# Active Bayesian meta-learning for brain cell classification

HoUston Learning Algorithms (HULA) Lab
Presented by Pengyu (Ben) Yuan

# Outline

- What is meta-learning?
  - Problem of supervised-learning
  - Model-agnostic meta-learning (MAML)
- t**A**sk-au**G**mented act**I**ve meta-**LE**arning (AGILE)
  - Problems of MAML
  - Task augmentations
  - Active-learning in real-task
- Experiments and results

# What is meta-learning?

# Problem of supervised-learning

- It often requires large & diverse data to train a good model.
- The human, on the other hand, can learn new concept or skills more efficiently.

Russakovsky et al. ‘14

GPT-2

Radford et al. ‘19

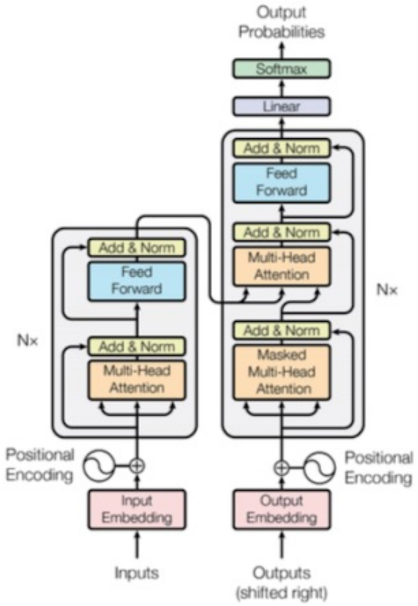



Figure 1: The Transformer - model architecture.

Vaswani et al. ‘18

Source: Finn & Levine, Meta-learning tutorial

# Meta-learning

- Supervised learning

$$\phi^* = \arg\max_{\phi} \log p(\phi|\mathcal{D})$$

$\mathcal{D} = \{(\mathbf{x}_q, \mathbf{y}_q)\}_{q=1}^{Q}$ :Observed dataset (Contains Q samples)

$\phi$ :Model parameters

- Incorporate additional data (to reduce Q)

$$\phi^* = \arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta}})$$

$\mathcal{D}_{\text{meta}} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$ :Additional datasets (From meta tasks)

$\mathcal{D}_i = \left\{\left(\mathbf{x}_q^i, \mathbf{y}_q^i\right)\right\}_{q=1}^{Q_i}$ :One meta dataset

- Meta learning

$$\phi^* = \arg\max_{\phi} \log p(\phi|\mathcal{D}, \theta^*)$$

$$\theta^* = \arg\max_{\theta} \log p(\theta|\mathcal{D}_{\text{meta}})$$

$\theta$ :Meta-learning parameters

# Meta-learning

- Supervised learning

$$\phi^* = \arg\max_{\phi} \log p(\phi|\mathcal{D})$$

$\mathcal{D} = \{(\mathbf{x}_q, \mathbf{y}_q)\}_{q=1}^{Q}$ :Observed dataset (Contains Q samples)

$\phi$ :Model parameters

- Incorporate additional data (to reduce Q)

$$\phi^* = \arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta}})$$

$\mathcal{D}_{\text{meta}} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$ :Additional datasets (From meta tasks)

$\mathcal{D}_i = \left\{\left(\mathbf{x}_q^i, \mathbf{y}_q^i\right)\right\}_{q=1}^{Q_i}$ :One meta dataset

- Meta learning

$$\phi^* = \arg\max_{\phi} \log p(\phi|\mathcal{D}, \theta^*)$$

$\theta^* = \arg\max_{\theta} \log p(\theta|\mathcal{D}_{\text{meta}})$ $\theta$ :Meta-learning parameters

