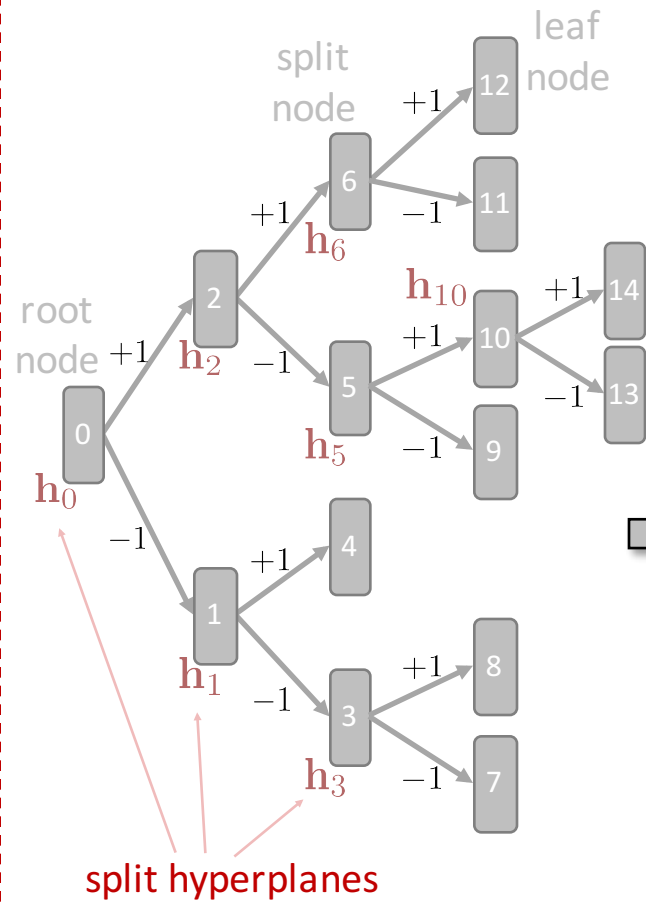
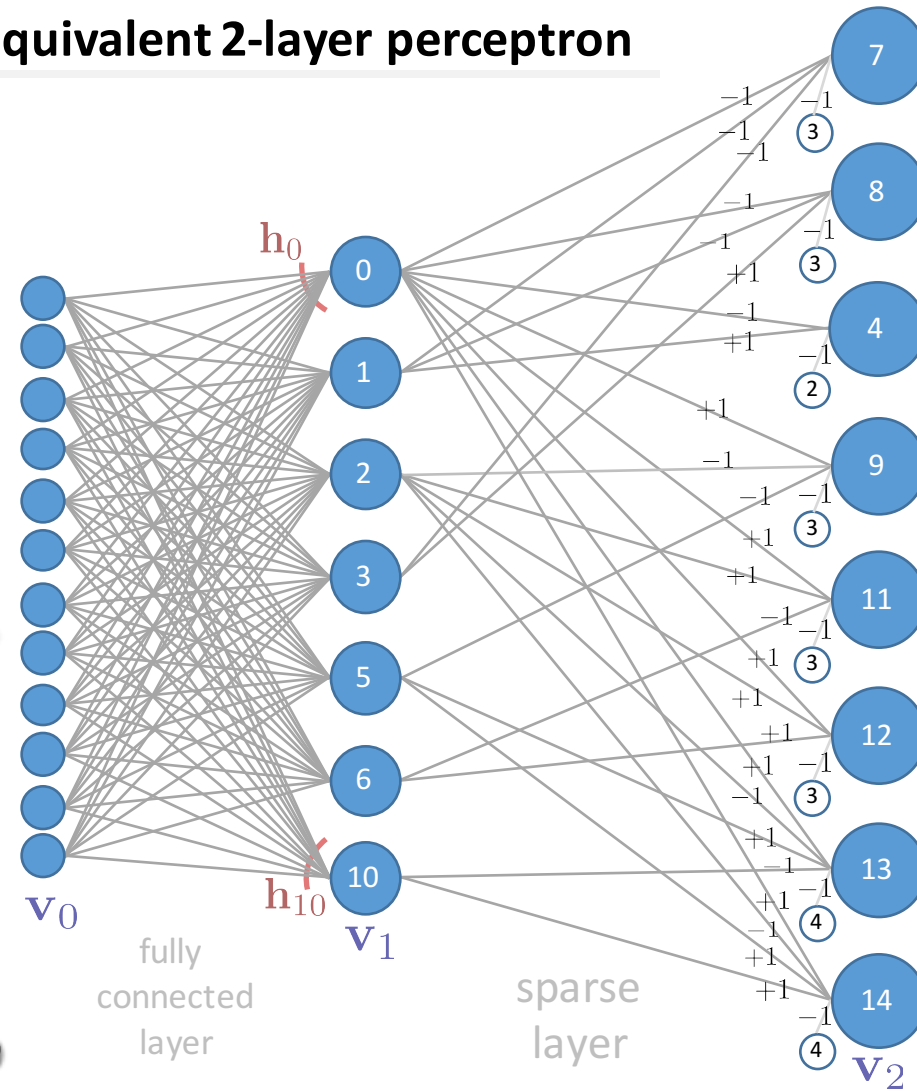


Decision tree



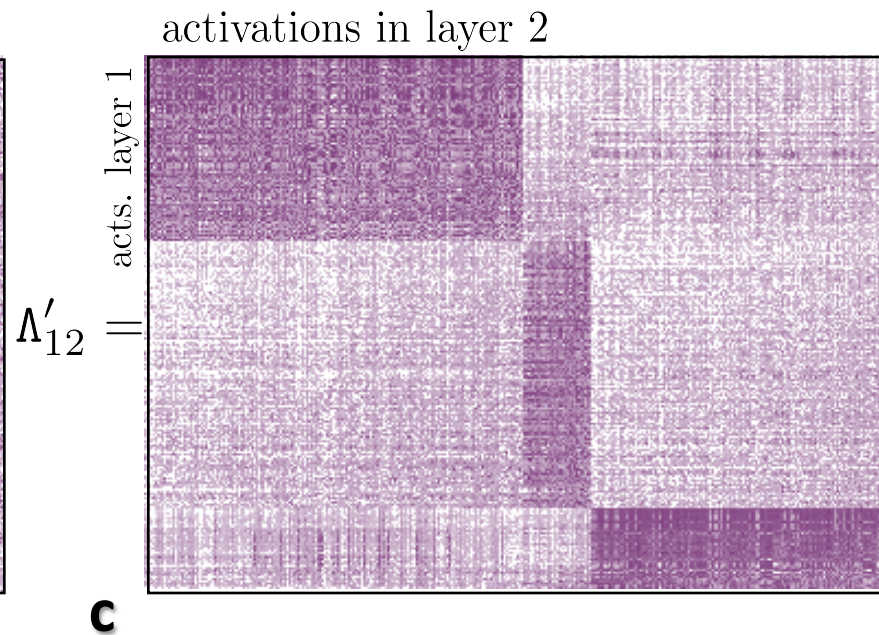
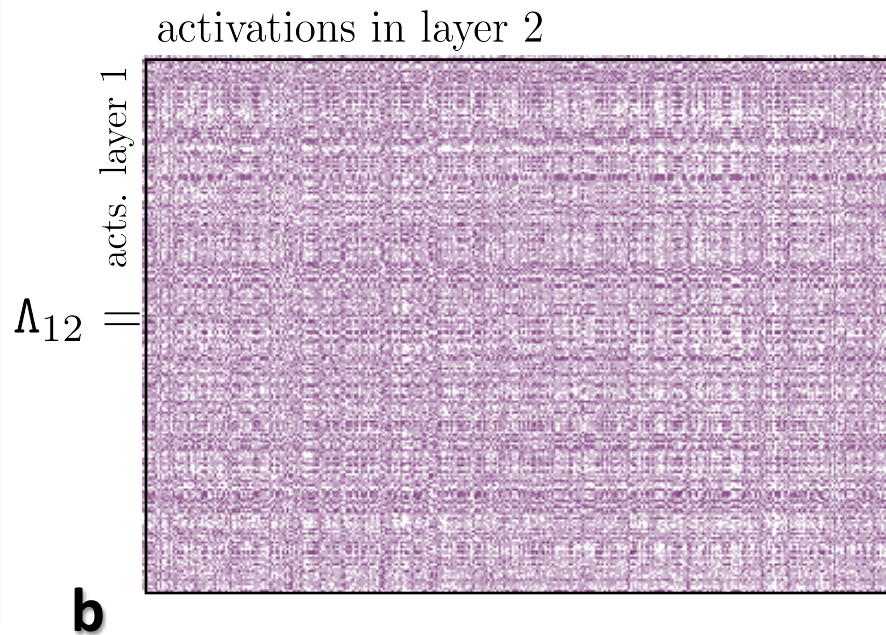
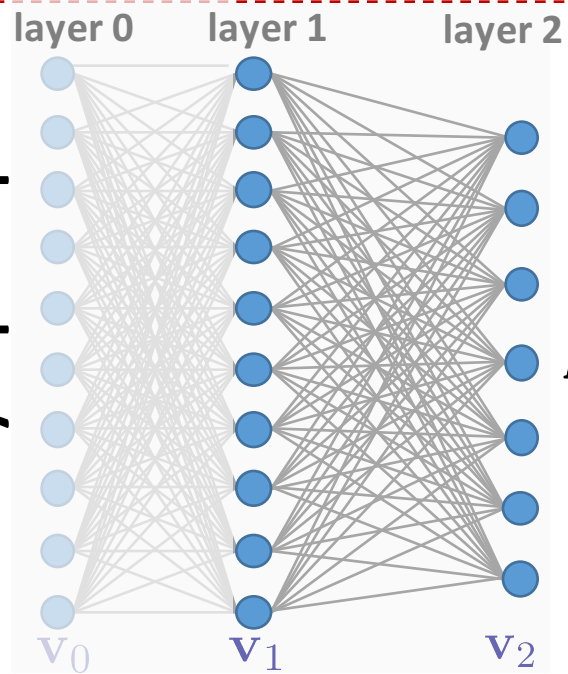
a

