MemFlow: A Lightweight Forward Memorizing Framework for Quick Domain Adaptive Feature Mapping

Supplementary Material

7. Supplementary Proof: Gaussian Distribution of Neuron Memory

Theorem 3 (Gaussian Distribution of Neuron Memory) *the distribution of memory signals in the k^{th} class on the i^{th} neuron of MemFlow follows the Gaussian distribution:*

$$\mathbb{P}(\hat{m}_i^k \mid y = k) \sim \mathcal{N}(\mu_i^k, (\sigma_i^k)^2), \tag{19}$$

.

Lemma 3.1 *Define the t-th round memory contribution of class k to neuron i as $Z_t^k = o_{i,t}^k$ (where $o_{i,t}^k$ is the neuron's output at round t). Then $\{Z_1^k, Z_2^k, \ldots, Z_T^k\}$ is i.i.d. across rounds.*

[**Proof of Lemma 3.1**]

For distinct $t_1 \neq t_2$, independence of $Z_{t_1}^k$ and $Z_{t_2}^k$ holds due to: (i) The activation indicators are round independent; (ii) The signal propagation rules are time-invariant (fixed topology and edge weights); (iii) Input features are i.i.d., so the initial output $o_{j,0}^k = x_j^k$ introduces no round-specific bias. Meanwhile, identical distribution follows from fixed propagation rules, weights, and activation threshold, ensuring $Z_{t_1}^k$ and $Z_{t_2}^k$ share the same distribution.

[**Proof of Theorem 3**] The theorem can be proved in the following steps:

Step 1: Memory Signal Decomposition. According to the memory update rule (Eq. (4)), $\hat{m}_i^k = \sum_{t=1}^T o_{i,t}^k = \sum_{t=1}^T Z_t^k$. By Lemma, $\{Z_t^k\}$ is i.i.d. with finite mean μ_Z^k and positive variance $(\sigma_Z^k)^2$.

Step 2: Apply Lindeberg-Lévy CLT. For i.i.d. sequences with finite mean/variance (satisfied by Lemma), the CLT gives:

$$\frac{\sum_{t=1}^T Z_t^k - T\mu_Z^k}{\sqrt{T}\sigma_Z^k} \xrightarrow{d} \mathcal{N}(0,1) \quad (T \to \infty). \tag{20}$$

Step 3: Gaussian Distribution Deduction. From Eq. (20), $\hat{m}_i^k$ converges to $\mathcal{N}(T\mu_Z^k, T(\sigma_Z^k)^2)$. For practical T, this approximates to $\mathbb{P}(\hat{m}_i^k \mid y = k) \sim \mathcal{N}(\mu_i^k, (\sigma_i^k)^2)$ with $\mu_i^k = T\mu_Z^k$ and $(\sigma_i^k)^2 = T(\sigma_Z^k)^2$.

Step 4: Consistency with Gaussian Blur. The Gaussian blur in Eq. (9) of the main text convolves the memory signal with kernel $g(x_1, x_2) = \exp(-(x_1 - x_2)^2/(2\sigma_1^2))$. As convolution preserves Gaussianity for independent Gaussians, we have

$$Q(\hat{m}_i^k \mid y = k) = \mathcal{N}(\mu_i^k, (\sigma_i^k)^2 + \sigma_1^2). \tag{21}$$

8. Supplementary Proof: Convergence of Reinforced Memorization

Theorem 4 (Convergence of Reinforced Memorization) *Let $\Theta_t = \{(\mu_i^k(t), \sigma_i^k(t)) \mid 1 \leq i \leq N, 1 \leq k \leq C\}$ denote the memory parameter set of MemFlow at iteration t of reinforced memorization. The error rate of pseudo-labels is $\epsilon \in [0,1]$, where $\epsilon = \mathbb{P}(\hat{y} \neq y^*)$ (y^* is the ground-truth label. Assume the following conditions hold:*

1. *Confidence Weight Boundedness: The update confidence $E_i(\hat{m}_i, \hat{y}) \in [\underline{E}, \overline{E}]$ for constants $0 < \underline{E} \leq \overline{E} < \infty$.*
2. *Parameter Update Smoothness: The memory parameters $\mu_i^k(t), \sigma_i^k(t)$ are updated via Eqs. (16)-(17) with fixed temperature $\beta \in (0,1)$ and batch size $B \geq 1$.*
3. *Ground-truth Distribution Existence: There exists a true parameter set $\Theta^* = \{(\mu_i^{k,*}, \sigma_i^{k,*})\}$ such that $\mathbb{P}(\hat{m}_i \mid y = k) \sim \mathcal{N}(\mu_i^{k,*}, (\sigma_i^{k,*})^2)$.*

Then the parameter sequence $\{\Theta_t\}$ converges almost surely (a.s.) to a bounded set $\Theta^\dagger$ satisfying:

$$\|\Theta^\dagger - \Theta^*\| \leq O\left(\frac{\epsilon\overline{E}}{1-\beta}\right), \tag{22}$$

where $\|\cdot\|$ denotes the Euclidean norm of the parameter vector. Moreover, as $\epsilon \to 0$, Θ_t converges to Θ^* at a geometric rate.

[Parameter Estimation Error] For each neuron i and class k, define the estimation error of mean and variance at iteration t as:

$$\delta\mu_i^k(t) = \mu_i^k(t) - \mu_i^{k,*}, \quad \delta\sigma_i^k(t) = \sigma_i^k(t) - \sigma_i^{k,*}. \tag{23}$$

The total parameter error is:

$$\Delta(t) = \sum_{i=1}^N \sum_{k=1}^C \left[(\delta\mu_i^k(t))^2 + (\delta\sigma_i^k(t))^2\right]. \tag{24}$$

Lemma 4.1 (Bounded Update Noise of $\mu_i^k(t)$) *Let $\tilde{y}_b$ and y_b^* denote the pseudo-label and ground-truth label of the b-th sample in a batch respectively. $\eta_\mu^k(t)$ denotes the update noise of $\mu_i^k(t)$:*

$$\begin{aligned}\eta_\mu^k(t) =& (1-\beta)\frac{1}{B}\sum_{b=1}^B E_i(\hat{m}_{i,b}, \tilde{y}_b)\mathbb{I}(\tilde{y}_b = k)\hat{m}_{i,b} - \\ & (1-\beta)\frac{1}{B}\sum_{b=1}^B E_i(\hat{m}_{i,b}, y_b^*)\mathbb{I}(y_b^* = k)\hat{m}_{i,b},\end{aligned} \tag{25}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Then $\eta_\mu(t)$ satisfies $|\eta_\mu(t)| \leq \overline{E} \cdot M \cdot \epsilon$, where $M = \sup_b |\hat{m}_{i,b}|$ (bounded by signal propagation rules).