**Meta-learning task**

# Brain cell classification example

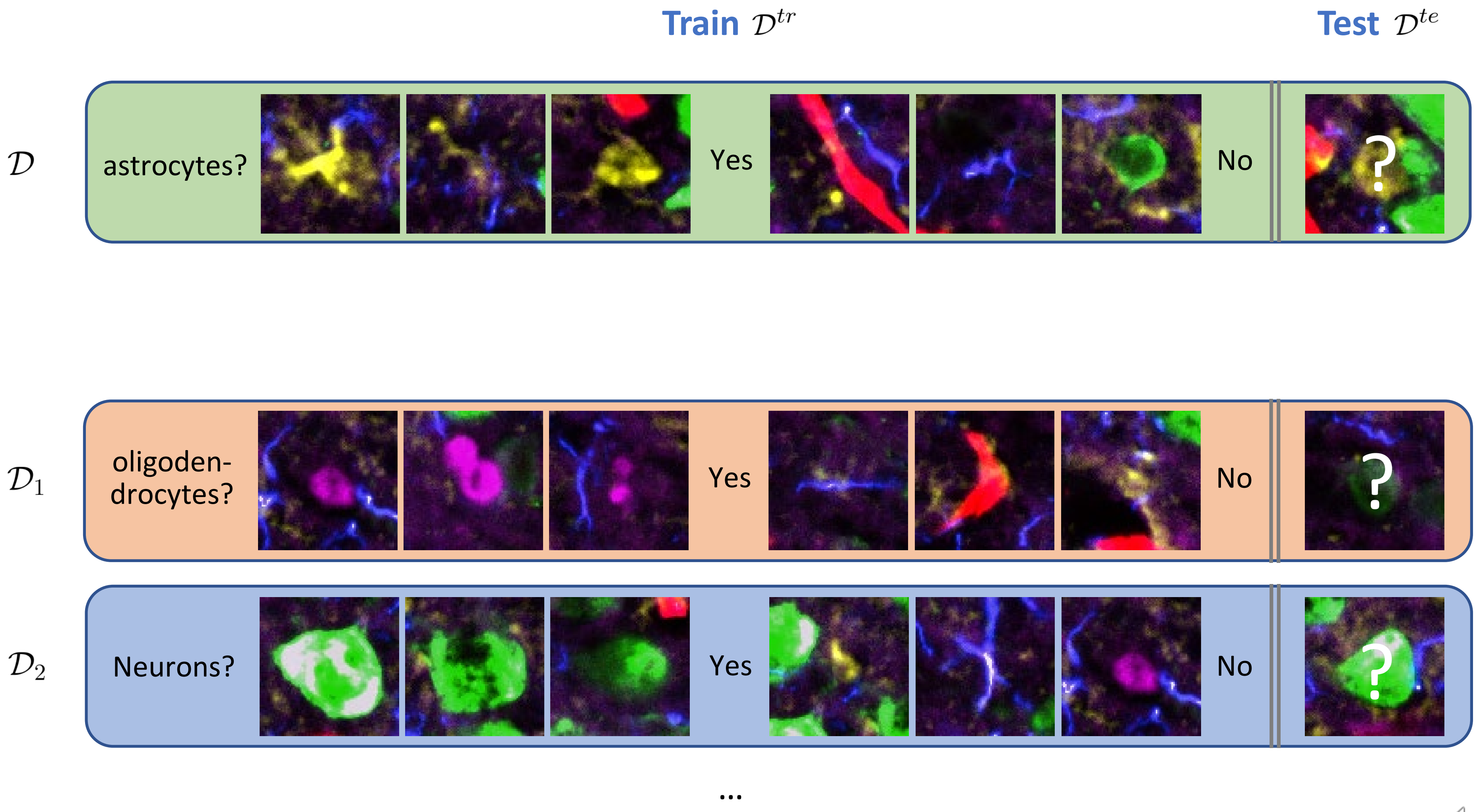
- Real task

$$\mathcal{D} = \{(\mathbf{x}_q, \mathbf{y}_q)\}_{q=1}^{Q}$$

- Meta tasks

$$\mathcal{D}_{\text{meta}} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$$

$$\mathcal{D}_i = \left\{\left(\mathbf{x}_q^i, \mathbf{y}_q^i\right)\right\}_{q=1}^{Q_i}$$