Equivalent 2-layer perceptron

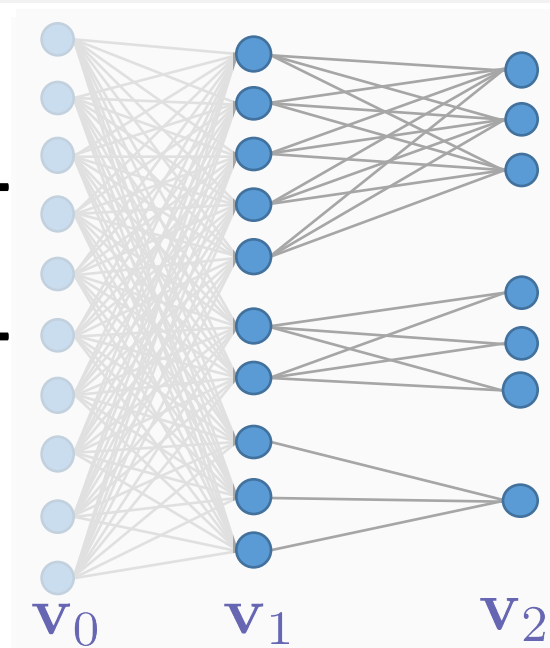


b

a Two-layer perceptron



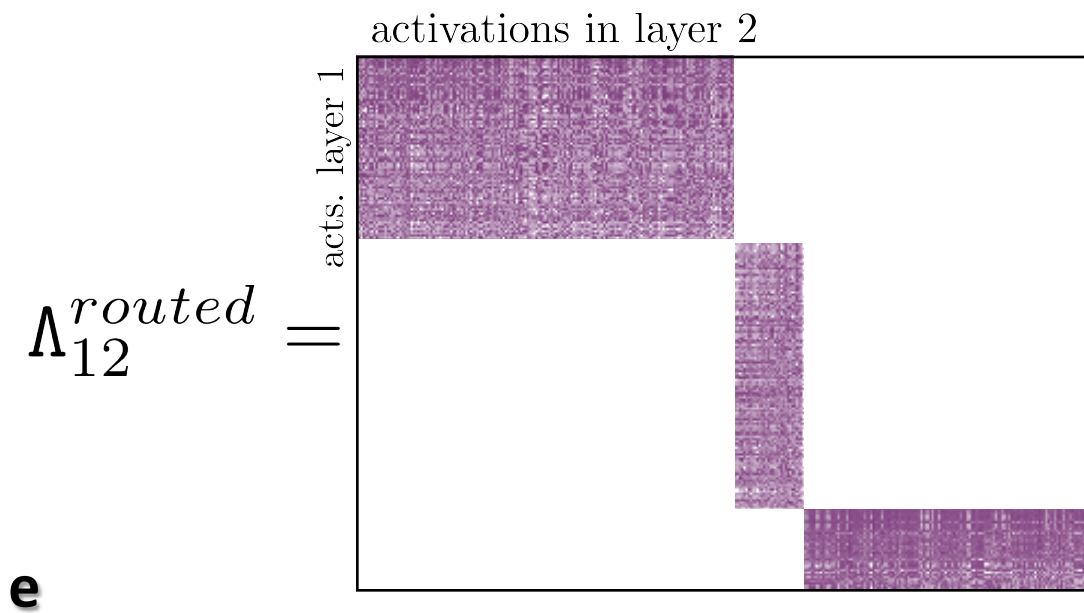
a Routed perceptron



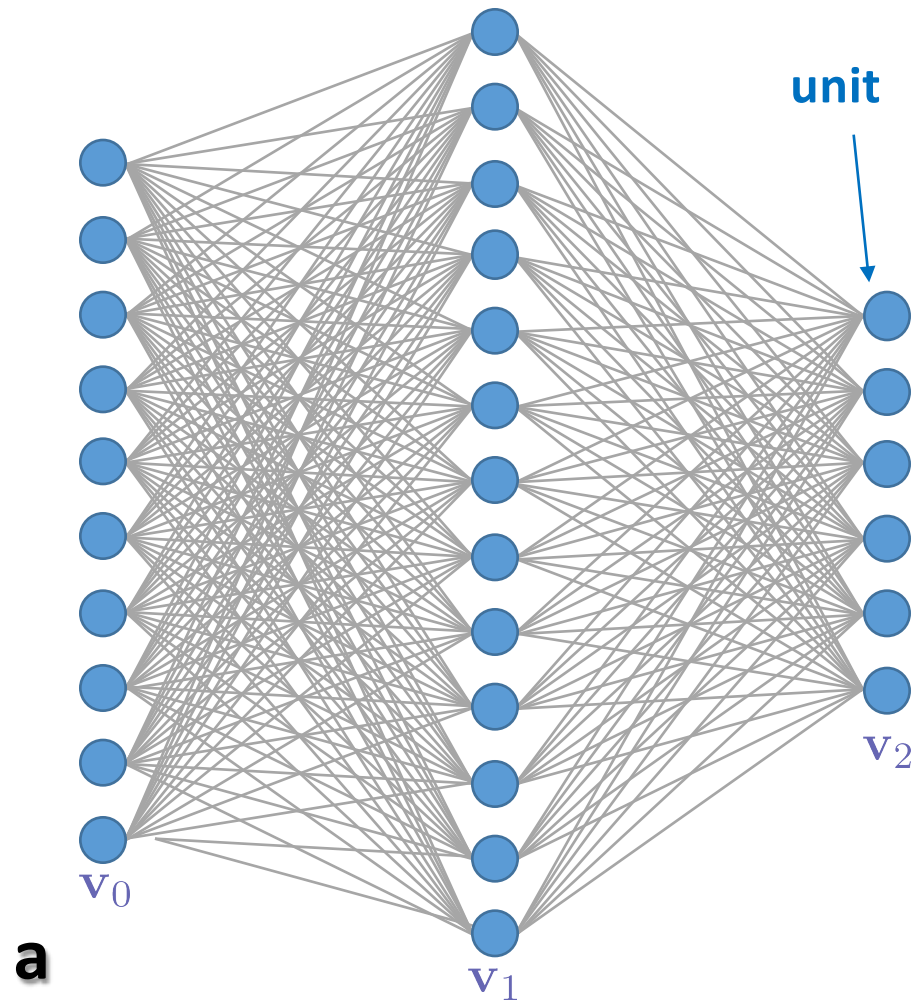
route 1

route 2

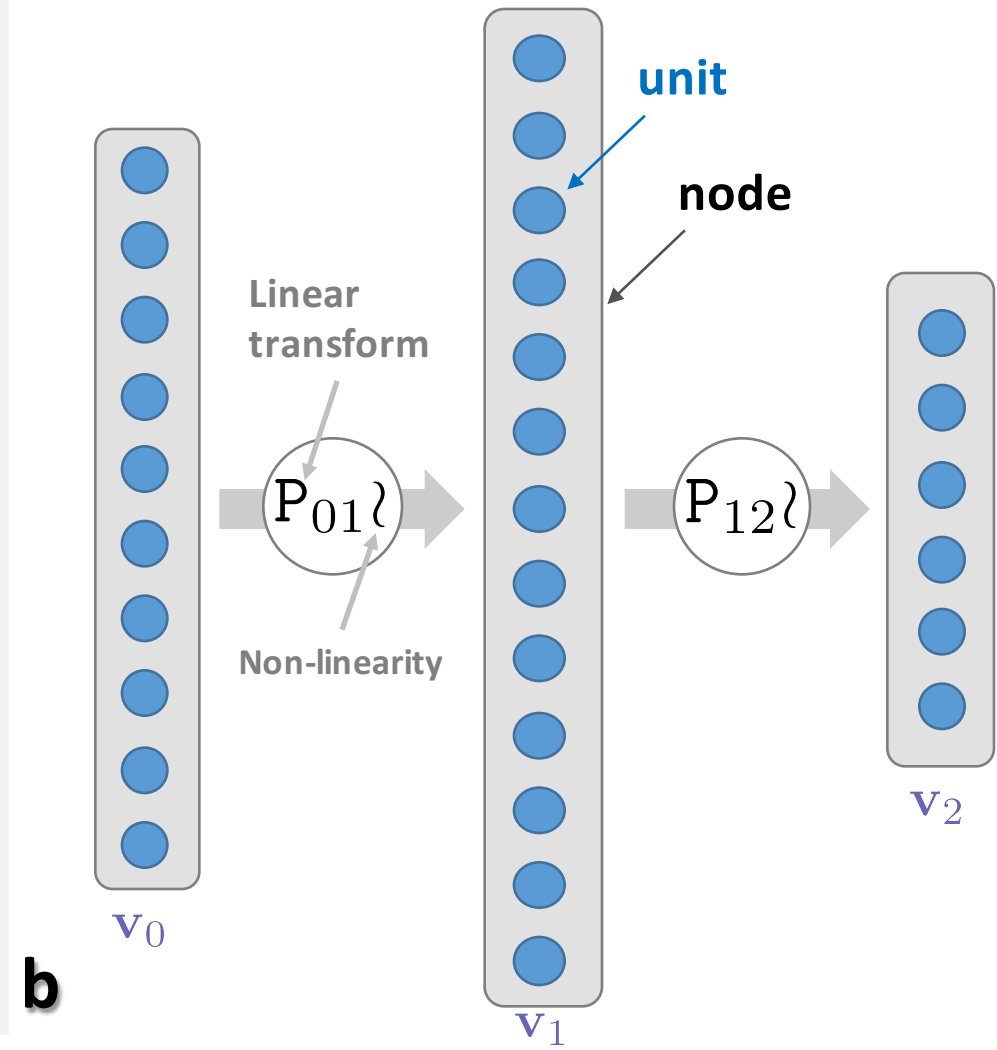
route 3

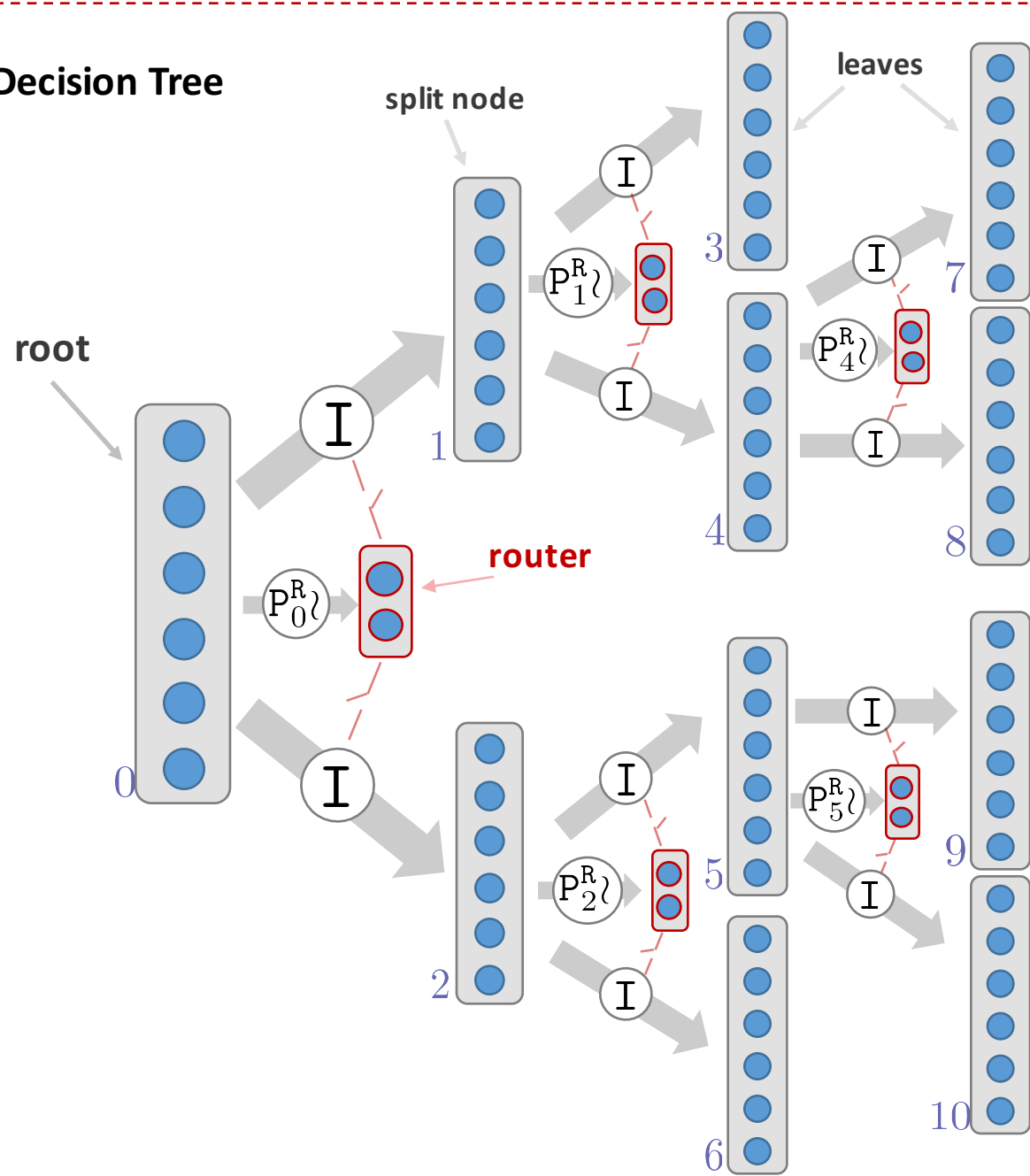


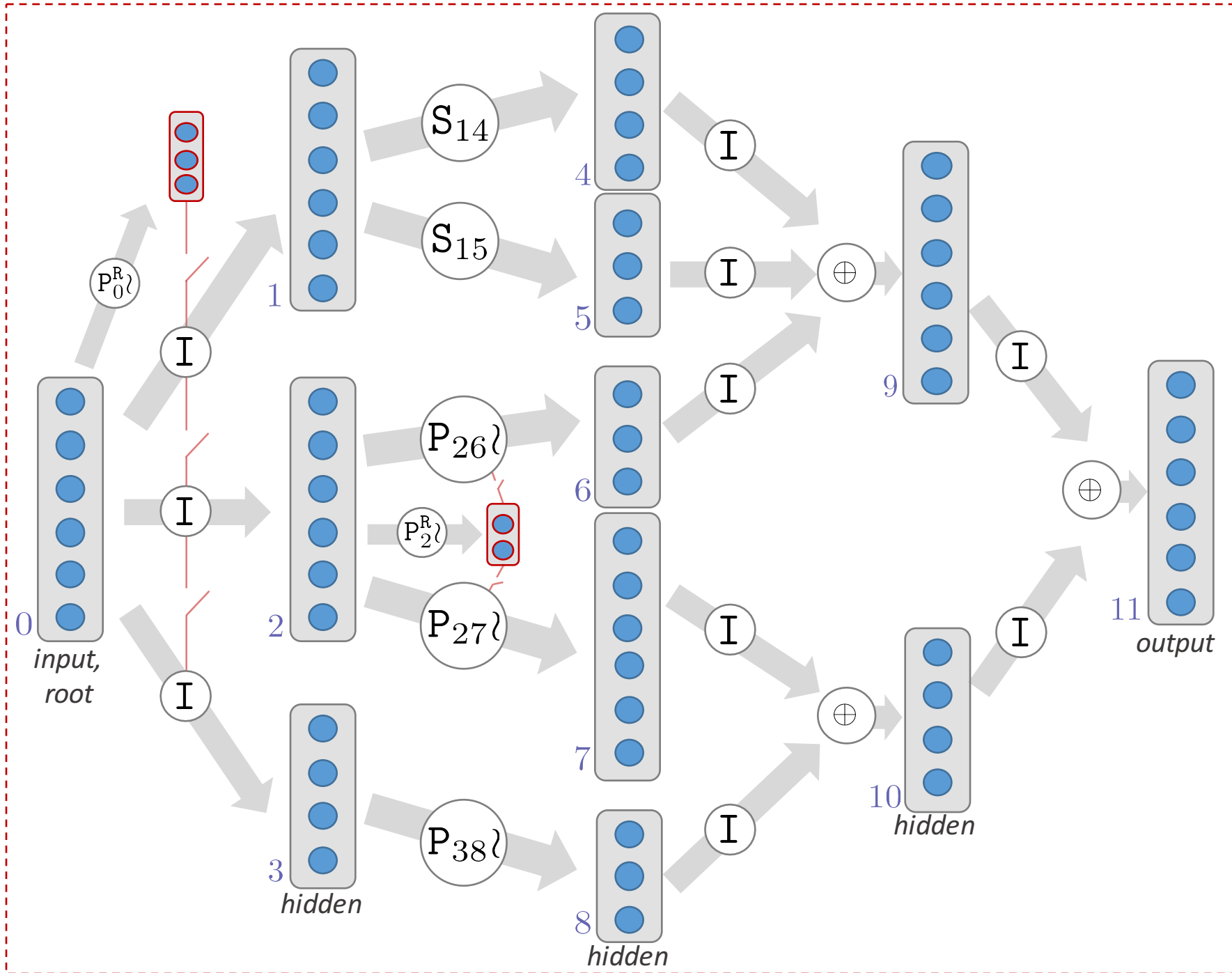
Old notation for a 2-layer perceptron

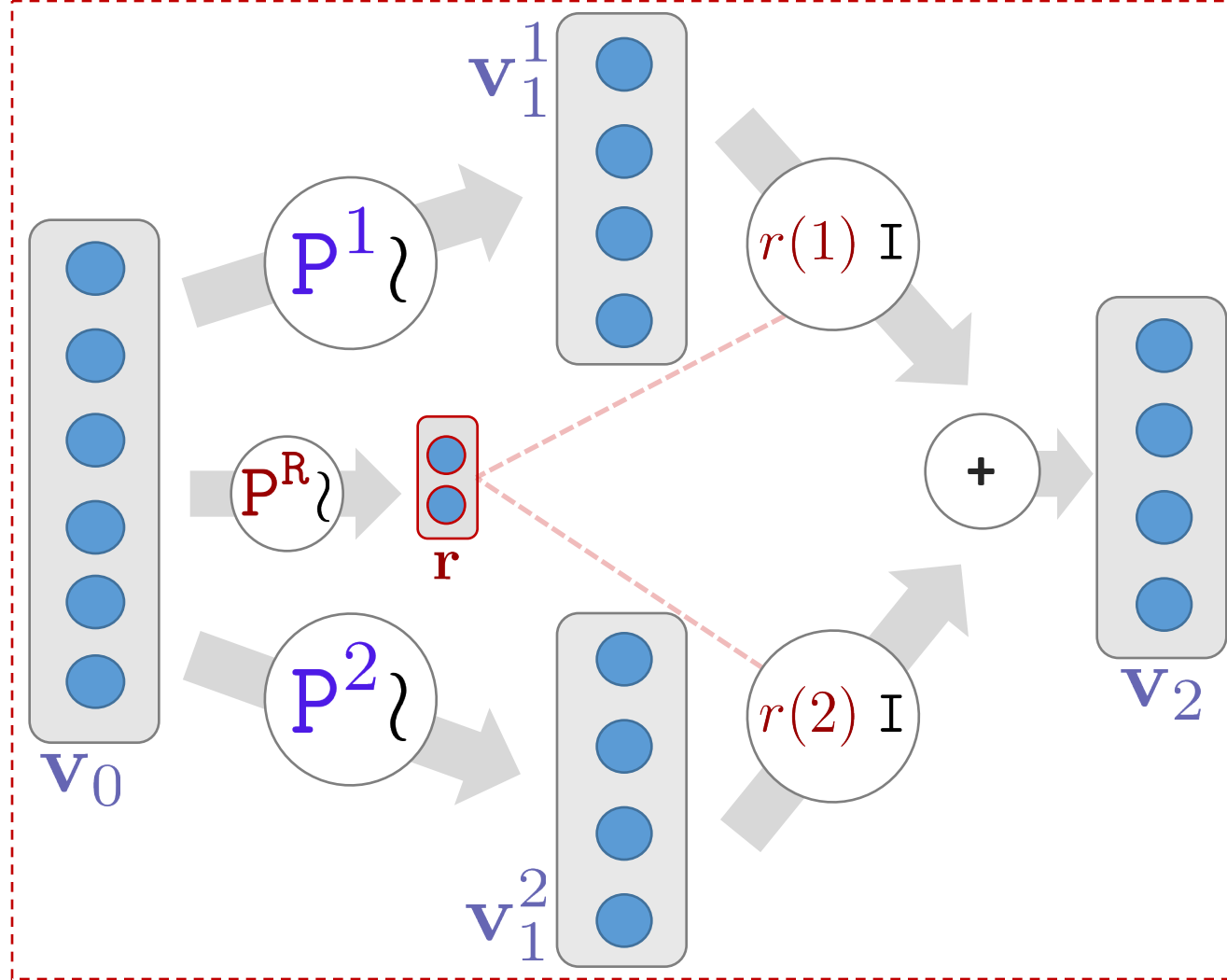