[**Proof of Lemma 4.1**] For any sample b, $\hat{m}_{i,b}$ is bounded by M . With $E_i \leq \overline{E}$, we have:

$$|\eta_\mu(t)^k| \leq (1-\beta)\frac{1}{B}\sum_{b=1}^{B} \overline{E} \cdot |\hat{m}_{i,b}| \cdot \mathbb{I}(\tilde{y}_b = k) \cdot \mathbb{I}(\tilde{y}_b \neq y_b^*)$$
$$\leq \overline{E} \cdot M \cdot \frac{1}{B}\sum_{b=1}^{B} \mathbb{I}(\tilde{y}_b \neq y_b^*)\overline{E} \cdot M \cdot \epsilon (asB \to \infty) \tag{26}$$

Lemma 4.2 (Bounded Update Noise of $\sigma_i^k(t)$) *Let $\eta_\sigma^k(t)$ denotes the update noise of $\sigma_i^k(t)$:*

$$\eta_\sigma^k(t) = (1-\beta)\frac{1}{B}\sum_{b=1}^{B} E_i(\hat{m}_{i,b}, \tilde{y}_b)\mathbb{I}(\tilde{y}_b = k)(\hat{m}_{i,b} - \mu_i^k)^2 -$$
$$(1-\beta)\frac{1}{B}\sum_{b=1}^{B} E_i(\hat{m}_{i,b}, y_b^*)\mathbb{I}(y_b^* = k)(\hat{m}_{i,b} - \mu_i^k)^2, \tag{27}$$

Then $\eta_\sigma^k(t)$satisfies $|\eta_\sigma^k(t)| \leq \overline{E} \cdot M' \cdot \epsilon$, where $M' = \sup_b(\hat{m}_{i,b} - \mu_i^k)^2$.

[**Proof of Lemma 4.2**] For any sample b, $(\hat{m}_{i,b} - \mu_i^k)^2 \leq M'$. With $E_i \leq \overline{E}$, we have:

$$|\eta_\sigma(t)^k| \leq (1-\beta)\frac{1}{B}\sum_{b=1}^{B} \overline{E} \cdot M' \cdot \mathbb{I}(\tilde{y}_b = k) \cdot \mathbb{I}(\tilde{y}_b \neq y_b^*)$$
$$\leq \overline{E} \cdot M' \cdot \frac{1}{B}\sum_{b=1}^{B} \mathbb{I}(\tilde{y}_b \neq y_b^*)\overline{E} \cdot M' \cdot \epsilon (asB \to \infty) \tag{28}$$

Lemma 4.3 (Contraction of Error Expectation for Multi-Class) *The expected total error satisfies:*

$$\mathbb{E}[\Delta(t+1)] \leq \beta^2 \mathbb{E}[\Delta(t)] + K \cdot \epsilon^2, \tag{29}$$

where $K = N \cdot C \cdot \max\left\{(\overline{E}M)^2, (\overline{E}M')^2\right\} \cdot \frac{1+\beta}{1-\beta}$ (constant), N is the number of neurons, and other variables are defined as in Lemma 4.1 and 4.2.

[**Proof of Lemma 4.3**] We first analyze the mean parameter error, then extend to variance error.

Step 1: Mean Parameter Error Recurrence. From the update rule Eq. (16), we have:

$$\mu_i^k(t+1) = \beta\mu_i^k(t) + (1-\beta)\frac{1}{B}\sum_{b=1}^{B} E_i(\hat{m}_{i,b}, \tilde{y}_b)\mathbb{I}(\tilde{y}_b = k)\hat{m}_{i,b}. \tag{30}$$

Substitute $\mu_i^k(t) = \mu_i^{k,*} + \delta\mu_i^k(t)$ and use the stationary condition of true parameters:

$$\mu_i^{k,*} = \beta\mu_i^{k,*} + (1-\beta)\frac{1}{B}\sum_{b=1}^{B} E_i(\hat{m}_{i,b}, y_b^*)\mathbb{I}(y_b^* = k)\hat{m}_{i,b}. \tag{31}$$

Subtracting these two equations gives the error recurrence:

$$\delta\mu_i^k(t+1) = \beta\delta\mu_i^k(t) + \eta_\mu^k(t), \tag{32}$$

where $\eta_\mu(t)$ is the update noise from Lemma 4.1. Taking the square and expectation:

$$\mathbb{E}\left[(\delta\mu_i^k(t+1))^2\right] = \beta^2\mathbb{E}\left[(\delta\mu_i^k(t))^2\right] + 2\beta\mathbb{E}\left[\delta\mu_i^k(t) \cdot \eta_\mu^k(t)\right] + \mathbb{E}\left[(\eta_\mu^k(t))^2\right]. \tag{33}$$

Step 2: Bounding Cross and Noise Terms. By Cauchy-Schwarz inequality:

$$\left|\mathbb{E}\left[\delta\mu_i^k(t) \cdot \eta_\mu^k(t)\right]\right| \leq \sqrt{\mathbb{E}\left[(\delta\mu_i^k(t))^2\right]} \cdot \sqrt{\mathbb{E}\left[(\eta_\mu^k(t))^2\right]}. \tag{34}$$