# Model-agnostic meta-learning (MAML)

- Fine-tuning/learning/adaptation

Update model parameters $\phi$

- Meta-learning

Update meta-parameters $\theta$

**Fine-tuning**
[test-time]

pre-trained parameters

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

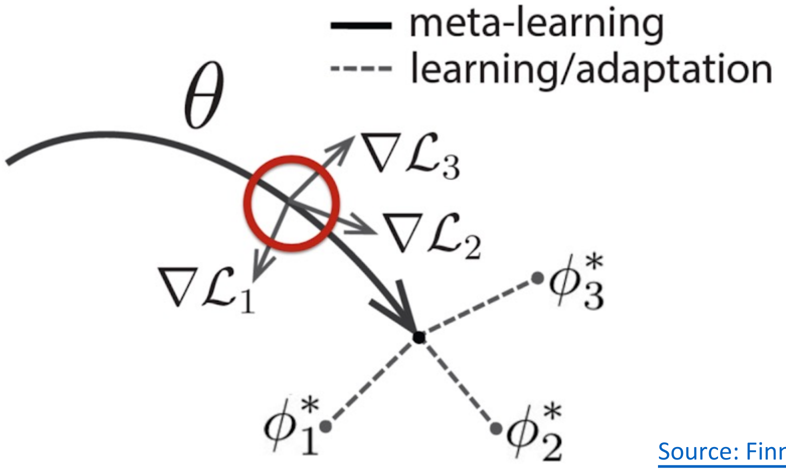
training data
for new task

**Meta-learning**

$$\min_\theta \sum_{\text{task } i} \mathcal{L}(\underbrace{\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})}_{\phi_i}, \mathcal{D}_i^{\text{ts}})$$

$\theta$ parameter vector being meta-learned

$\phi_i^*$ optimal parameter vector for task i




Source: Finn & Levine, Meta-learning tutorial

# tAsk-auGmented actIve meta-LEarning (AGILE)

# Problem of MAML

- It requires a lot of meta-task to train the meta-parameters
- There is no uncertainty for the classification results
- It doesn’t use the most important samples for adaptation
- It is not dynamic enough

# Task augmentations

- Not enough meta-tasks → meta-overfitting
- Task augmentations:
  1. Flipping the label

     $$y' = z(1-y) + (1-z)y,$$

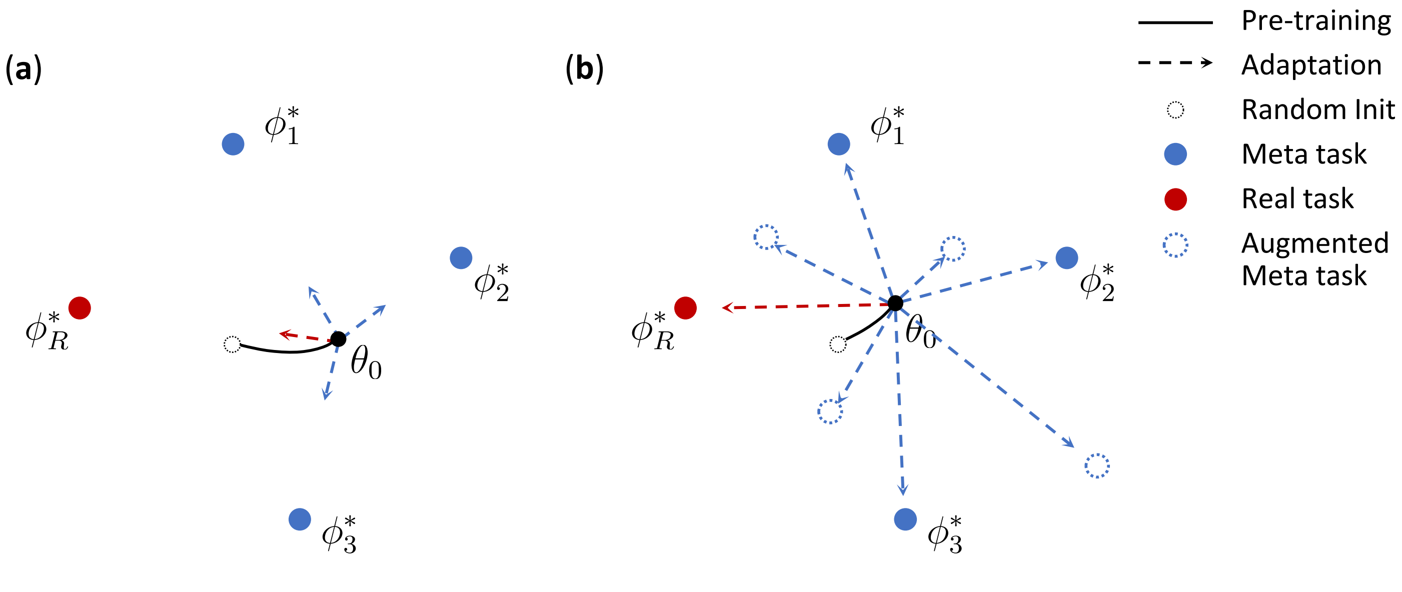
     where $z \sim \text{Bernoulli}(p_f)$

