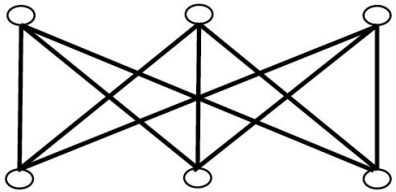
Randomly initialized neural network N

A subnetwork τ' of N

Ramanujan et al., 2020

Supermask - EdgePopup

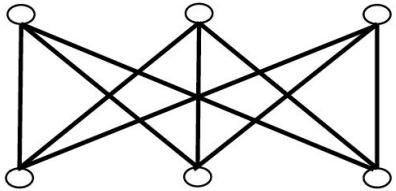
For each **edge** (u, v)
with fixed **weight** w_{uv}
assign a **score** s_{uv}



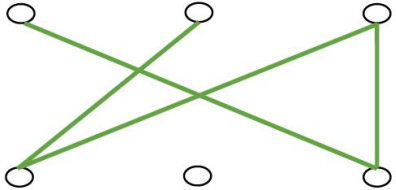
Ramanujan et al., 2020

Supermask - EdgePopup

For each **edge** (u, v)
with fixed **weight** w_{uv}
assign a **score** s_{uv}



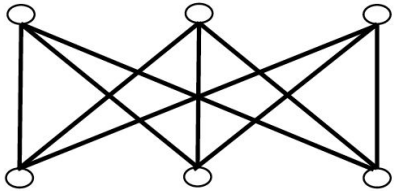
Forward: Use the
edges corresponding
to the top-k% scores



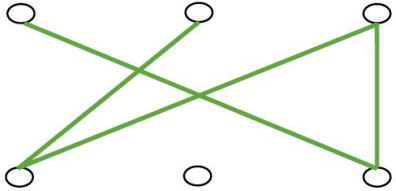
Ramanujan et al., 2020

Supermask - EdgePopup

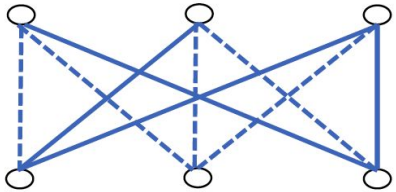
For each **edge** (u, v) with fixed **weight** w_{uv} assign a **score** s_{uv}



Forward: Use the edges corresponding to the top-k% scores



Backward: Update **all** the scores with the straight-through estimator



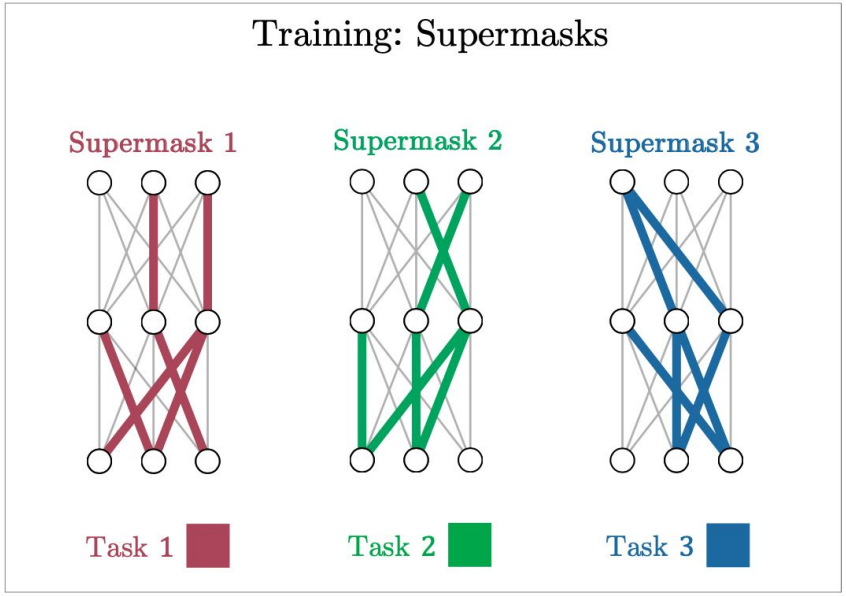
i.e. if the **weighted output** $w_{uv}\mathcal{Z}_u$ of node u is aligned with the negative gradient to v's **input** I_v, increase s_{uv}

$$s_{uv} \leftarrow s_{uv} - \alpha \frac{\partial \mathcal{L}}{\partial I_v} w_{uv} \mathcal{Z}_u$$