From Lemma 4.1, $\mathbb{E}\left[(\eta_\mu(t))^2\right] \leq \left(\overline{E}M \cdot \epsilon\right)^2$. We prove $\mathbb{E}\left[(\delta\mu_i^k(t))^2\right] \leq D$ (uniformly bounded) by induction. Firstly, in the base case ($t = 0$), initial parameters are finite, so $\mathbb{E}\left[(\delta\mu_i^k(0))^2\right] \leq D_0$ (constant). In the inductive step, assume $\mathbb{E}\left[(\delta\mu_i^k(t))^2\right] \leq D$, then:

$$\mathbb{E}\left[(\delta\mu_i^k(t+1))^2\right] \leq \beta^2 D + 2\beta\sqrt{DC_\mu} + C_\mu. \tag{35}$$

,where $C_\mu = (\overline{E}M \cdot \epsilon)^2$. Setting $D = \frac{C_\mu}{(1-\beta)^2}$ (positive solution) satisfies the inequality, so $\sqrt{\mathbb{E}\left[(\delta\mu_i^k(t))^2\right]} \leq \frac{\sqrt{C_\mu}}{1-\beta}$.

Substituting back into Eq.(33), we have:

$$\mathbb{E}\left[(\delta\mu_i^k(t+1))^2\right] \leq \beta^2\mathbb{E}\left[(\delta\mu_i^k(t))^2\right] + 2\beta \cdot \frac{C_\mu}{1-\beta} + C_\mu$$
$$= \beta^2\mathbb{E}\left[(\delta\mu_i^k(t))^2\right] + \frac{1+\beta}{1-\beta}.C_\mu \tag{36}$$

Step 3: Extending to Variance and Total Error. For variance error $\delta\sigma_i^k(t)$, the same derivation gives:

$$\mathbb{E}\left[(\delta\sigma_i^k(t+1))^2\right] \leq \beta^2\mathbb{E}\left[(\delta\sigma_i^k(t))^2\right] + \frac{1+\beta}{1-\beta}C_\sigma, \tag{37}$$

where $C_\sigma = (\overline{E}M' \cdot \epsilon)^2$. Summing over all i and k:

$$\mathbb{E}[\Delta(t+1)] \leq \beta^2\mathbb{E}[\Delta(t)] + NC \cdot \frac{1+\beta}{1-\beta} \cdot \max\{C_\mu, C_\sigma\}.$$

Defining $K = N \cdot C \cdot \max\left\{(\overline{E}M)^2, (\overline{E}M')^2\right\} \cdot \frac{1+\beta}{1-\beta}$ completes the proof.

[**Proof of Theorem 4**] We proceed in two key steps to establish convergence and error bounds.

Step 1: Solving the Error Recurrence. From Lemma 4.3, the expected total error follows a contractive linear recurrence:

$$\mathbb{E}[\Delta(t+1)] \leq \beta^2 \mathbb{E}[\Delta(t)] + K \cdot \epsilon^2. \tag{38}$$

Unfolding the recurrence (summing the geometric series):

$$\mathbb{E}[\Delta(t)] \leq \beta^{2t}\Delta(0) + K \cdot \epsilon^2 \sum_{s=0}^{t-1} (\beta^2)^s. \tag{39}$$

Since $\beta \in (0,1)$, the geometric series converges to $\sum_{s=0}^{\infty}(\beta^2)^s = \frac{1}{1-\beta^2}$. Thus:

$$\mathbb{E}[\Delta(t)] \leq \beta^{2t}\Delta(0) + \frac{K\epsilon^2}{(1-\beta^2)}. \tag{40}$$

As $t \to \infty$, $\beta^{2t} \to 0$, so $\mathbb{E}[\Delta(t)]$ converges to the bounded value $\frac{K\epsilon^2}{(1-\beta^2)}$.

Step 2: Almost Sure Convergence. By Markov's inequality, for any $\delta > 0$:

$$\mathbb{P}(\Delta(t) > \delta) \leq \frac{\mathbb{E}[\Delta(t)]}{\delta} \to \frac{K\epsilon^2}{\delta(1-\beta^2)}. \tag{41}$$

Choosing $\delta > \frac{K\epsilon^2}{(1-\beta^2)}$ implies $\mathbb{P}(\Delta(t) > \delta) \to 0$ as $t \to \infty$. By the Borel-Cantelli lemma, the events $\{\Delta(t) > \delta\}$ occur only finitely often a.s., so $\Delta(t)$ converges a.s. to a set $\Theta^\dagger$ with:

$$\|\Theta^\dagger - \Theta^*\| = \sqrt{\Delta^\dagger} \leq \sqrt{\frac{K\epsilon^2}{(1-\beta^2)}} = O\left(\frac{\epsilon\overline{E}}{1-\beta}\right). \tag{42}$$

Convergence Rate For $\epsilon = 0$ (perfect pseudo-labels), $\eta_\mu(t) = \eta_\sigma(t) = 0$, so $\Delta(t) = \beta^{2t}\Delta(0)$ (geometric convergence). For $\epsilon > 0$, the limit error bound is proportional to ϵ, so higher pseudo-label error slows convergence and expands the error floor.

9. Derivation of Equation (9)

We present a detailed derivation of the fuzzy memory distribution $Q(\hat{m}_i|y=k)$ in Equation (9), which is obtained by convolving the memory signal distribution $Pr(\hat{m}|y=k)$ with the Gaussian blur kernel $g(\hat{m}, \hat{m}_i)$. For brevity, we use $\hat{\sigma}_i^k$ to denote$(\sigma_i^k)^2$, and use $\hat{\sigma}_1$ to denote $(\sigma_1)^2$

9.1. Preliminaries and Notations

We first clarify the mathematical formulations of key components involved in the derivation:

1. *Memory signal distribution:* For the i-th neuron and k-th class, the memory signal $\hat{m}$ follows a Gaussian distribution $N(\mu_i^k, {\hat{\sigma_i}}^k)$, where μ_i^k and ${\hat{\sigma_i}}^k$ denote the mean and variance, respectively:

$$Pr(\hat{m}|y=k) = \frac{1}{\sqrt{2\pi{\hat{\sigma_i}}^k}} \exp\left(-\frac{(\hat{m}-\mu_i^k)^2}{2{\hat{\sigma_i}}^k}\right) \tag{43}$$

2. *Gaussian blur kernel:* The kernel function for smoothing memory signals is defined as (without normalization factor, consistent with the main text):

$$g(\hat{m}, \hat{m}_i) = \exp\left(-\frac{(\hat{m}-\hat{m}_i)^2}{2\hat{\sigma}_1}\right) \tag{44}$$

   where $\hat{\sigma}_1$ is the bandwidth parameter of the Gaussian kernel, and $\hat{m}_i$ is the memory signal of the i-th neuron for the current input.