  2. Shuffling the order of input channels

     $$\mathbf{x}' = \mathbf{x} * \mathbf{s}_{ij}, \quad i, j = 1, 2, 3 \ldots c$$

     where $\{\mathbf{s}_{ij}\}_{i=1}^{c} \in \mathbb{R}^{1\times 1\times c}$

  3. Rotating the images



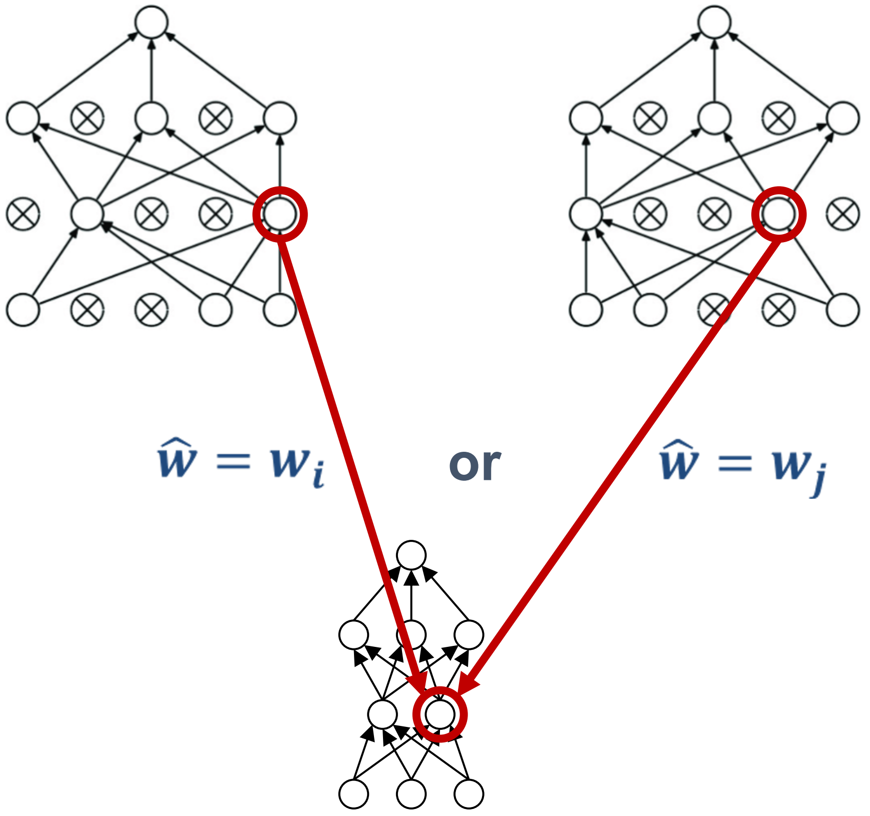

**Comparison of (a) transfer learning and (b) task-augmented meta-learning.**

# Active-learning in real-task

- Active-learning:
  Use the most valuable samples for adaptation
- Valuable:
  High uncertainty obtained from Monte-Carlo Dropout

$$H\left(\mathbf{y}^{\mathrm{te}}|\mathbf{x}^{\mathrm{te}}, \mathcal{D}^{\mathrm{train}}\right) = -\sum_{\mathbf{y}^{\mathrm{te}} \in \mathcal{Y}} p(\mathbf{y}^{\mathrm{te}}|\mathbf{x}^{\mathrm{te}}, \mathcal{D}^{\mathrm{train}}) \log p(\mathbf{y}^{\mathrm{te}}|\mathbf{x}^{\mathrm{te}}, \mathcal{D}^{\mathrm{train}})$$

- Dynamic:
  Random number of training samples for each task during meta-training