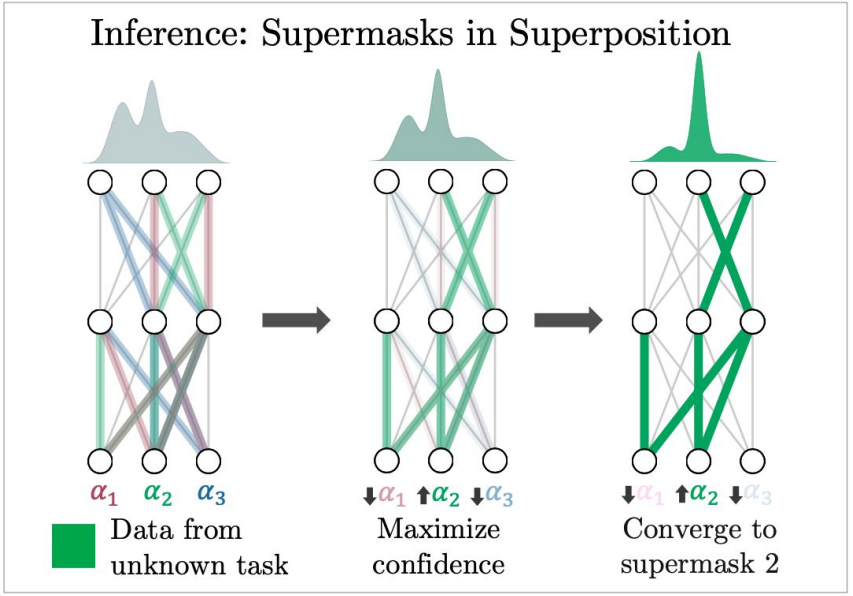
Ramanujan et al., 2020

SupSup Overview

Expressive power of subnetworks



Inference of task identity as an optimization problem



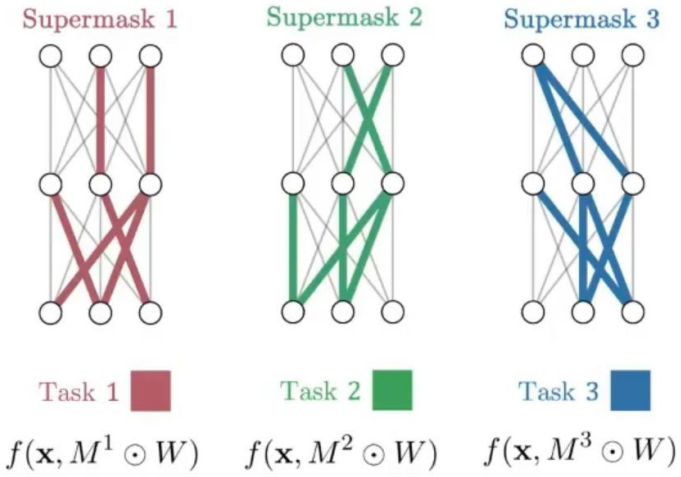
Setup

- General Setting:
  - l-way classification task.
  - Output, $p = f(x, W)$
- Continual Learning Setting:
  - k-different l-way tasks.
  - Output, $p = f(x, W \odot M)$
  - Constant input sizes across tasks.

Scenario GG (task ID given at train, given at inference)

- Training: Learn a binary mask Mi per task, keep the weights fixed.

$$p = f(x, W \odot M^i)$$



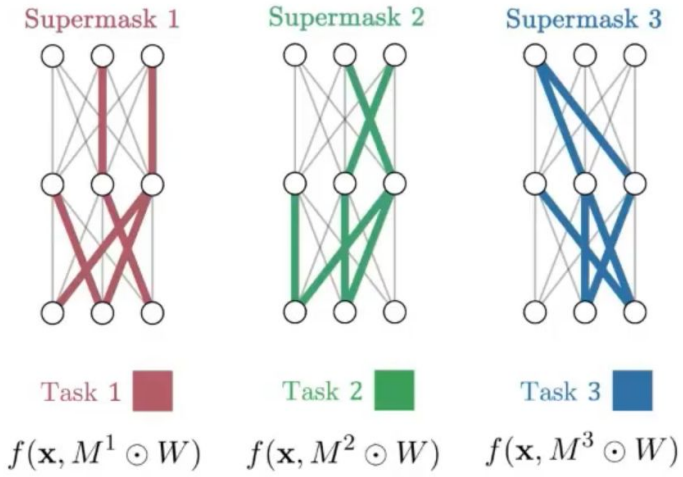
Extends Mallya et al. 2018

Scenario GG (task ID given at train, given at inference)

- Training: Learn a binary mask Mi per task, keep the weights fixed.

$$p = f(x, W \odot M^i)$$

- Inference: Use the corresponding task ID.