3. *Convolution integral definition:* The fuzzy memory distribution $Q(\hat{m}_i|y=k)$ is formally defined as the convolution of $Pr(\hat{m}|y=k)$ and $g(\hat{m}, \hat{m}_i)$:

$$Q(\hat{m}_i|y=k) = \int_{-\infty}^{+\infty} Pr(\hat{m}|y=k) \cdot g(\hat{m}, \hat{m}_i) d\hat{m} \tag{45}$$

9.2. Step 1: Substitute and Combine the Integrand

Substitute $Pr(\hat{m}|y=k)$ and $g(\hat{m}, \hat{m}_i)$ into the convolution integral, and extract constant terms outside the integral sign:

$$\begin{aligned} Q(\hat{m}_i|y=k) = \frac{1}{\sqrt{2\pi{\hat{\sigma_i}}^k}} \int_{-\infty}^{+\infty} \exp(-\frac{(\hat{m}-\mu_i^k)^2}{2{\hat{\sigma_i}}^k} \\ - \frac{(\hat{m}-\hat{m}_i)^2}{2\hat{\sigma}_1}) d\hat{m} \end{aligned} \tag{46}$$

For brevity, we denote the exponent term as $E(\hat{m})$:

$$E(\hat{m}) = -\frac{1}{2}\left[\frac{(\hat{m}-\mu_i^k)^2}{{\hat{\sigma_i}}^k} + \frac{(\hat{m}-\hat{m}_i)^2}{\hat{\sigma}_1}\right] \tag{47}$$

9.3. Step 2: Complete the Square for the Exponent Term

Expand the quadratic terms in $E(\hat{m})$:

$$\begin{aligned} (\hat{m}-\mu_i^k)^2 &= \hat{m}^2 - 2\mu_i^k\hat{m} + (\mu_i^k)^2, \\ (\hat{m}-\hat{m}_i)^2 &= \hat{m}^2 - 2\hat{m}_i\hat{m} + (\hat{m}_i)^2 \end{aligned} \tag{48}$$

Substitute these expansions back into $E(\hat{m})$ and group like terms (i.e., $\hat{m}^2$, $\hat{m}$, and constant terms):

$$\begin{aligned} E(\hat{m}) = -\frac{1}{2}[\hat{m}^2\left(\frac{1}{{\hat{\sigma_i}}^k} + \frac{1}{\hat{\sigma}_1}\right) - 2\hat{m}\left(\frac{\mu_i^k}{{\hat{\sigma_i}}^k} + \frac{\hat{m}_i}{\hat{\sigma}_1}\right) \\ + \left(\frac{(\mu_i^k)^2}{{\hat{\sigma_i}}^k} + \frac{(\hat{m}_i)^2}{\hat{\sigma}_1}\right) \end{aligned} \tag{49}$$

We define simplified coefficients to streamline the completing-the-square process: Quadratic coefficient: $a = \frac{1}{\hat{\sigma_i}^k} + \frac{1}{\hat{\sigma}_1} = \frac{\hat{\sigma_i}^k + \hat{\sigma}_1}{\hat{\sigma_i}^k \hat{\sigma}_1}$, Linear coefficient: $b = \frac{\mu_i^k}{\hat{\sigma_i}^k} + \frac{\hat{m}_i}{\hat{\sigma}_1}$, and Constant term: $c = \frac{(\mu_i^k)^2}{\hat{\sigma_i}^k} + \frac{(\hat{m}_i)^2}{\hat{\sigma}_1}$

Applying the completing-the-square method to the quadratic expression inside the brackets:

$$a\hat{m}^2 - 2b\hat{m} + c = a\left(\hat{m} - \frac{b}{a}\right)^2 + \left(c - \frac{b^2}{a}\right) \tag{50}$$

Substitute this back into $E(\hat{m})$:

$$E(\hat{m}) = -\frac{a}{2}\left(\hat{m} - \frac{b}{a}\right)^2 - \frac{1}{2}\left(c - \frac{b^2}{a}\right) \tag{51}$$

9.4. Step 3: Evaluate the Gaussian Integral

Substitute the simplified $E(\hat{m})$ back into the convolution integral, and split the exponent into two separate terms (with the constant part moved outside the integral):

$$\begin{aligned} Q(\hat{m}_i|y=k) = &\frac{1}{\sqrt{2\pi\hat{\sigma_i}^k}} \exp\left(-\frac{1}{2}\left(c - \frac{b^2}{a}\right)\right) \\ &\cdot \int_{-\infty}^{+\infty} \exp\left(-\frac{a}{2}\left(\hat{m} - \frac{b}{a}\right)^2\right) d\hat{m} \end{aligned} \tag{52}$$

We use the standard Gaussian integral formula $\int_{-\infty}^{+\infty} \exp(-px^2)dx = \sqrt{\frac{\pi}{p}}$ (by substituting $x = \hat{m} - \frac{b}{a}$ and $p = \frac{a}{2}$):

$$\int_{-\infty}^{+\infty} \exp\left(-\frac{a}{2}x^2\right) dx = \sqrt{\frac{2\pi}{a}} \tag{53}$$