New notation



[illegible]





$$E(\theta) = \frac{1}{2} \sum_i^N \|y_i^* - y_i(\theta)\|^2 \quad \text{The energy to be minimized}$$

$$y(\theta) = r(\theta) Y(\theta) \quad \text{Network's forward mapping}$$

$$Y = \begin{bmatrix} \vdots & \vdots & \vdots \\ - & y^j & - \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \text{Matrix of outputs for all routes}$$

$$y^j = \sigma(P^j x) \quad \text{Intermediate output for j-th route}$$

$$r = \sigma(Rx) \quad \text{Soft routing weights}$$

$$\Delta \theta_{t+1} := -\rho \left. \frac{\partial E}{\partial \theta} \right|_t \quad \text{The parameter update rule}$$

$$\theta := \{R, \{P^j\}\} \quad \text{The parameters to be optimized}$$

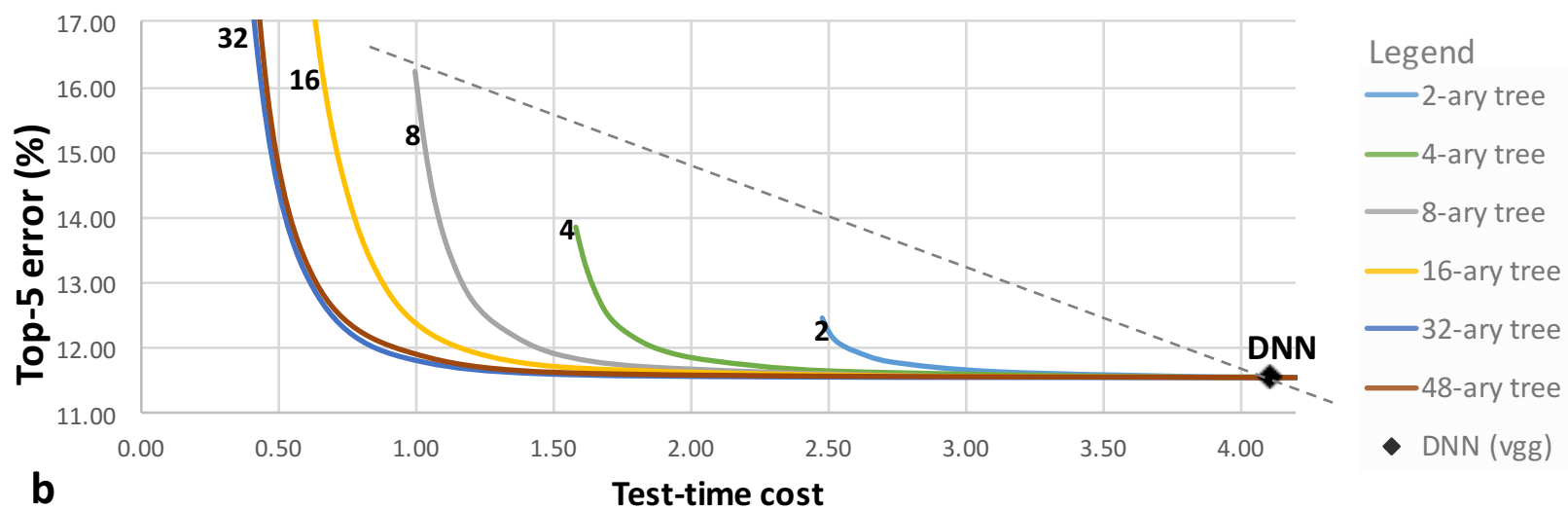
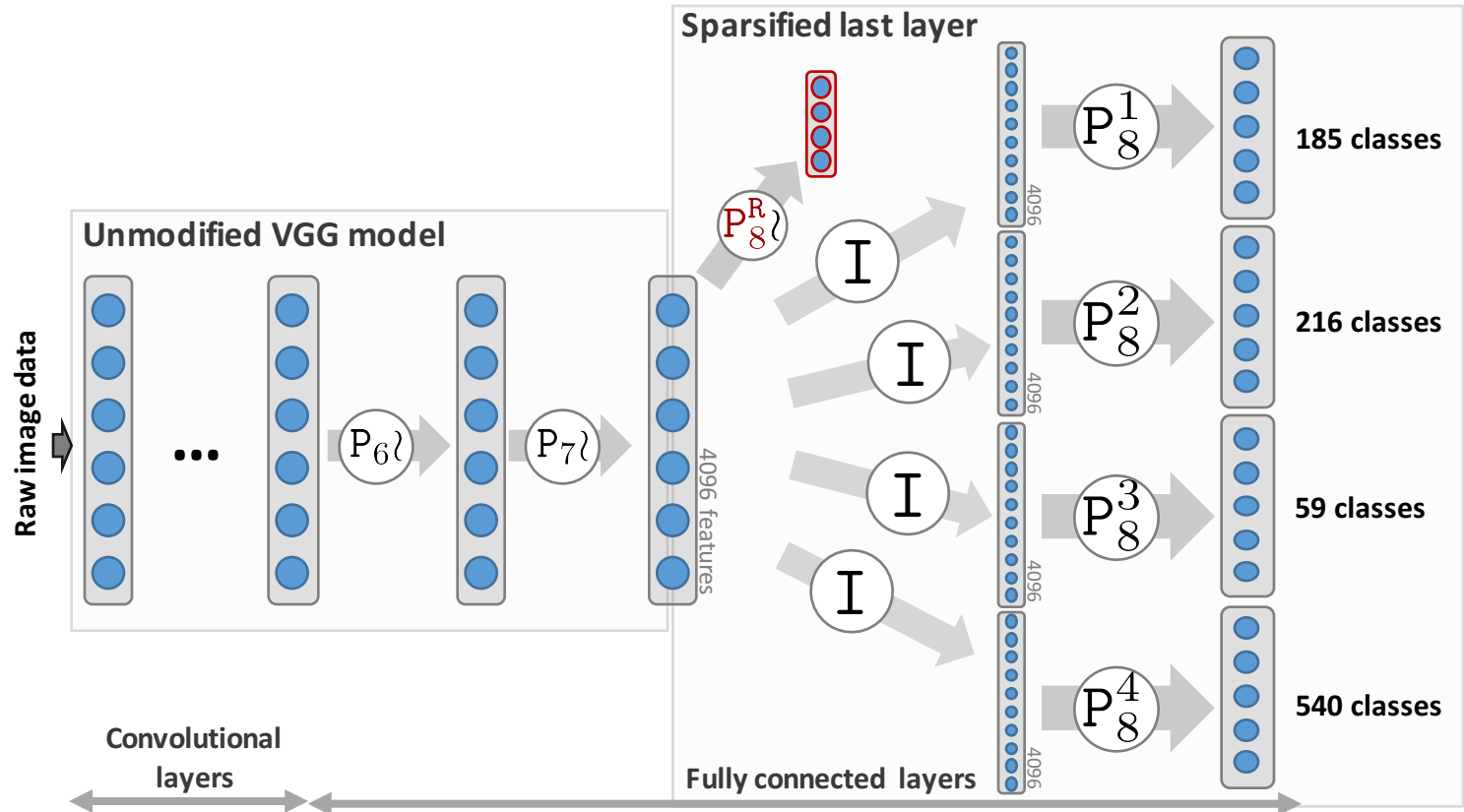
$$\mathcal{T} = \{(x_1, y_1^*), \dots, (x_i, y_i^*), \dots, (x_N, y_N^*)\} \quad \text{The labelled training set}$$