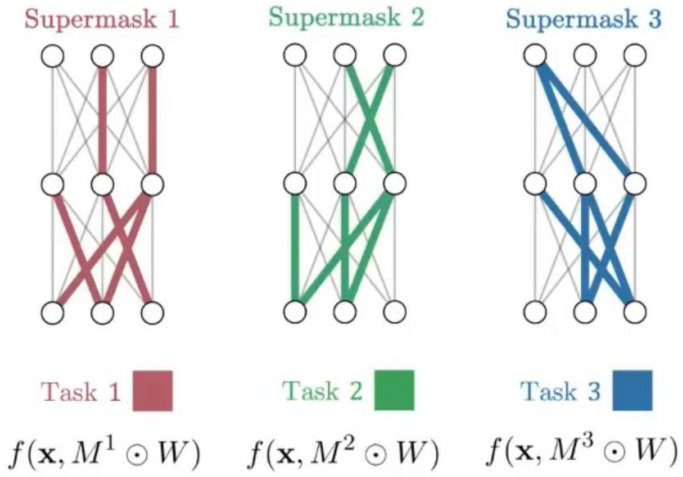
Extends Mallya et al. 2018

Scenario GG (task ID given at train, given at inference)

- Training: Learn a binary mask Mi per task, keep the weights fixed.

$$p = f(x, W \odot M^i)$$

- Inference: Use the corresponding task ID.
- Benefits: Less storage and time cost.

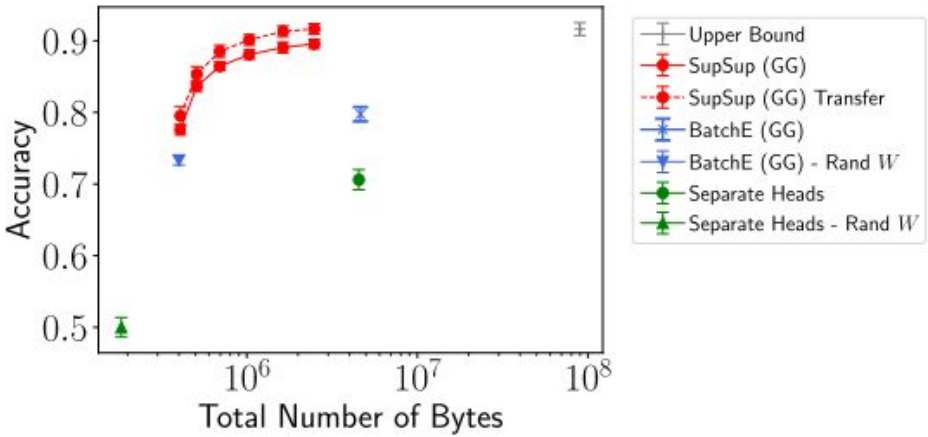


Extends Mallya et al. 2018

Scenario GG: Performance

Algorithm	Avg Top 1 Accuracy (%)	Bytes
Upper Bound	92.55	10222.81M
SupSup (GG)	89.58	195.18M
	88.68	100.98M
	86.37	65.50M
BatchE (GG)	81.50	124.99M
Single Model	-	102.23M

Dataset : SplitImageNet



Dataset : SplitCIFAR100

Scenario GNs & GNu (task ID given at train, not at inference)

- Training : Same as scenario GG.

Scenario GNs & GNu (task ID given at train, not at inference)

- Training : Same as scenario GG.
- Inference :
  - Step 1 : Infer the task.
  - Step 2 : Use the corresponding supermask.

Scenario GNs & GNu (task ID given at train, not at inference)

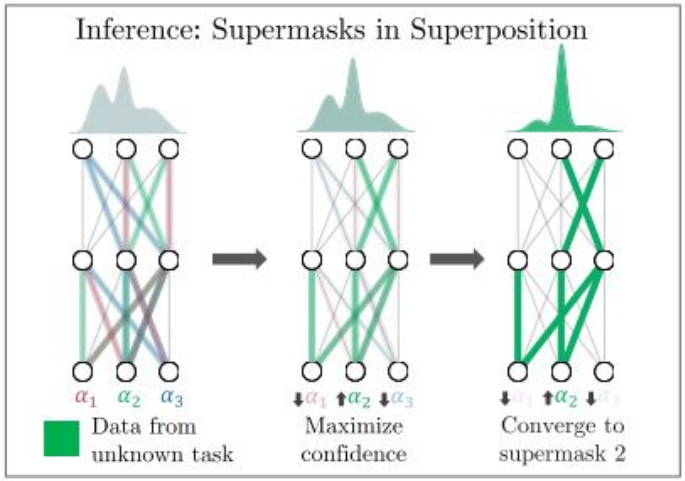
- Training : Same as scenario GG.
- Inference :
  - Step 1 : Infer the task.
  - Step 2 : Use the corresponding supermask.
- Task ID Inference Procedure
  - Associate each of k learned supermasks Mi with coefficient α_i. $\alpha_i \in [0, 1]$
  - Initialize $\alpha_i = 1/k$
  - Output of the superimposed model is given by, $p = f(x, W \odot (\sum_{i=1}^{k} \alpha_i M^i))$
  - Find coefficients that minimize the output entropy $\mathcal{H}$ of $p(\alpha)$.