Substitute the integral result into the expression and simplify the constant term:

$$Q(\hat{m}_i|y=k) = \frac{1}{\sqrt{2\pi\hat{\sigma_i}^k}} \exp\left(-\frac{1}{2}\left(c - \frac{b^2}{a}\right)\right) \cdot \sqrt{\frac{2\pi}{a}} \tag{54}$$

Cancel out $\sqrt{2\pi}$ and further simplify:

$$Q(\hat{m}_i|y=k) = \frac{1}{\sqrt{a\hat{\sigma_i}^k}} \exp\left(-\frac{1}{2}\left(c - \frac{b^2}{a}\right)\right) \tag{55}$$

9.5. Step 4: Simplify the Constant Term in the Exponent

Substitute a, b, c back into $c - \frac{b^2}{a}$ and simplify. First, expand b^2:

$$\begin{aligned} b^2 &= \left(\frac{\mu_i^k}{\hat{\sigma_i}^k} + \frac{\hat{m}_i}{\hat{\sigma}_1}\right)^2 \\ &= \frac{(\mu_i^k)^2\hat{\sigma}_1^2 + 2\mu_i^k\hat{m}_i\hat{\sigma_i}^k\hat{\sigma}_1 + (\hat{m}_i)^2{\sigma_i^k}^2}{{\sigma_i^k}^2\hat{\sigma}_1^2} \end{aligned} \tag{56}$$

Substitute $a = \frac{\hat{\sigma_i}^k + \hat{\sigma}_1}{\hat{\sigma_i}^k\hat{\sigma}_1}$ and simplify the fraction $\frac{b^2}{a}$:

$$\frac{b^2}{a} = \frac{(\mu_i^k)^2\hat{\sigma}_1 + 2\mu_i^k\hat{m}_i\hat{\sigma_i}^k + (\hat{m}_i)^2\hat{\sigma_i}^k}{\hat{\sigma_i}^k + \hat{\sigma}_1} \tag{57}$$

Combine this with c and simplify (noting that cross terms cancel out):

$$c - \frac{b^2}{a} = \frac{(\hat{m}_i - \mu_i^k)^2}{\hat{\sigma_i}^k + \hat{\sigma}_1} \tag{58}$$

9.6. Step 5: Final Simplification

Substitute $a = \frac{\hat{\sigma_i}^k + \hat{\sigma}_1}{\hat{\sigma_i}^k\hat{\sigma}_1}$ into the constant term:

$$\frac{1}{\sqrt{a\hat{\sigma_i}^k}} = \frac{1}{\sqrt{\frac{\hat{\sigma_i}^k + \hat{\sigma}_1}{\hat{\sigma_i}^k\hat{\sigma}_1} \cdot \hat{\sigma_i}^k}} = \frac{\sqrt{\hat{\sigma}_1}}{\sqrt{\hat{\sigma_i}^k + \hat{\sigma}_1}} \tag{59}$$

To match the compact form in the main text, we adopt a notation simplification by redefining $\hat{\sigma_i}^k \to 2\hat{\sigma_i}^k$ (scaling the variance by 2, which preserves the mathematical nature of the Gaussian distribution). This leads to:

$$\frac{\sqrt{\hat{\sigma}_1}}{\sqrt{2\hat{\sigma_i}^k + \hat{\sigma}_1}} \cdot \frac{1}{2} = \frac{\sqrt{\hat{\sigma}_1}}{2\sqrt{2\hat{\sigma_i}^k + \hat{\sigma}_1}} \tag{60}$$

and the exponent term becomes:

$$\exp\left(-\frac{(\hat{m}_i - \mu_i^k)^2}{2\hat{\sigma_i}^k + \hat{\sigma}_1}\right) \tag{61}$$

9.7. Final Result

Combining all the above steps, we derive the fuzzy memory distribution as presented in Equation (9):

$$Q(\hat{m}_i|y=k) = \frac{\sqrt{\hat{\sigma}_1}}{2\sqrt{2\hat{\sigma_i}^k + \hat{\sigma}_1}} \exp\left(-\frac{(\hat{m}_i - \mu_i^k)^2}{2\hat{\sigma_i}^k + \hat{\sigma}_1}\right) \tag{62}$$

10. More Experimental Results

10.1. Detail Accuracy of Each Task

To align with the evaluation protocol of existing SFUDA methods, the metric for the VisDA-C dataset was modified, which is shown in Table.5. The adaptation performance from the Synthetic to Real task $S \to R$ is evaluated using the mean per-class accuracy, while the reverse adaptation task $R \to S$ is no longer considered.

In addition, the detailed results about the accuracy of each task on Digits and Office-31 ares shown in Table4 and Table.4, respectively. The experimental results demonstrate that MemFlow outperforms both traditional classifiers and the retrain-last-layers approach across all tasks, providing more detailed evidence of its distinct advantages in the DAMap scenario. Furthermore, compared with

Table 4. Accuracy(%) and adaptation time per instance (ms) on the Digits and Office31 datasets, where $\overline{Acc}$ and $\overline{Time}$ are the average accuracy and time cost across all tasks. † means the reproduced results.

Method	$S \to M$	$U \to M$	$M \to U$	$\overline{Acc}(\uparrow)$	$\overline{Time}(\downarrow)$
DAMap Methods					
w/o DA	72.4	87.3	79.2	79.6	-
retrain@last	77.1	92.1	91.6	86.9	1.213
retrain@BLS	80.8	93.9	92.5	89.0	0.063
retrain@KNN	77.6	91.7	92.1	87.1	0.044
retrain@DCT	56.0	66.2	77.4	66.6	0.155
retrain@RF	73.8	88.5	91.2	84.5	0.458
retrain@SVM	75.5	91.6	92.6	86.6	3.720
retrain@BAG	78.4	92.5	92.7	87.8	0.494
retrain@NBY	81.1	89.5	91.8	87.5	0.022
retrain@XGB	73.1	87.4	89.8	83.4	0.451
MemFlow	**82.5**	**93.8**	**91.2**	**89.1**	**0.013**
SFUDA Methods					
SHOT	98.9	97.5	98.0	98.1	0.091
AaD	98.4	96.7	97.9	97.7	0.319
PFC	98.5	97.8	98.0	98.1	1.056
TPDS	98.9	98.0	98.4	98.4	3.362