**Randomly dropout neurons at different iterations equivalent to sampling from a distribution.**

Source: Nguyen et al, Bayesian deep learning tutorial

# Experiments and results

# Datasets

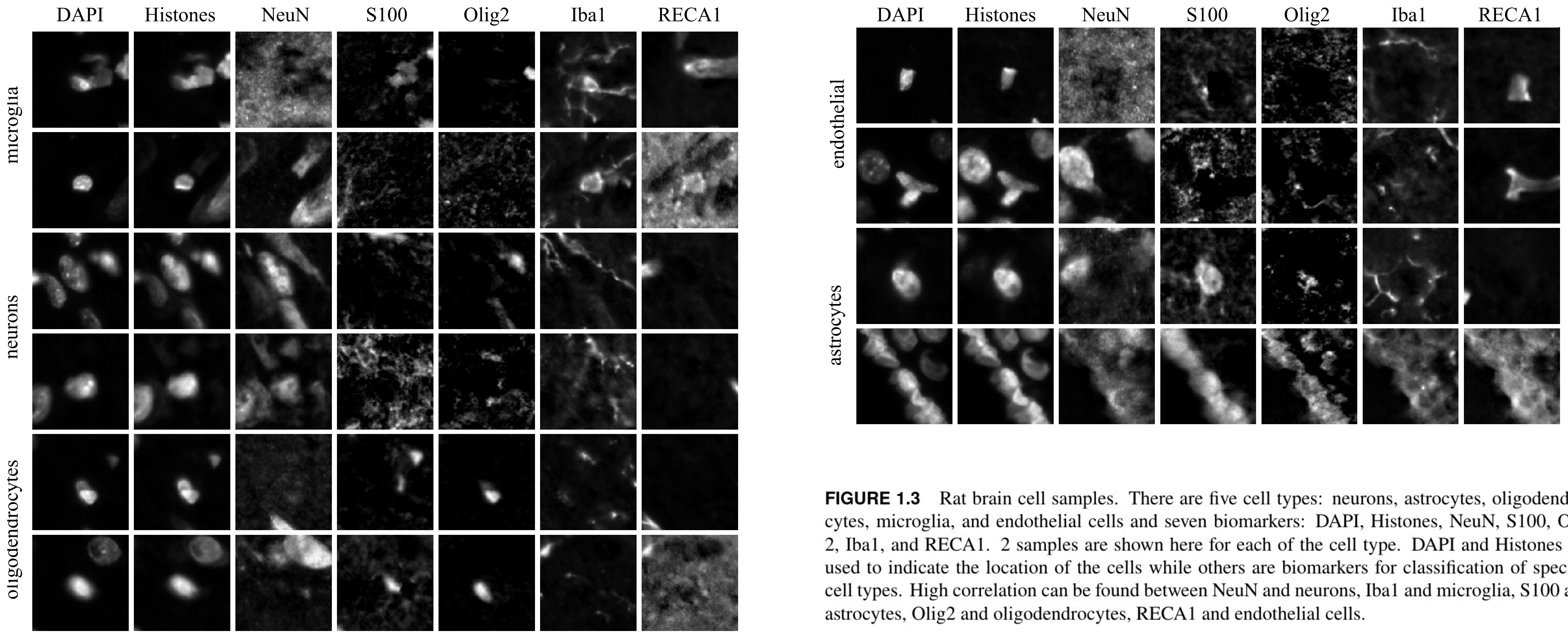



**FIGURE 1.3** Rat brain cell samples. There are five cell types: neurons, astrocytes, oligodendrocytes, microglia, and endothelial cells and seven biomarkers: DAPI, Histones, NeuN, S100, Olig 2, Iba1, and RECA1. 2 samples are shown here for each of the cell type. DAPI and Histones are used to indicate the location of the cells while others are biomarkers for classification of specific cell types. High correlation can be found between NeuN and neurons, Iba1 and microglia, S100 and astrocytes, Olig2 and oligodendrocytes, RECA1 and endothelial cells.