How to pick the supermask ?

- Option 1 : Try each supermask individually, and pick the one with lowest entropy output.

How to pick the supermask ?

- Option 1 : Try each supermask individually, and pick the one with lowest entropy output.
- Option 2 : Stack all supermasks together, weight each mask, change α_i's to maximize the confidence.



Scenario GNs & GNu: One Shot Algorithm

Algorithm 1 One-Shot$(f, \mathbf{x}, W, k, \{M^i\}_{i=1}^k, \mathcal{H})$

1: $\alpha \leftarrow \left[\frac{1}{k} \quad \frac{1}{k} \quad \dots \quad \frac{1}{k}\right]$ ▷ Initialize α

2: $\mathbf{p} \leftarrow f\left(\mathbf{x}, W \odot \left(\sum_{i=1}^k \alpha_i M^i\right)\right)$ ▷ Superimposed output

3: **return** $\arg\max_i \left(-\frac{\partial \mathcal{H}(\mathbf{p})}{\partial \alpha_i}\right)$ ▷ Return coordinate for which objective maximally decreasing

Scenario GNs & GNu: Binary Algorithm

Algorithm 2 Binary$(f, \mathbf{x}, W, k, \{M^i\}_{i=1}^k, \mathcal{H})$

$\alpha \leftarrow \left[\frac{1}{k} \quad \frac{1}{k} \quad \dots \quad \frac{1}{k}\right]$ ▷ Initialize α

while $\|\alpha\|_0 > 1$ **do** ▷ Iterate until α has a single nonzero entry

  $\mathbf{p} \leftarrow f\left(\mathbf{x}, W \odot \left(\sum_{i=1}^k \alpha_i M^i\right)\right)$ ▷ Superimposed output

  $g \leftarrow -\nabla_\alpha \mathcal{H}(\mathbf{p})$ ▷ Gradient of objective

  for $i \in \{1, \dots, k\}$ **do** ▷ In code this **for** loop is vectorized

    if $g_i \leq$ **median** (g) **then**

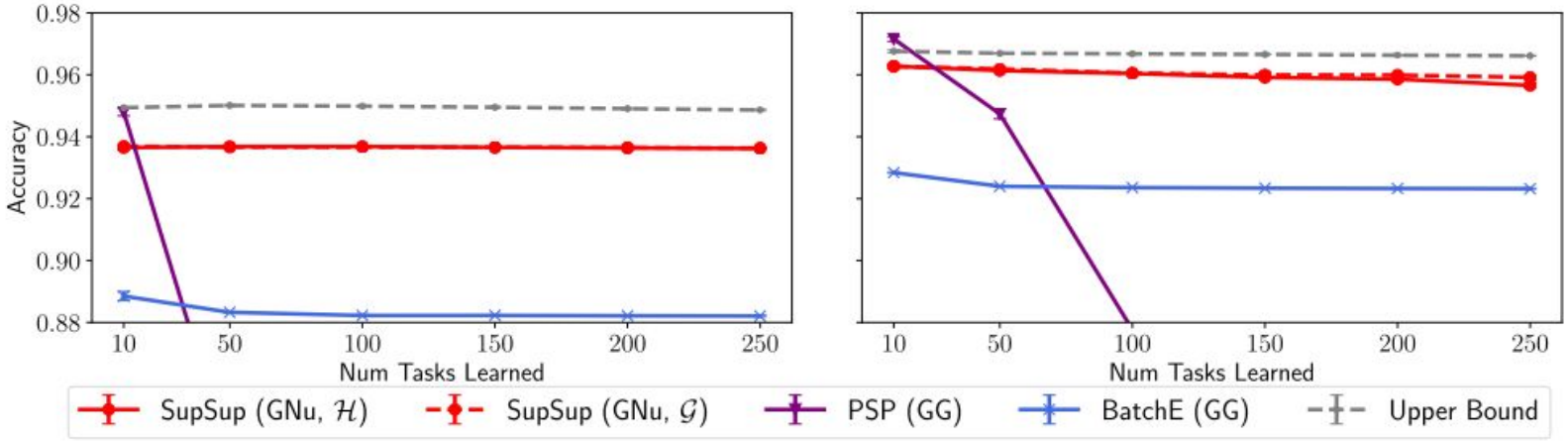
      $\alpha_i \leftarrow 0$ ▷ Zero out α_i for which objective minimally decreasing