Method	$A \to D$	$A \to W$	$D \to A$	$D \to W$	$W \to A$	$W \to D$	$\overline{Acc}(\uparrow)$	$\overline{Time}(\downarrow)$
DAMap Methods								
w/o DA	80.5	76.7	60.5	94.7	63.2	98.6	79.0	-
retrain@last	84.3	79.8	60.6	94.2	64.7	99.0	80.4	1.397
retrain@BLS	80.9	79.4	62.1	94.8	66.0	99.4	80.4	0.084
retrain@KNN	90.0	84.8	61.9	95.0	62.8	99.6	82.3	0.046
retrain@DCT	41.4	36.0	21.4	39.9	25.0	53.4	36.2	0.382
retrain@RF	81.5	77.2	56.6	89.7	58.7	99.2	77.2	1.504
retrain@XGB	70.9	64.4	39.6	72.0	41.4	88.6	61.4	0.636
retrain@SVM	83.5	78.0	59.9	94.7	62.4	99.2	79.6	0.473
retrain@BAG	87.3	87.0	62.3	95.6	64.1	99.4	82.6	0.753
retrain@NBY	84.3	81.6	63.3	92.6	65.3	97.4	80.8	0.058
MemFlow	**92.2**	**88.9**	**68.2**	**97.5**	**70.4**	**99.4**	**86.1**	**0.160**
SFUDA Methods								
SHOT	94.0	90.1	74.7	98.4	74.9	99.9	88.7	4.682
AaD	96.4	92.1	75.0	99.1	76.5	100.0	89.9	2.656
PFC	97.3	94.0	75.6	99.2	76.6	100.0	90.5	3.901
TPDS	97.1	94.5	75.7	98.7	75.5	99.8	90.2	25.230

Table 5. Accuracy(%) and adaptation time per instance (ms) on the VisDA-C dataset, where $\overline{Acc}$ and $\overline{Time}$ are the average accuracy and time cost across all task. † means the reproduced results.

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	$\overline{Acc}(\uparrow)$	$\overline{Time}(\downarrow)$
DAMap Methods														
w/o DA	60.9	21.6	50.9	67.6	65.8	6.3	82.2	23.2	57.3	30.6	84.6	8.0	46.6	-
retrain@last	65.9	17.5	73.0	68.1	70.4	27.6	86.3	23.1	78.1	45.9	79.8	14.1	54.1	2.375
retrain@BLS	63.8	1.6	61.1	74.0	69.3	1.2	83.6	0.2	96.8	10.9	80.2	1.9	45.4	0.096
retrain@KNN	70.4	29.0	77.8	72.2	73.6	58.7	92.1	17.0	64.4	14.6	80.6	5.8	54.7	0.045
retrain@DCT	41.8	6.5	58.9	36.5	30.4	18.4	39.9	13.8	25.7	19.4	64.1	20.2	31.3	0.21
retrain@RF	60.0	13.8	76.7	58.7	44.4	29.7	77.8	16.5	40.1	46.4	81.0	13.9	46.6	0.602
retrain@XGB	52.0	10.1	73.3	56.9	45.2	23.1	70.6	19.5	39.9	36.5	80.9	22.2	44.2	5.695
retrain@SVM	63.4	17.1	73.0	63.9	62.1	53.1	85.3	20.2	34.9	38.7	82.3	19.0	51.1	13.235
retrain@BAG	66.9	19.4	73.9	78.1	75.8	60.0	93.1	15.1	66.7	3.2	80.2	3.8	53.0	2.018
retrain@NBY	64.5	58.0	81.6	51.7	68.3	17.0	52.5	57.6	64.6	40.1	69.6	0.4	52.2	0.026
MemFlow	**68.4**	**56.0**	**75.0**	**67.1**	**80.3**	**69.3**	**83.5**	**44.3**	**70.7**	**12.9**	**80.3**	**16.9**	**60.4**	**0.012**
SFUDA Methods														
SHOT	94.3	88.5	80.1	57.3	93.1	94.9	80.7	80.3	91.5	89.1	86.3	58.2	82.9	5.251
AaD	97.4	90.5	80.8	76.2	97.3	96.1	89.8	82.9	95.5	93	92	64.7	88.0	3.060
PFC	91.3	83.7	77.1	49.8	89.7	88.7	79.8	78.9	87.6	84.9	80.1	57.9	79.1	3.766
TPDS	97.6	91.5	89.7	83.4	97.5	96.3	92.2	82.4	96.0	94.1	90.9	40.4	87.6	67.288

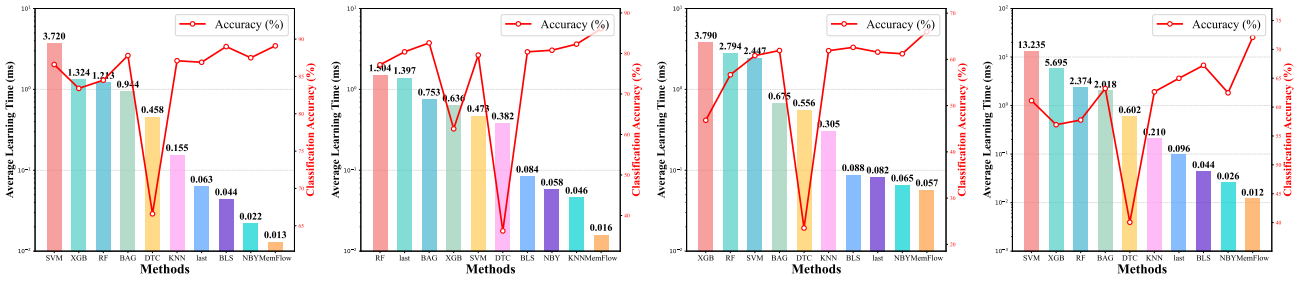
SFUDA methods, MemFlow achieves comparable performance while reducing time consumption by a significant margin. This remarkable performance effectively validates the feasibility of rapid domain adaptation on edge devices.