# Experiments and results

- Settings:
  - Five cell types:
    - 3 meta-tasks: neurons , oligodendrocytes, microglia
    - 2 real-tasks: astrocytes and endothelial cells

    or
    - 3 meta-tasks: neurons , astrocytes, endothelial cells
    - 2 real-tasks: oligodendrocytes and microglia
  - Seven biomarkers:
    DAPI, Histones, NeuN, S100, Olig 2, Iba1 and RECA1
- Network: CNN
- Baselines:
  - lower bound (supervised training with a small dataset)
  - upper bound (fully supervised training)
  - a pretrained model (transfer learning)
  - a state-of-the-art method (MAML)

**Table 1.** Methods configuration comparison which differ mainly in the data they use and the training framework. Meta-learning methods are supposed to perform well with few training samples and little training time. (# means the number of)

| Methods | Use data | | | in Real-train | | # Meta tasks |
|---|---|---|---|---|---|---|
| | Meta-train | Meta-test | Real-train | # samples | # gradient updates | |
| Vanilla_limit | - | - | ✓ | 16 (1%) | 100 | 0 |
| Vanilla_full | - | - | ✓ | 960 (60%) | 100 | 0 |
| Transfer | ✓ | - | ✓ | 16 (1%) | 100 | 3 |
| MAML | ✓ | ✓ | ✓ | 16 (1%) | 1 | 3 |
| AGILE(phase I) | ✓ | ✓ | ✓ | 16 (1%) | 1 | many |
| AGILE(phase II) | ✓ | ✓ | ✓ | 16 (1%) | 1 | many |
| AGILE(phase II) | ✓ | ✓ | ✓ | 160 (10%) | 1 | many |

# Experiments and results

- Few shot classification results:

Task split1

**TABLE 1.2** Quantitative results of different methods in rat brain cell classification experiments with first task split. Vanilla method use all available training data (60%) and act as the upper bound while AGILE method get the highest accuracy using very few training data (1%).

| **Methods** (Size %) | **Precision** | **Recall** | **F1-score** | **Accuracy**(± Std) | **CI$_{95}$** |
|---|---|---|---|---|---|
| Vanilla_limit (1%) | 0.642 | 0.622 | 0.632 | 0.637(±0.062) | 0.632 - 0.642 |
| Vanilla_full (60%) | 0.937 | **0.965** | **0.951** | **0.950**(±0.021) | 0.948 - 0.952 |
| Transfer (1%) | 0.447 | 0.433 | 0.440 | 0.449(±0.085) | 0.449 - 0.456 |
| MAML (1%) | 0.408 | 0.402 | 0.405 | 0.409(±0.030) | 0.406 - 0.412 |
| AGILE(phase I) (1%) | 0.791 | 0.790 | 0.791 | 0.791(±0.054) | 0.786 - 0.796 |
| AGILE(phase II) (1%) | 0.883 | 0.926 | 0.904 | 0.902(±0.048) | 0.898 - 0.906 |
| AGILE(phase II) (10%) | **0.950** | 0.951 | **0.951** | **0.950**(±0.044) | 0.946 - 0.954 |

Task split2

**TABLE 1.3** Quantitative results of different methods in rat brain cell classification experiments with second task split.

| **Methods** (Size %) | **Precision** | **Recall** | **F1-score** | **Accuracy**(± Std) | **CI$_{95}$** |
|---|---|---|---|---|---|
| Vanilla_limit (1%) | 0.745 | 0.711 | 0.728 | 0.738(±0.084) | 0.715 - 0.761 |
| Vanilla_full (60%) | **0.948** | 0.958 | **0.952** | **0.952**(±0.011) | 0.946 - 0.960 |
| Transfer (1%) | 0.713 | 0.710 | 0.712 | 0.708(±0.089) | 0.700 - 0.716 |
| MAML (1%) | 0.669 | 0.678 | 0.674 | 0.675(±0.108) | 0.666 - 0.684 |
| AGILE(phase I) (1%) | 0.929 | 0.892 | 0.910 | 0.913(±0.055) | 0.908 - 0.918 |
| AGILE(phase II) (1%) | 0.896 | 0.874 | 0.885 | 0.888(±0.088) | 0.861 - 0.915 |
| AGILE(phase II) (4%) | 0.939 | **0.965** | **0.952** | **0.952**(±0.053) | 0.936 - 0.968 |

# Experiments and results

- **Fast adapting ability:**
  (a) AGILE method learns faster compared with other baselines.