  $\alpha \leftarrow \alpha / \|\alpha\|_1$ ▷ Re-normalize α to sum to 1

return $\arg\max_i \alpha_i$

Scenario GNu: Performance

Dataset : PermutedMNIST

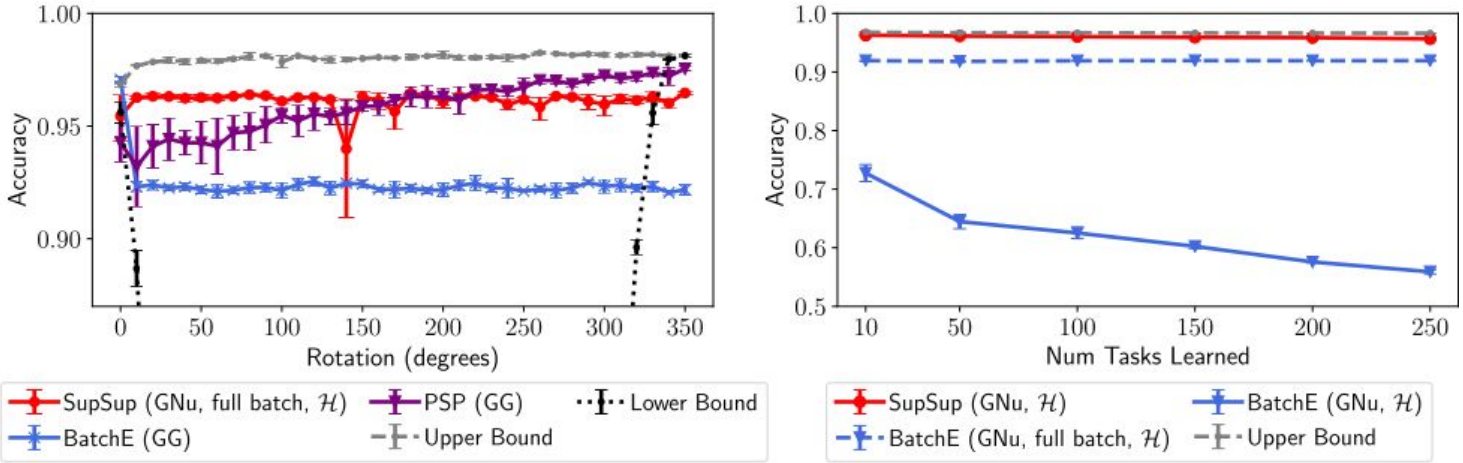


LeNet 300-100 Model

FC 1024-1024 Model

Scenario GNu: Performance

- Dataset : RotatedMNIST
- Model : FC 1024-1024 Model



Scenario NNs (task ID not given at train or inference)

- Training:
  - SupSup attempts to infer the task ID.
  - If uncertain,, then the data likely doesn't belong to a task seen so far, and a new mask is allocated.
  - SupSup is uncertain when perform task identity inference is approximately uniform.

$$v = softmax(-\nabla_\alpha \mathcal{H}(p(\alpha))) \approx uniform$$

Scenario NNs (task ID not given at train or inference)

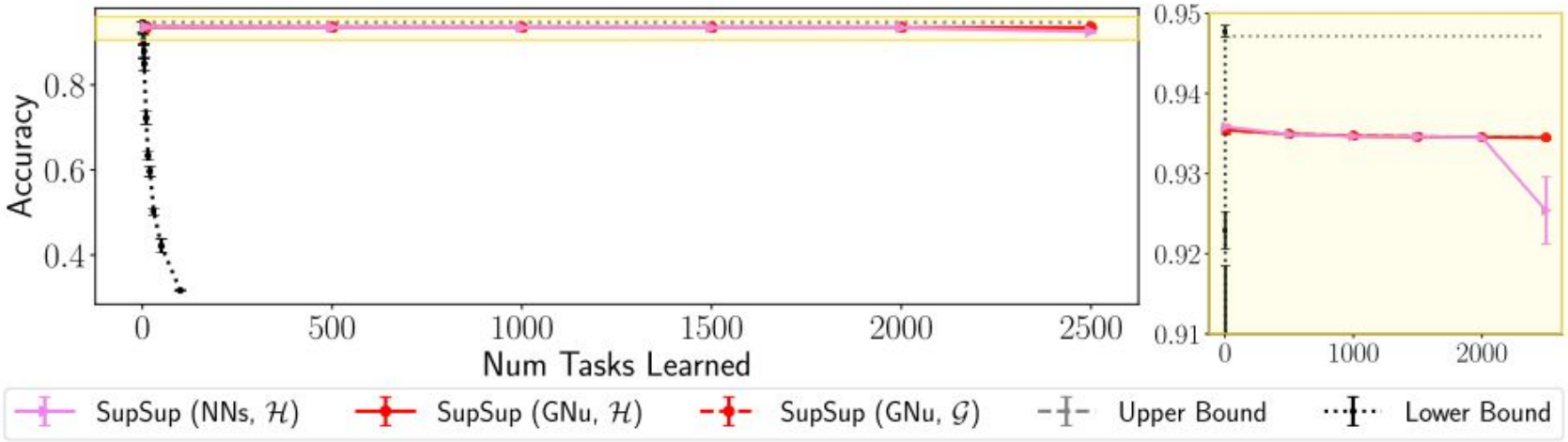
- Training:
  - SupSup attempts to infer the task ID.
  - If uncertain,, then the data likely doesn't belong to a task seen so far, and a new mask is allocated.
  - SupSup is uncertain when perform task identity inference is approximately uniform.