To intuitively demonstrate the performance and efficiency advantages of MemFlow in the DAMap scenario, Figure 6 summarizes the accuracy and time consumption across all datasets. The results show that MemFlow significantly surpasses all other classifiers. It achieves a substantial increase in accuracy while drastically reducing the time required compared to both the traditional classifier and the retrain-last-layers approach. Notably, MemFlow is even twice as fast as the lightweight retrain@NBY model, indicating its strong potential for real-time domain adaptation tasks.

10.2. MemFlow in SFUDA Setting

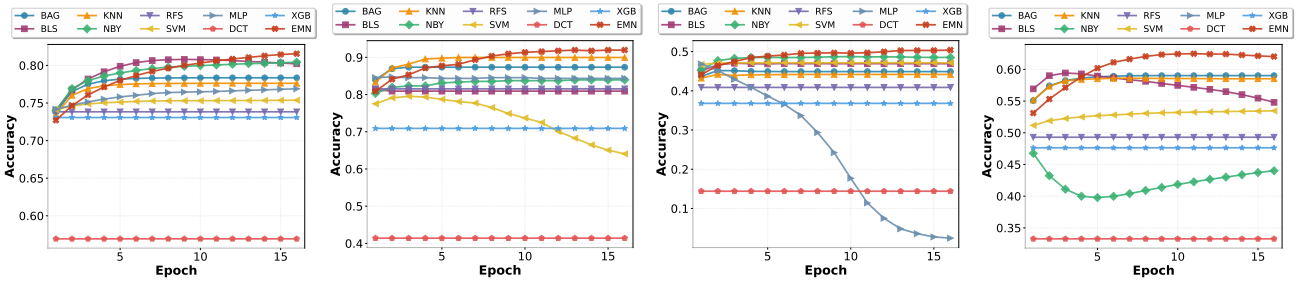
To demonstrate the effectiveness of MemFlow in the SFUDA scenario, we designed it as a plug-and-play module that can be integrated into existing SFUDA methods. Since MemFlow does not require gradient backpropagation, it serves as an instructor that assigns confidence scores to the pseudo-labels generated by the deep model. These confidence scores are then used to modulate the degree of gradient updates for each sample, thereby reducing the negative impact of incorrect pseudo-labels on the model.

Specifically, after the backbone network is pre-trained on the source domain, MemFlow rapidly memorizes the features extracted by the backbone, as detailed in Section 3.4. During training on the target domain, given the output features of deep neural model, it generates self-



(a) Digits (b) Office31 (c) Office-Home (d) VisDA-C

Figure 6. Illustration of Accuracy and Time cost per instance across different methods on DAMap in all datasets.



(a) $S \rightarrow M$ in Digits Dataset (b) $A \rightarrow D$ in Office31 Dataset (c) $A \rightarrow D$ in Office-Home Dataset (d) $S \rightarrow R$ in VisDA-C Dataset

Figure 7. The accuracy trend on each epochs of domain adaptation in the different tasks with various datasets dataset.

supervised prediction of samples as confidence distributions α_{Conf}. Based on the confidence distribution α_{Conf}, the downstream classification loss can be reformulated as the weighted loss: $L_{Conf} = \alpha_{Conf} * L$.

In this way, MemFlow leverages its ability to capture feature distribution shifts to assign reliability to pseudo-labels, effectively mitigating the influence of erroneous pseudo-labels.

Furthermore, MemFlow is updated only during the pseudo label generation process of the original SFUDA method. Through employing the Reinforced Memorization strategy described in Section 3.6, the mean μ and variance σ of neurons in MemFlow are updated via momentum to adapt to the target domain distribution.

It is worth noting that since the AaD method does not originally include a pseudo-label generation step, integrating MemFlow requires adding such a process, leading to additional time overhead. In contrast, for other SOTA methods that already include pseudo-label generation, the inclusion of MemFlow introduces only a minimal time increase, approximately 1 ms.

10.3. Accuracy in Different Epochs of DAMap

Fig. 7 further depicts the accuracy of the models over the epochs of learning pseudo labels on the target domain on each datasets. In all cases, MemFLow can improve the performance steadily, which benefits from the confidence-based parameter updating in Section 3.6. Notably, it can also be observed that only retraining last layers (retrain@last) faces a significant drop in performance in $A \rightarrow D$ of Office-Home Dataset, which may be caused by the accumulated errors of noisy pseudo labels. While the propose MemFLow can adaptively update the parameter to avoid the misleading of noisy pseudo labels.

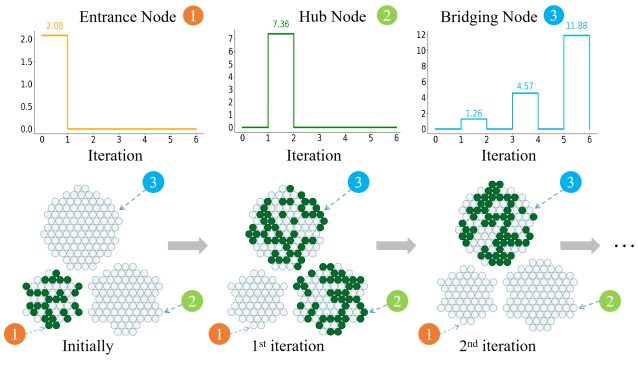
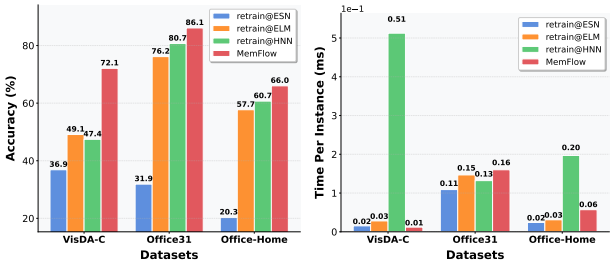


Figure 8. The change of the activation states and the output signals $o_{i,t}$ of the neurons in different iterations of signal propagation. For simplicity, we only show the nodes here while ignoring the connections. The green nodes indicate the activated ones with non-zero output.