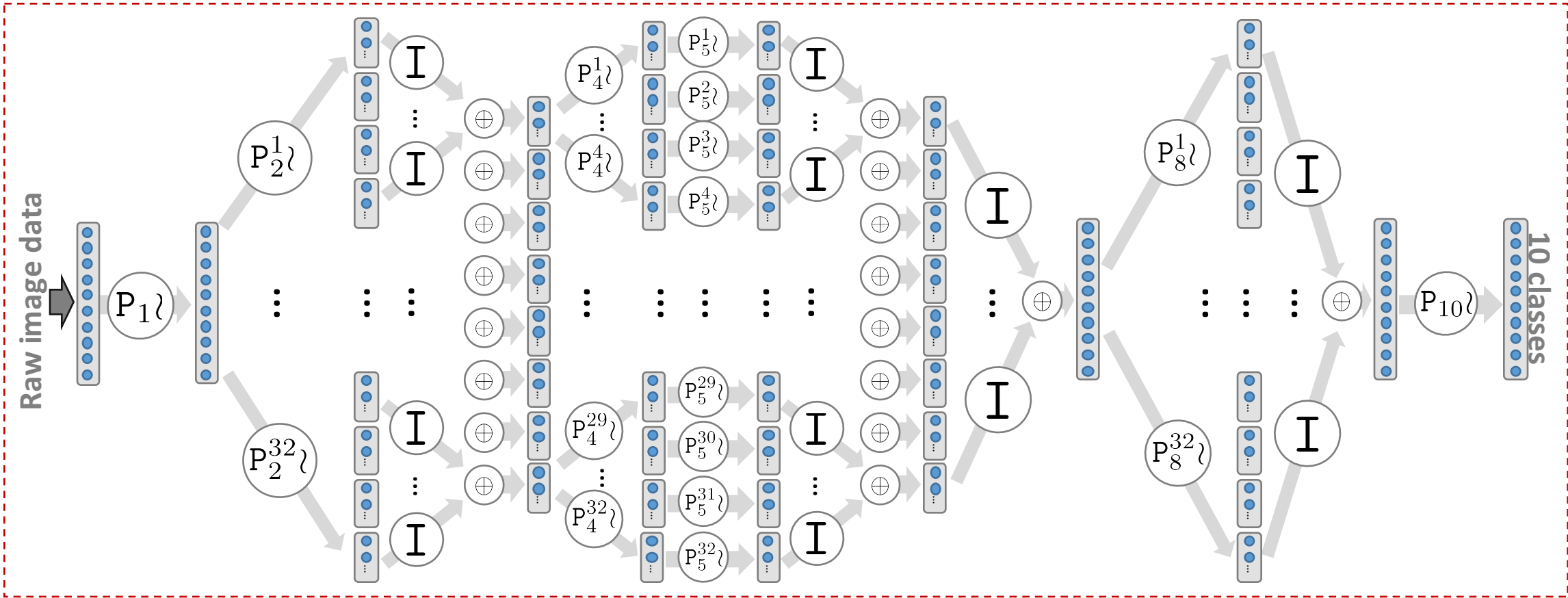
Chain rule to compute partial derivatives for gradient descent

$$\phi^j := P^j x \quad y^j = \sigma(\phi^j)$$

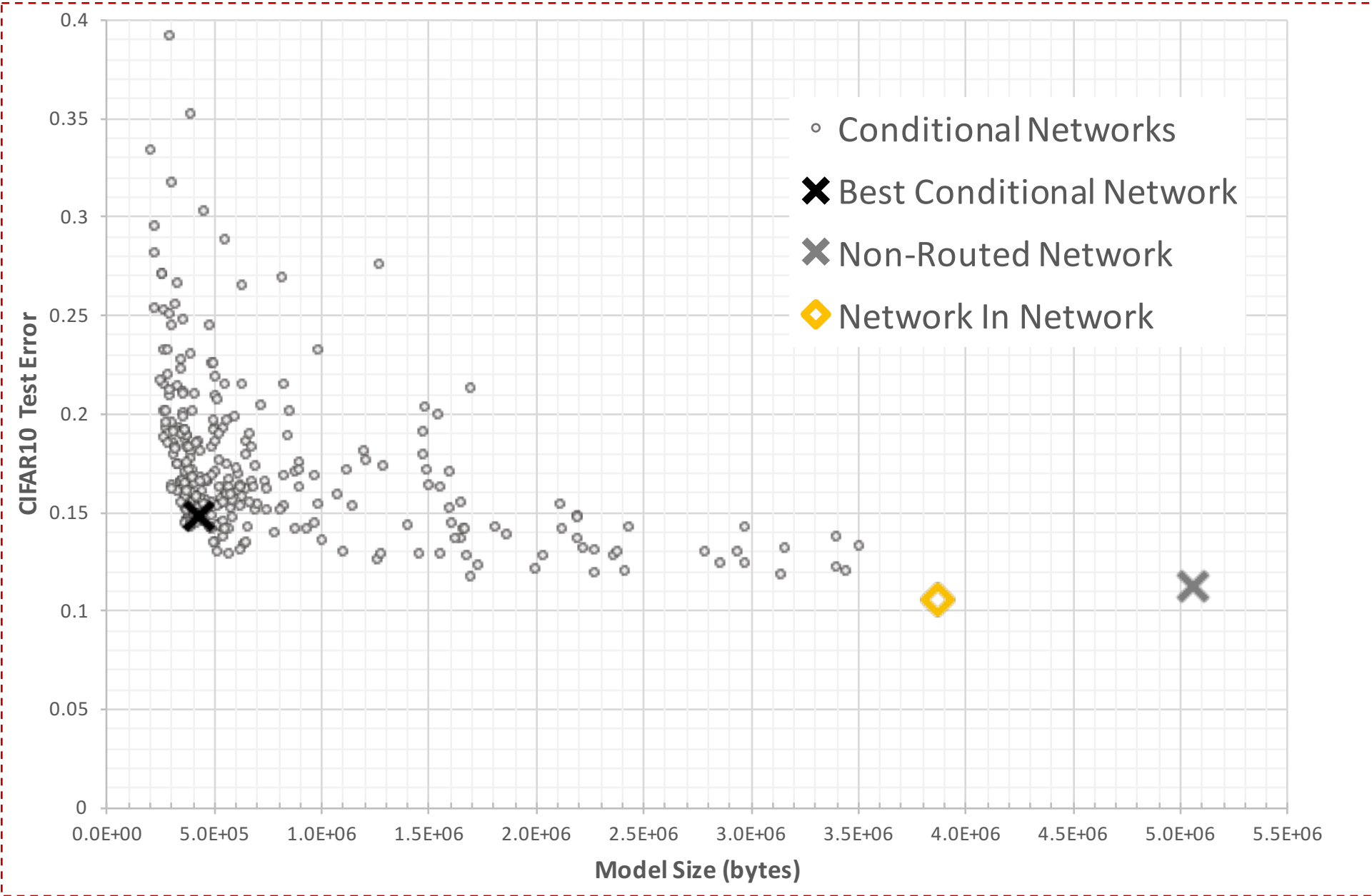
$$\frac{\partial E}{\partial \theta} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial \theta} = \frac{\partial E}{\partial y} \left(\frac{\partial r}{\partial R} Y + \sum_j r(j) \frac{\partial y^j}{\partial \phi^j} \frac{\partial \phi^j}{\partial P^j} \right)$$

Routing weights influence the back-propagating errors differently for different routes