$$v = softmax(-\nabla_\alpha \mathcal{H}(p(\alpha))) \approx uniform$$

- Inference: Similar to Scenario GN.

Scenario NNs: Performance

- Dataset : PermutedMNIST
- Model : LeNet 300-100



Design Choices - Hopfield Network

- The space required to store the masks grows linearly as the number of tasks increases.
- Encoding the learnt masks into a Hopfield network can further reduce the model size.
- A Hopfield network implicitly encodes a series of binary strings $z^i \in \{-1, 1\}^d$ with an associated energy function $E(z)$.
- Each z^i is a local minima of $E(z)$, and can be recovered with gradient descent.

Hopfield, 1982

Design Choices - Hopfield Network

- During training, when a new mask is learnt, the corresponding binary string is encoded into the Hopfield network by updating $E(z)$.
- During inference, when a new batch of data comes, gradient descent is performed on the following problem to recover the mask:

$$\min_z \frac{\gamma t}{T} E(z) + (1 - \frac{t}{T}) \mathcal{H}(p)$$

Design Choices - Hopfield Network

- During training, when a new mask is learnt, the corresponding binary string is encoded into the Hopfield network by updating $E(z)$.
- During inference, when a new batch of data comes, gradient descent is performed on the following problem to recover the mask:

$$\min_z \frac{\gamma t}{T} E(z) + (1 - \frac{t}{T}) \mathcal{H}(p)$$

Minimize the energy function to push the solution towards a mask encoded before

Minimize the entropy to push the solution towards the correct mask

Design Choices - Hopfield Network

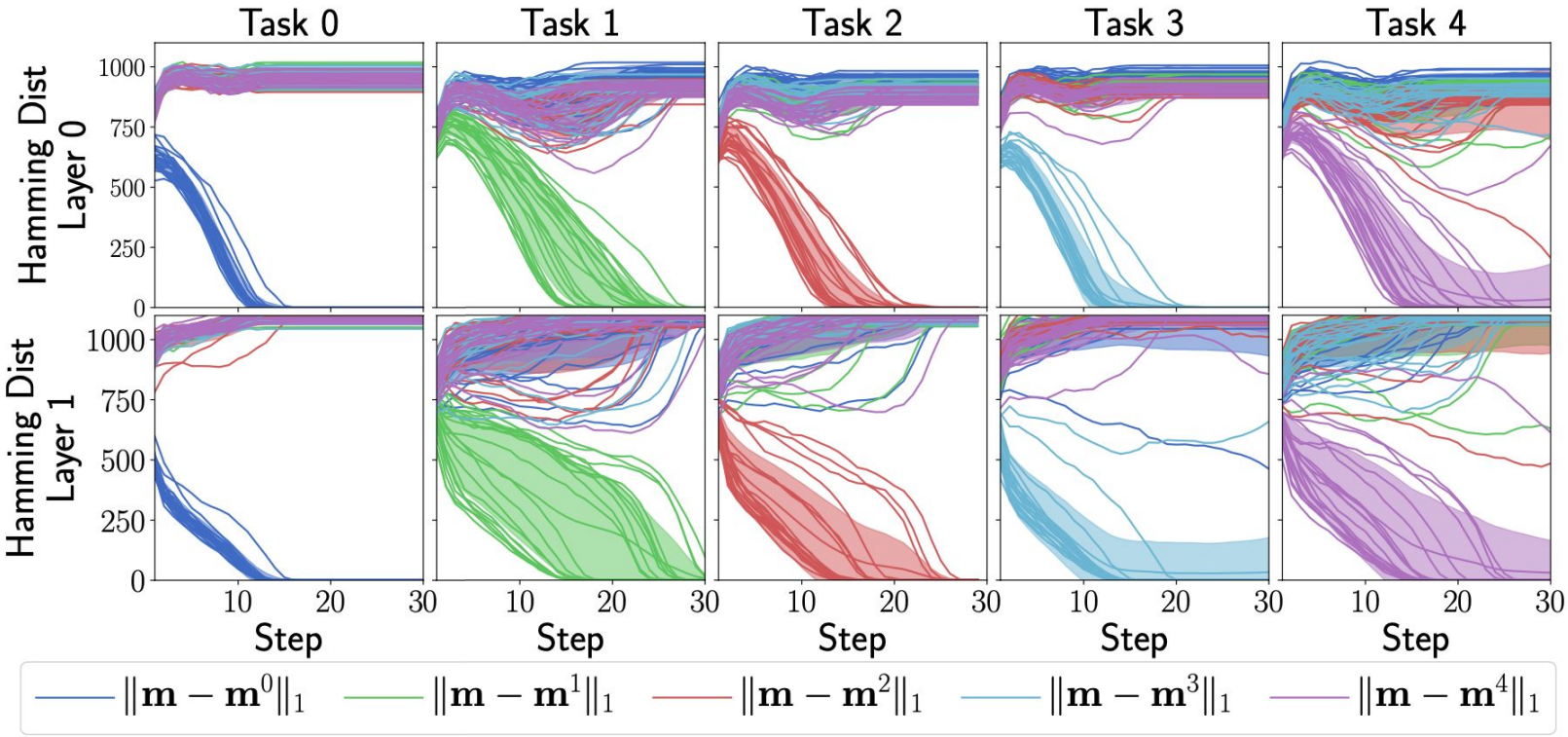
- During training, when a new mask is learnt, the corresponding binary string is encoded into the Hopfield network by updating $E(z)$.
- During inference, when a new batch of data comes, gradient descent is performed on the following problem to recover the mask:

$$\min_z \frac{\gamma t}{T} E(z) + (1 - \frac{t}{T}) \mathcal{H}(p)$$

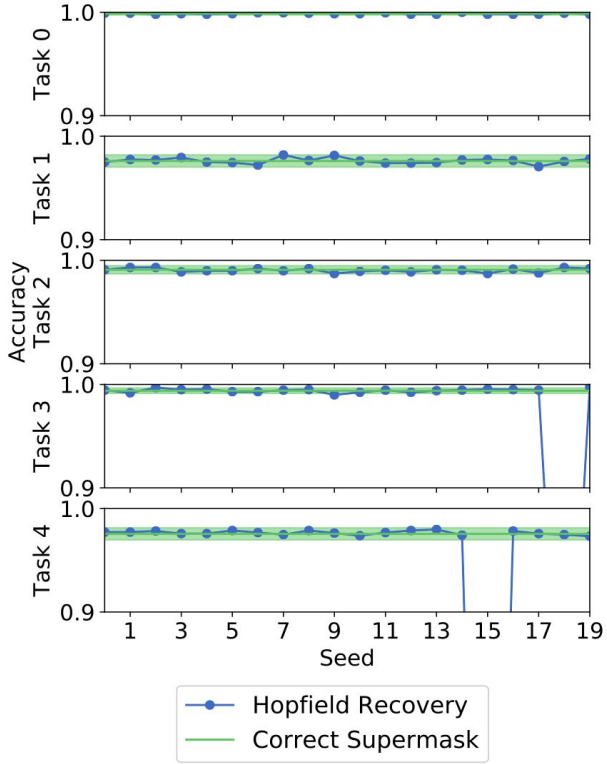
The strength of the Hopfield term increases as gradient descent goes on