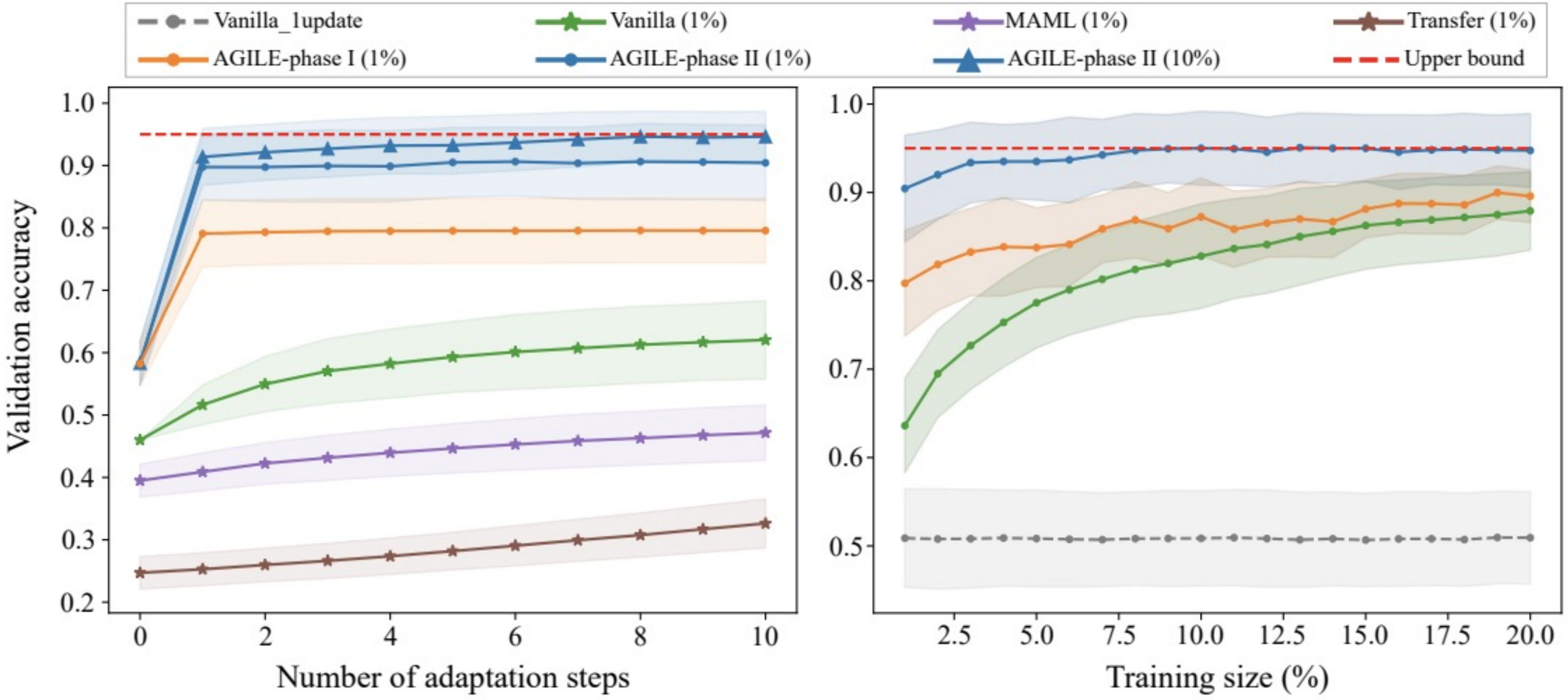
  Fewer updates

- **Adapt with few samples:**
  (b) AGILE method can get a much better performance with smaller training size.

  Fewer samples

Few is enough

Task split1



# Experiments and results

- **Fast adapting ability:**
  (a) AGILE method learns faster compared with other baselines.