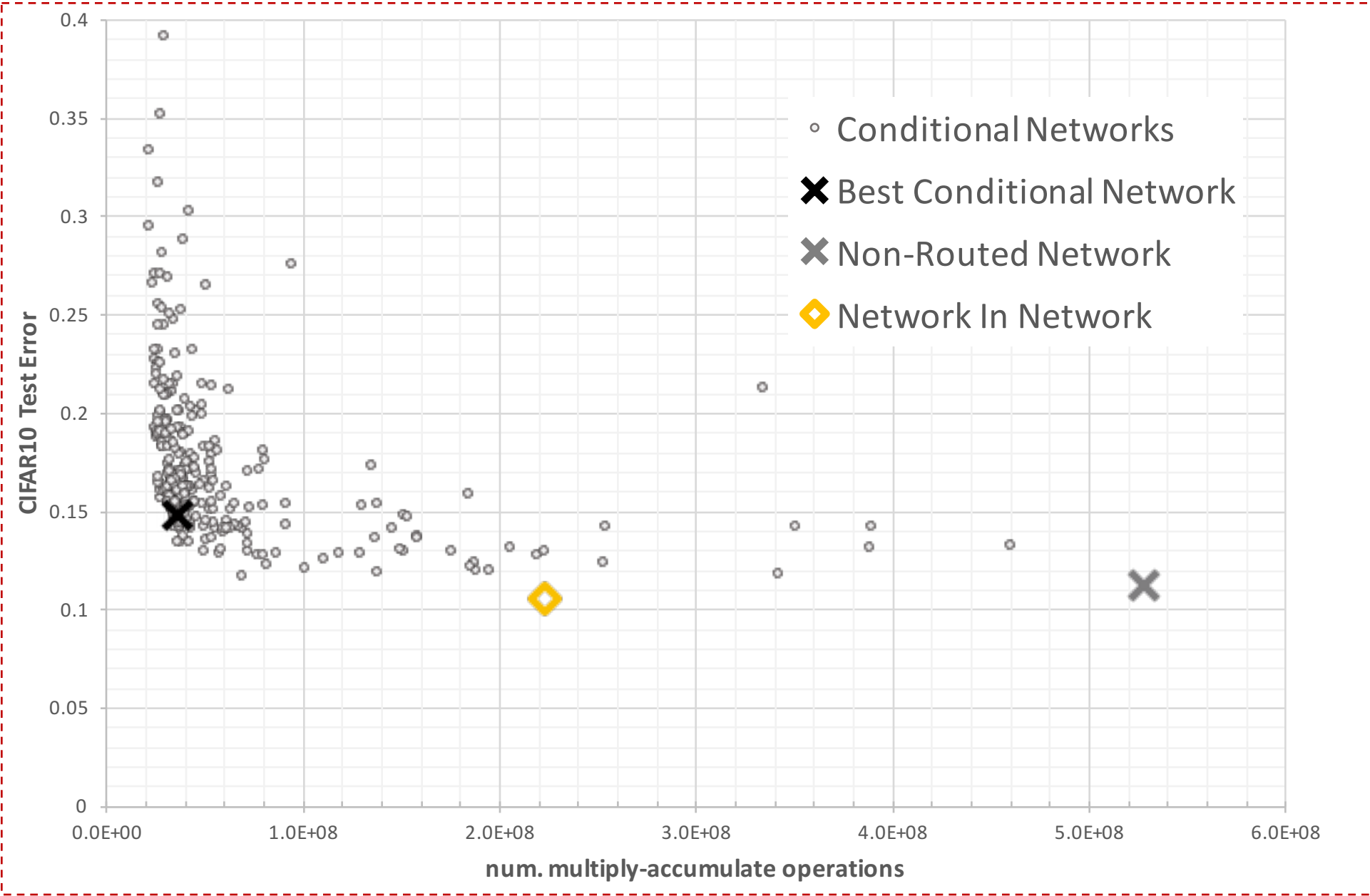




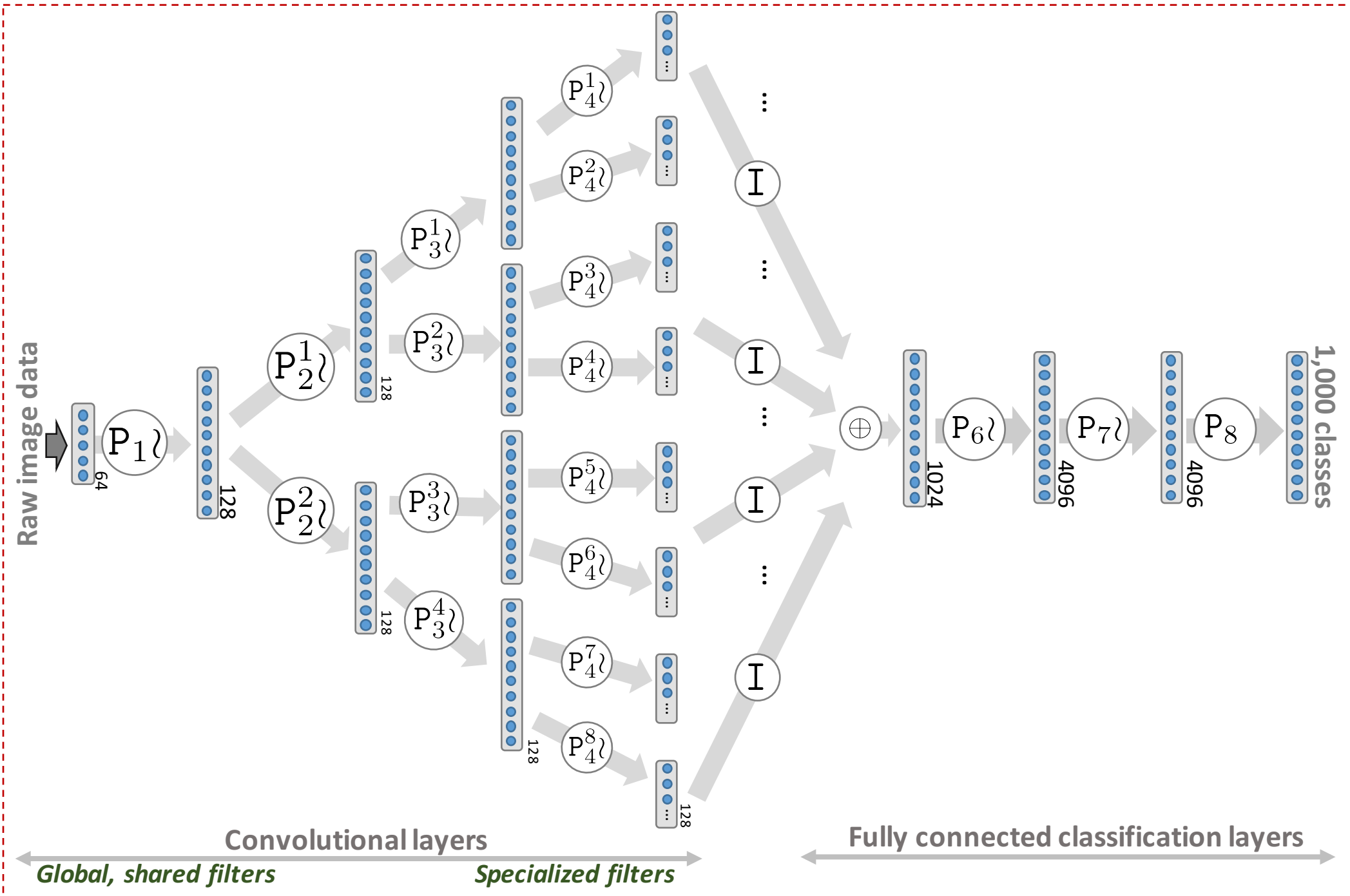
CIFAR



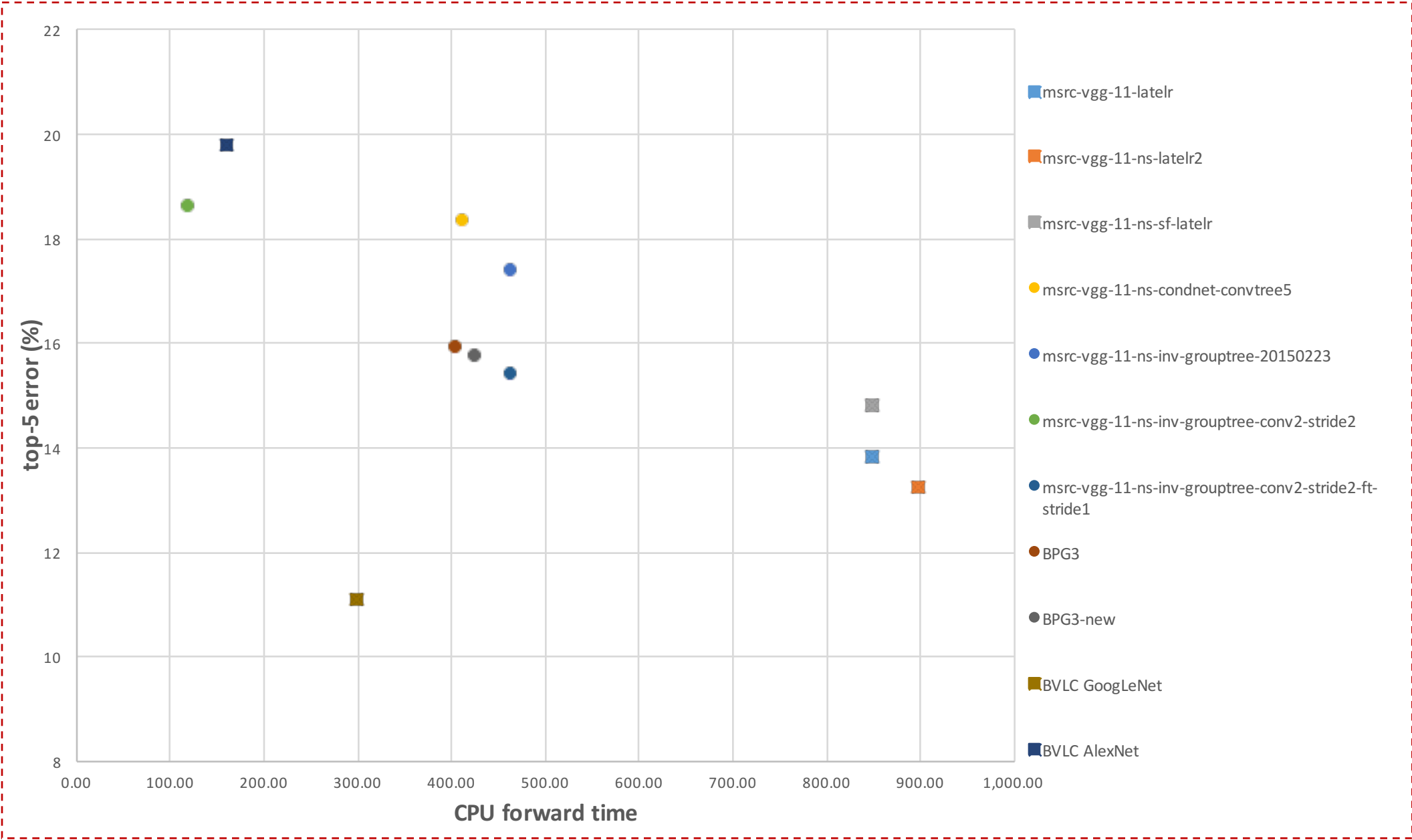
CIFAR