The strength of the Entropy term decreases

Design Choices - Hopfield Network



Design Choices - Hopfield Network



Design Choices - Superfluous Neurons

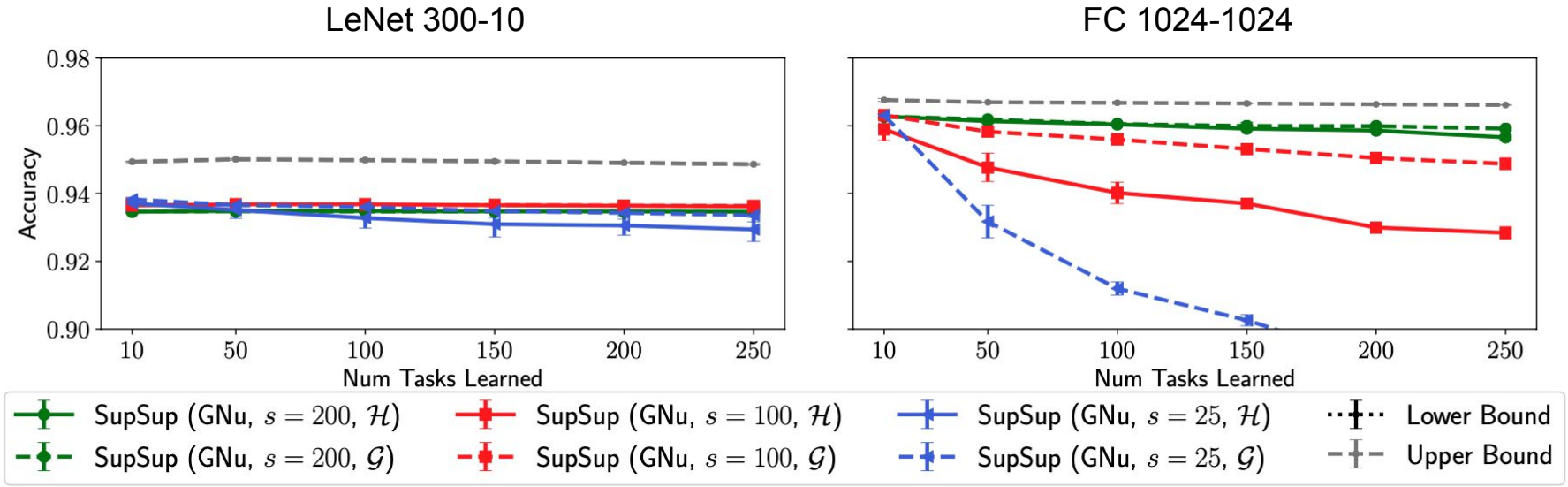
- In practice, the authors find it helps significantly to add extra neurons to the final layer.
- During training, the standard cross-entropy loss will push the values of the extra neurons down.
- The authors propose an objective $\mathcal{G} = \text{logsumexp(p)}$. When computing the gradient of $\mathcal{G}$, only the gradients w.r.t the extra neurons are enabled.

Design Choices - Superfluous Neurons

- In practice, the authors find it helps significantly to add extra neurons to the final layer.
- During training, the standard cross-entropy loss will push the values of the extra neurons down.
- The authors propose an objective $\mathcal{G} = \text{logsumexp(p)}$. When computing the gradient of $\mathcal{G}$, only the gradients w.r.t the extra neurons are enabled.
- $\mathcal{G}$ can be used as an alternative to $\mathcal{H}$ during the inference of task ID. For example, in the one-shot case:

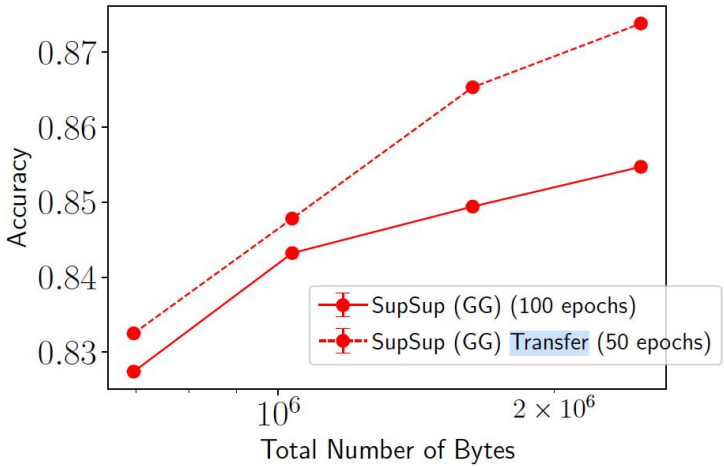
$$id = \text{argmax}_i\left(-\frac{\partial \mathcal{H}(p(\alpha))}{\partial \alpha_i}\right) \Rightarrow id = \text{argmax}_i\left(-\frac{\partial \mathcal{G}(p(\alpha))}{\partial \alpha_i}\right)$$

Design Choices - Superfluous Neurons



Design Choices - Transfer

- If each supmask is initialized randomly, the models for subsequent tasks cannot leverage the knowledge learnt from the previous tasks.
- In the transfer setting, the score matrix (for EdgePopup) for a new task is initialized with the running mean of the supermasks for all the previous tasks.



Conclusion & Discussion

- SupSup leverages the expressive power of subnetworks to perform a large amount of tasks with a single network. It infers task identities by solving an optimization problem when unknown.
- SupSup achieves the state-of-the-art performance when task identities are given, and performs well even when task identities are missing at both training and inference time.
- Task inference will fail if the models are not well calibrated and overly confident for the wrong task.
- It relies on the Supermask setting, and is hard to be generalized to the common settings where the weights of the models are trained.