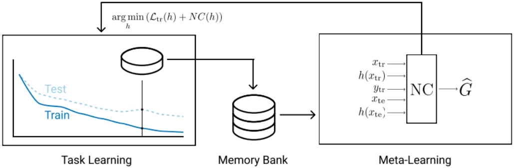
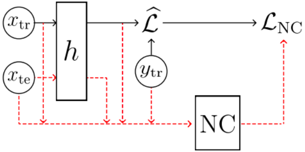
Meta-learning to Predict Generalization

Reference - *Neural Complexity Measures*

During meta-learning tasks include train and test data: the gap between train/test loss is available. NC trains another network to predict this generalization gap



$$\mathcal{L}_{NC} \propto \|\mathcal{L}_{Test} - \mathcal{L}_{Train} - NC(H)\|$$
$$Q = f(X_{te}), K = f(X_{tr}), V = [K; Y_{tr}]$$
$$NC(H) \equiv NC(X_{tr}, X_{te}, Y_{tr}, h(X_{tr}), h(X_{te})) =$$
$$\frac{1}{m'} \sum_{i=1}^{m'} g(A); \text{ where } A = \frac{Softmax(QK^T)}{\sqrt{d}} V$$

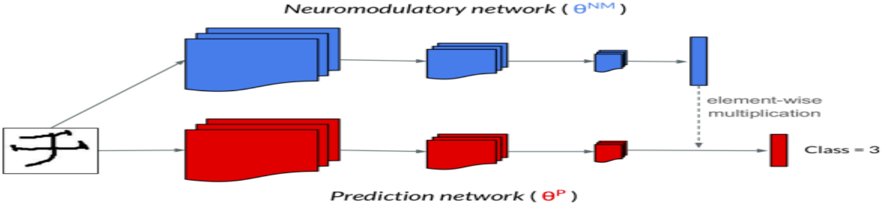


NC-regularized loss: $\mathcal{L}_{reg} = \mathcal{L}_{Train} + \lambda NC(H)$; λ increased gradually over tasks.

Continual Learning: Neuro-modulated Meta-learning

Reference - *Learning to Continually Learn*

(i) Outer-loop loss on meta-train test set and samples D_R from tasks seen so far
(ii) Multiplicative modulation of prediction network by a modulatory network.



Meta-training:

Sample D_{Train}, D_{Test}, D_R; $\theta_0^P = \theta^P$

for $i \in 0 \ldots k$,

$$\Delta\theta_i^P = -\beta \nabla_{\theta_i^P} \mathcal{L}(\theta^{NM}, \theta_i^P, D_{Train})$$

$$\Delta\theta^{NM,P} = -\alpha \nabla_{\theta^{NM,P}} \mathcal{L}(\theta^{NM}, \theta_k^P, D_{Test}, D_R)$$

Meta-testing:

$\mathcal{T}_{Test} -= (D_{Train}, D_{Test}) \sim \mathcal{T}$

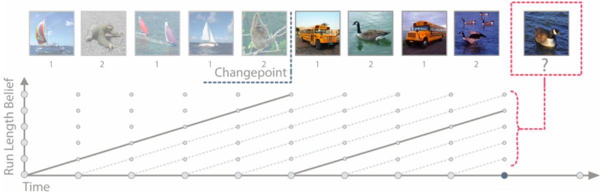
for $i \in 0 \ldots k$,

$$\Delta\theta^P = -\beta \nabla_{\theta^P} \mathcal{L}(\theta^{NM}, \theta^P, D_{Train})$$

Continual Meta-learning without task boundaries

Reference - *Continual Meta-learning without tasks.*

Task boundaries are unknown and change, but discretely and with probability λ. Algorithm keeps track of $b_t(r)$ its belief that, and $\eta_t[r]$ *adapted* parameters if, the current task has run for r steps, $\forall r \in \{0 \dots t-1\}$. [e.g., $\eta_t[r]$ = hidden representation of a model-based meta-learner using past r observations.]



Adaptation: $p_\theta(\hat{y}_t|x_{1:t}, y_{1:t-1}) = \sum_{r=0}^{t-1} b_t(r) p(\hat{y}_t|x_t, \eta_{t-1}[r])$ & $l_t = NLL(\hat{y}_t, y_t)$
$\hat{b}_t(r) = p(y_t|x_{1:t}, \eta_{t-1}[r]) b_t(r)$, and $b_{t+1}(r)|_{r>0} = (1-\lambda)\hat{b}_t(r-1)$, $b_{t+1}(0) = \lambda$
Update $\eta_t[r]$; and every k steps update θ using $\nabla_\theta \sum_{t-k}^{t} l_t$.