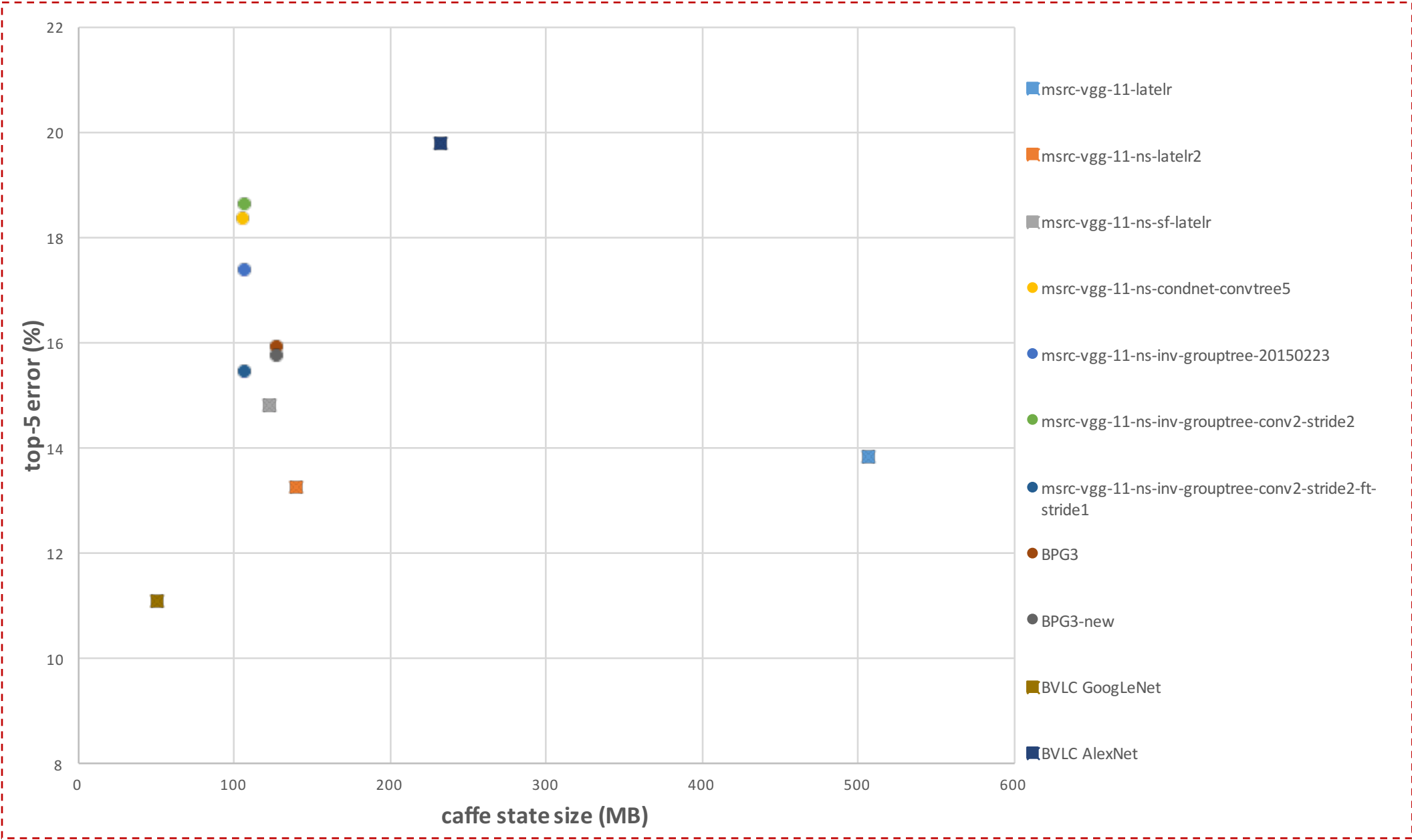
ImageNet

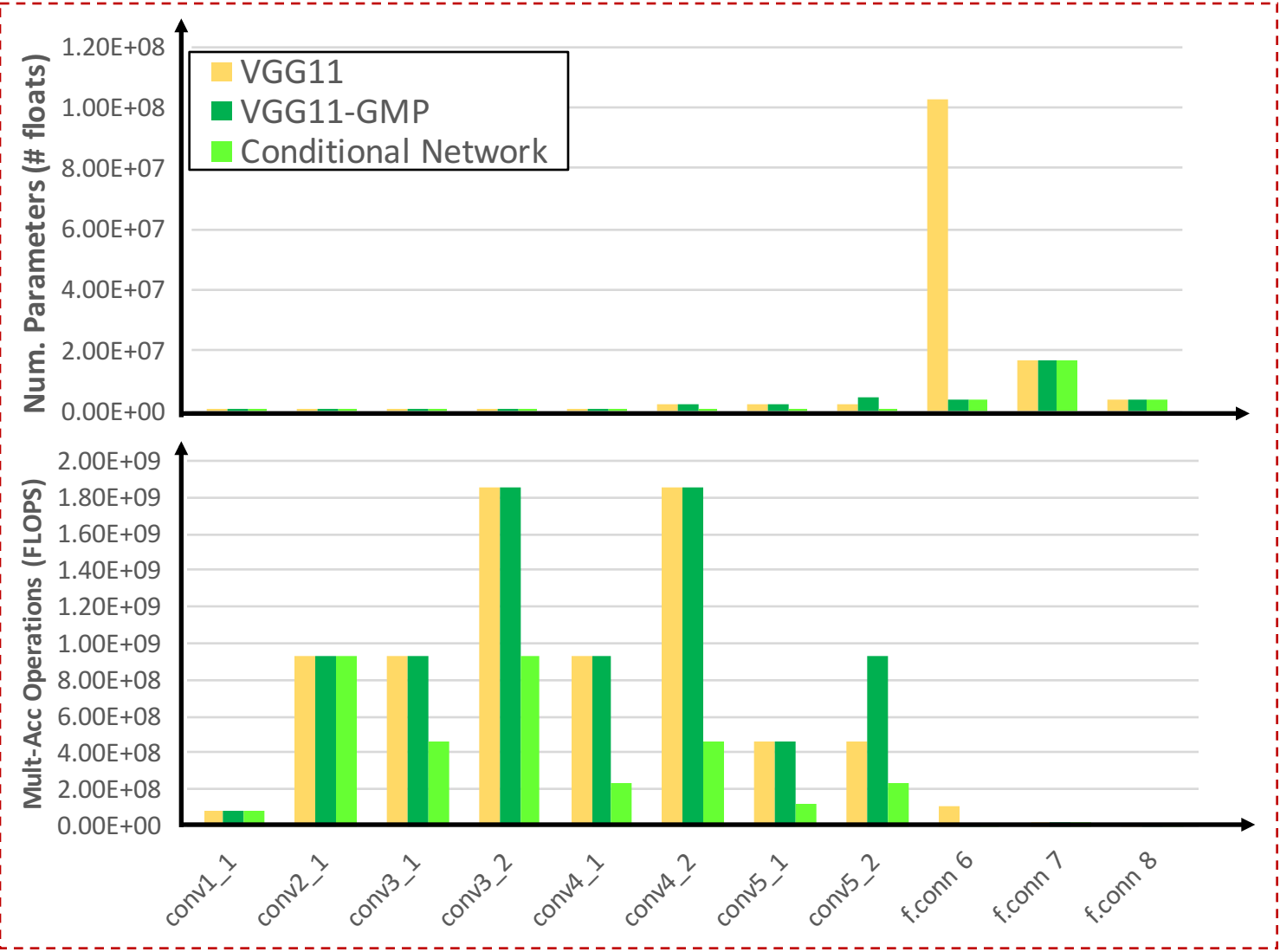


ImageNet

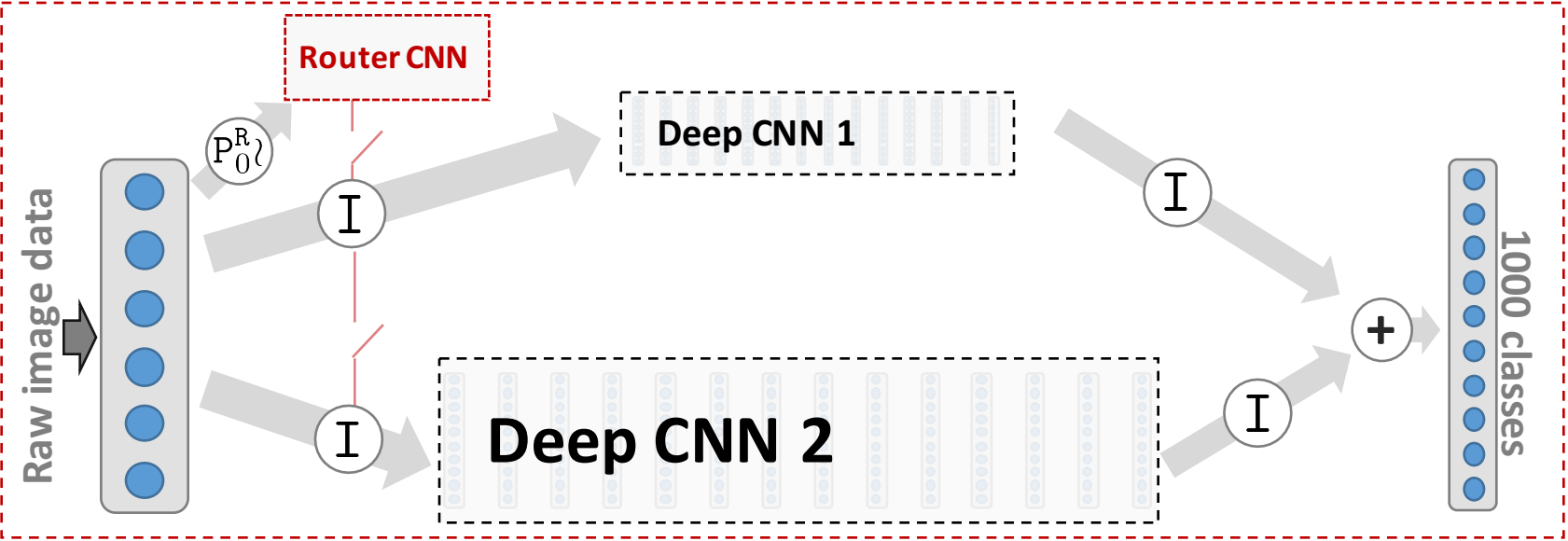