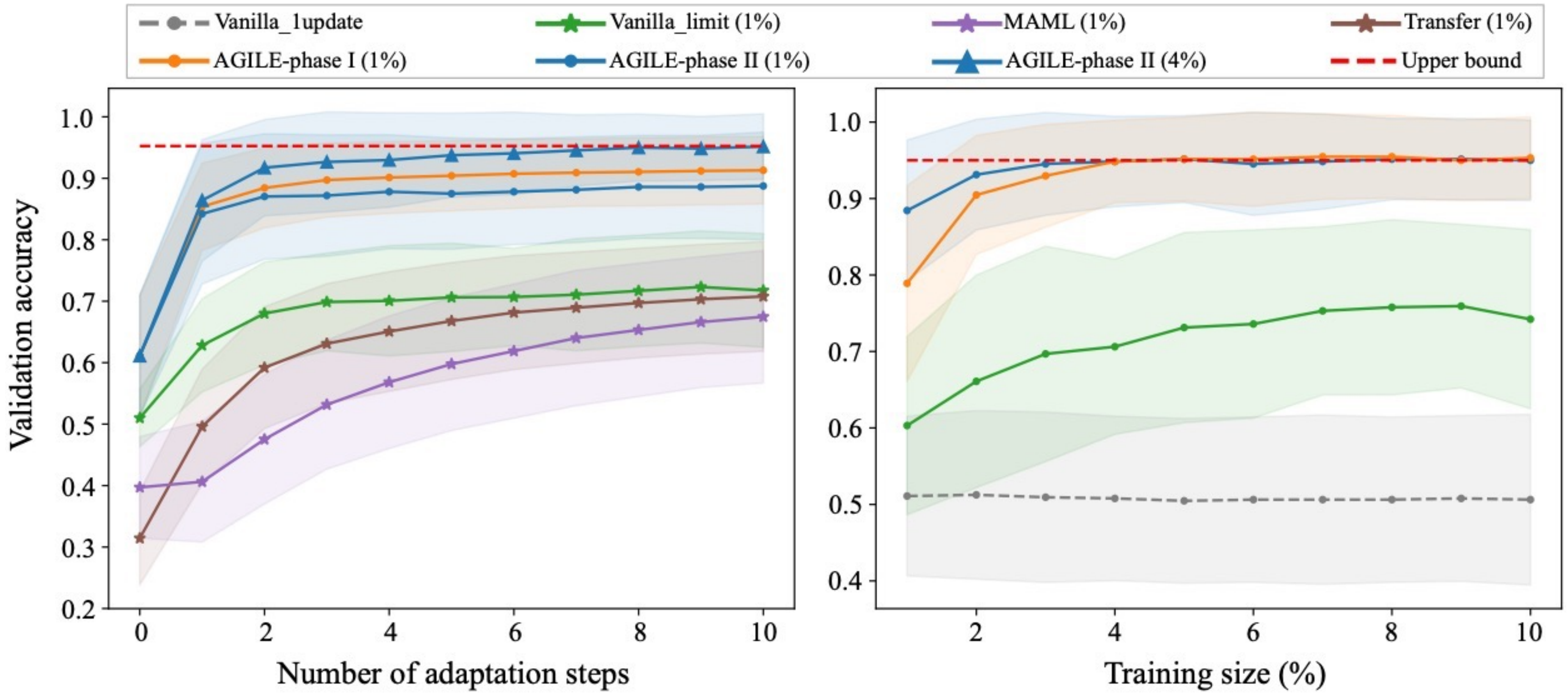
  Fewer updates

- **Adapt with few samples:**
  (b) AGILE method can get a much better performance with smaller training size.

  Fewer samples

Few is enough

Task split2




thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
bedankt
dziękuję
go raibh maith agat
mochchakkeram
sukriya
kop khun krap
terima kasih
dank je
спасибо
рахмат
Баярлалаа
faafetai lava
mersi
kia ora
barka
welalin
tack
spas
vinaka
blagodaram
спасибі
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
хвала
asante
manana
obrigada
murakoze
tenki
djiere dieuf
tau
дякую
mamnun
chokrane
kiitos
dankie
nanni
nandri
dhanyavad
maururu
hvala
köszönöm
gràçie
bayarlalaa
enkosi
chnorakaloutioun
gratias ago
gràcies
sulpáy
dankon
aćiū
akun
taiku
arigatô
takk
dakujem
trugarez
sobodi
děkuji
mèsi
didi madloba
sagolun
najis tuke
ありがとう
tanemirt
grazie
kam sah hamnida
rahmat
rahmet
shukriya
mercé
мерси
diolch
dhanyavadagalu
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