Table 6. Comparison with SNN Method in Office-Home Dataset, where $\overline{Acc}$ are the average accuracy across all task

	$A \to C$	$A \to P$	$A \to R$	$C \to A$	$C \to P$	$C \to R$	$P \to A$	$P \to C$	$P \to R$	$R \to A$	$R \to C$	$R \to P$	$\overline{Acc}(\uparrow)$
LIF[2]	45.4	68.4	74.1	52.2	62.0	64.6	53.5	42.1	73.1	65.6	50.0	77.4	60.7
SRM[1]	40.4	63.6	68.0	44.7	58.2	60.2	44.8	37.9	66.9	60.2	45.2	75.0	55.4
ALIF[3]	42.3	65.3	71.4	46.1	59.0	61.3	47.1	38.9	68.9	61.3	45.6	76.2	57.0
MemFlow+LIF	46.1	63.2	67.4	48.1	59.1	63.8	50.4	47.3	76.5	65.8	51.7	77.0	59.7
MemFlow+SRM	22.8	16.3	59.0	23.3	5.5	47.8	14.0	47.3	76.5	41.6	33.2	60.9	37.3
MemFlow+ALIF	36.2	52.1	48.7	30.9	29.9	46.5	25.0	47.3	76.5	47.7	31.3	48.1	43.3
MemFlow	**50.4**	**76.5**	**76.9**	**59.3**	**71.1**	**69.7**	**59.9**	**47.3**	**76.5**	**69.5**	**53.9**	**81.0**	**66.0**



(a) Accuracy (b) Time Per Instance

Figure 9. Comparison with Gradient-free Nerual Network in VisDA-C, Office31 and Office-Home Dataset.

10.4. Performance of Spiking Mechanism

The Spiking mechanism of MemFlow is illustrated in Fig. 8. In contrast to traditional spiking neural networks, our spiking mechanism differs in that its output amplitude is dynamically tied to the magnitude of the positive hidden state (instead of fixed binary spikes), enabling fine-grained encoding of signal strength. Furthermore, MemFlow integrates an explicit cumulative memory signal that transforms transient spiking activity into persistent traces, which yields a long-term memory to support retrieval of input-related information.

To further demonstrate the advantage of the proposed spiking procedure in MemFlow, We supply the comparison between the proposed model and other spiking neural networks, such as Leaky Integrate and Fire (LIF)[2], Adaptive Leaky Integrate and Fire (ALIF)[3] and Spike Response Model (SRM) [1]. Further, we replace the original spiking method in MemFlow with other spiking neural network. The experiments results are shown in Tab.6, which confirms the superior performance of the specific spiking design in MemFlow.

10.5. Comparison With Gradient-free nerual networks

In this section, we compared the performance of the proposed MemFlow against several Gradient-free nerual networks:

- retrain@ELM is a single-hidden-layer feedforward Extreme Learning Machine[6] network that uses random, fixed weights for the hidden layer and only trains the output weights analytically.
- retrain@ESN[7] is a recurrent Echo State Network with a fixed, randomly connected reservoir of neurons whose dynamic state encodes the input history, and only a simple linear readout layer is trained.
- retrain@HNN is a Hopfield Network[4][5] that functions as an associative memory by converging to a stable state closest to a given input pattern.

The experimental results, summarized in Fig.9, demonstrate the superior Performance of MemFlow. Crucially, while the computational time of MemFlow is comparable to that of these highly efficient models, it achieves a significant accuracy across all benchmark datasets. This key advantage stems from the fundamental difference in learning paradigm: whereas ELM, ESN, and Hopfield are fundamentally designed to fit a static input-output mapping function, MemFlow memorizes the associations within distributed neurons. This architecture enables a dynamic and reinforced memorization process, especially within the unlabeled target domain, allowing it to refine its internal representations and achieve greater generalization without a commensurate increase in computational overhead.

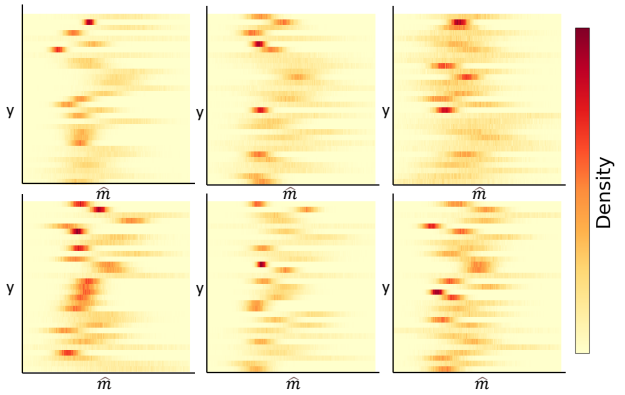


Figure 10. Visualization of the memories on six randomly chosen neurons after training on the Office-31 dataset

10.6. Visualization

Fig. 10 depicts the memory units of randomly selected neurons, revealing various captured-distribution across different neurons in MemFlow. This diversity suggests that the different nodes exhibit specialized roles during the memory storage.

References

[1] Gerstner, Wulfram. Time-dependent renewal theory and its application to neurons. *Neural Computation*, 7(5):851–869, 1995. 17

[2] Knight, Bruce W. Dynamics of encoding in a population of neurons. *The Journal of General Physiology*, 59(6):734–766, 1972. 17

[3] Brette, Romain and Gerstner, Wulfram. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642, 2005. 17

[4] Hopfield, John J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. 17

[5] Hopfield, John J. and Tank, David W. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, 1985. 17

[6] Huang, Guang-Bin and Zhu, Qin-Yu and Siew, Chee-Kheong. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE International Joint Conference on Neural Networks (IJCNN)*, volume 2, pages 985–990. IEEE, 2004. 17

[7] Jaeger, Herbert and Haas, Harald. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004. 17