ImageNet

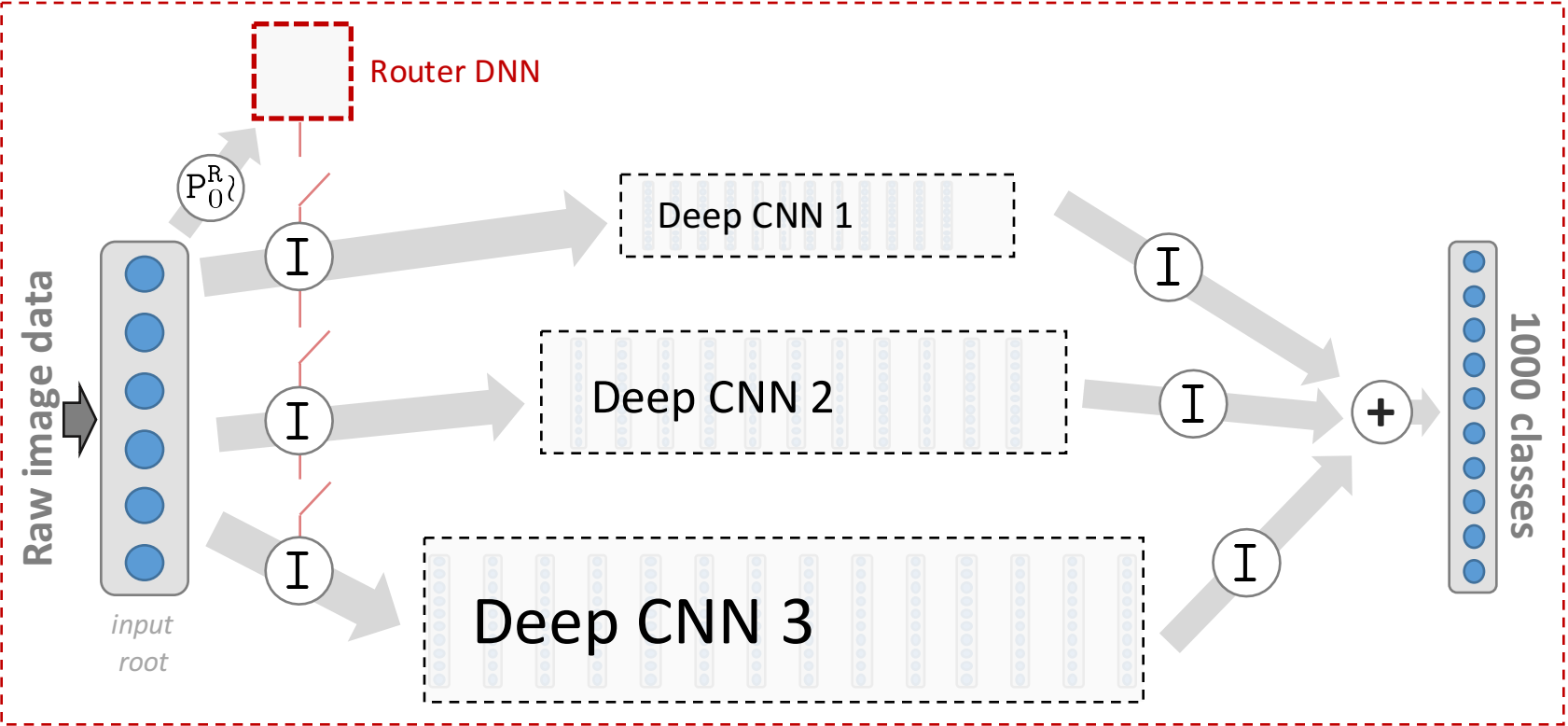




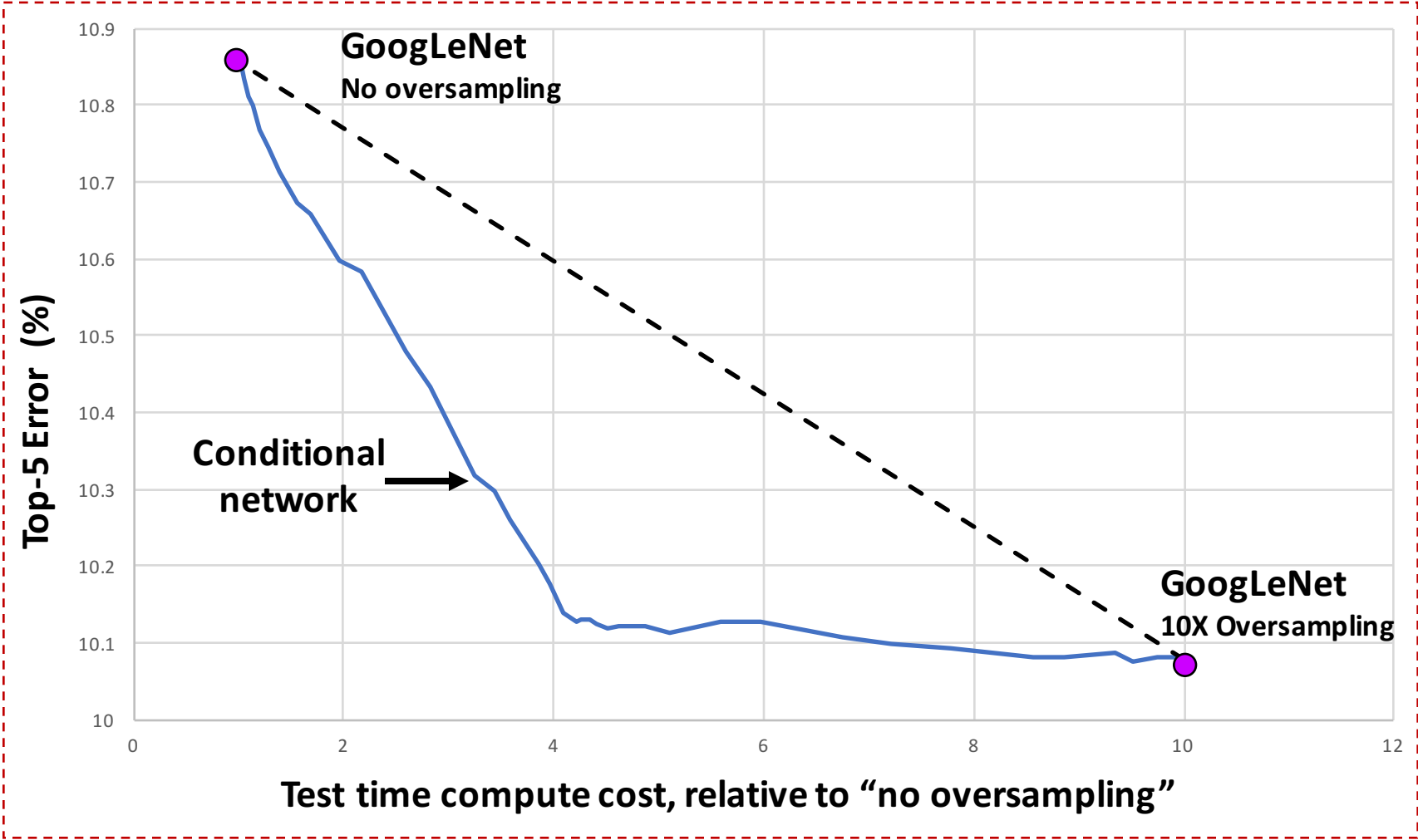
Mixing



Mixing



Mixing



OLD unused

