Knowledge Distillation Using Early Exit LLMs

Experiments on Confidence Scoring

Edvin 24V0074 Sagar 24D0367

CS 769
Optimization in Machine Learning

10 April 2025

Overview

1. Knowledge Distillation
2. KD with early exits
3. Fixing overconfidence in Dynamic Neural Networks
4. Early-Exit with Nested Prediction Sets

Section Overview

1. Knowledge Distillation
2. KD with early exits
3. Fixing overconfidence in Dynamic Neural Networks
4. Early-Exit with Nested Prediction Sets

Knowledge Distillation

- Introduced in [Hinton et al., 2015]

Large Cumbersome Models are difficult to deploy
need to train smaller models efficiently

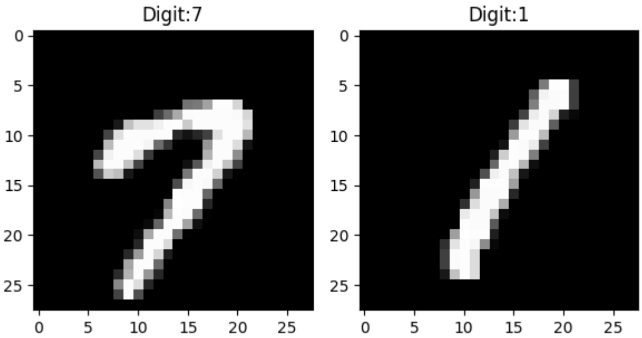
Knowledge Distillation

- Introduced in [Hinton et al., 2015]
- Student - teacher models

Smaller Student model tries to mimic larger teacher model that generalises well

Knowledge Distillation

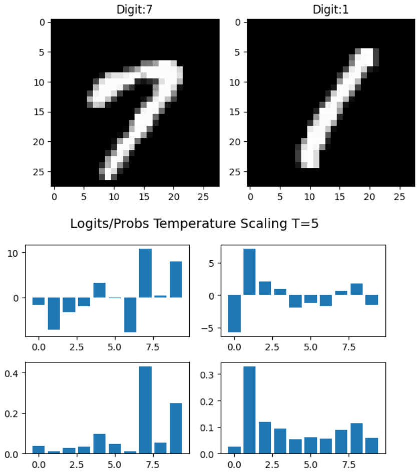
- Introduced in [Hinton et al., 2015]
- Student - teacher models
- Teacher Provides "Soft Targets"



$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_i/T)}$$

Knowledge Distillation

- Introduced in [Hinton et al., 2015]
- Student - teacher models
- Teacher Provides "Soft Targets"
- Loss: kl divergence + cross-entropy



Section Overview

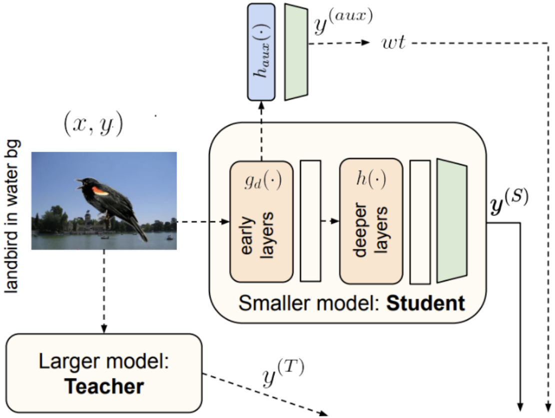
KD with early exits

- Proposed by
  [Tiwari et al., 2024]

KD with early exits

- Proposed by [Tiwari et al., 2024]
- Student model relies on spurious correlations

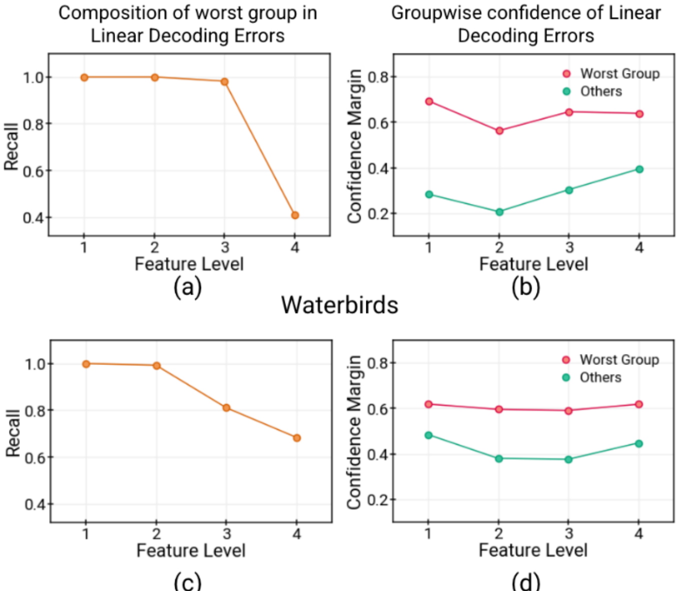
Smaller Models trained via KD rely more on spurious correlations than the teacher model, which leads to Poor performance on group fairness metrics by the student



$$\mathcal{L}_{student} = (1 - \lambda) \cdot l_{ce}(y^{(S)}, y) + \lambda \cdot \boldsymbol{wt} \cdot l_{kd}(y^{(S)}, y^{(T)})$$

KD with early exits

- Proposed by [Tiwari et al., 2024]
- Student model relies on spurious correlations
- Student's early Layers overconfident on hard instances



KD with early exits

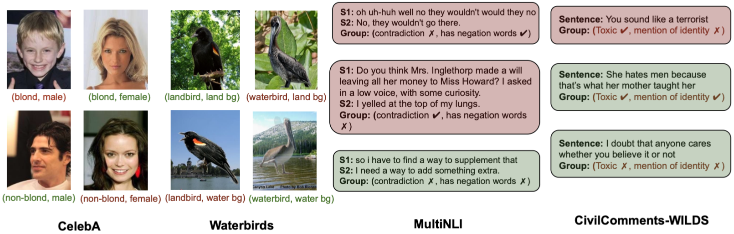
- Proposed by [Tiwari et al., 2024]
- Student model relies on spurious correlations
- Student's early Layers overconfident on hard instances
- DEDIER training

$$\mathcal{L}_{student} = \sum_{D_w} (1-\lambda) \cdot l_{ce} + \lambda \cdot \mathbf{wt} \cdot l_{kd}$$

where $\mathbf{wt} = \exp^{\beta . \mathbf{cm} . \alpha}$ and $\mathbf{cm}(\mathbf{p}) = \mathbf{p_{max}} - \max_{\mathbf{p_k} \in \mathbf{p} - \mathbf{p_{max}}} \mathbf{p_k}$

KD with early exits

- Proposed by [Tiwari et al., 2024]
- Student model relies on spurious correlations
- Student's early Layers overconfident on hard instances
- DEDIER training
- Experiments



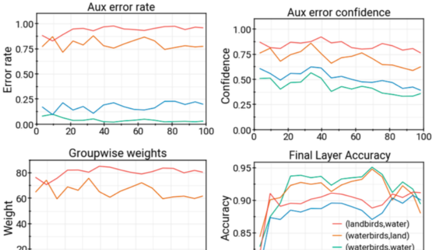
KD with early exits

- Proposed by [Tiwari et al., 2024]
- Student model relies on spurious correlations
- Student's early Layers overconfident on hard instances
- DEDIER training
- Experiments

Compared to some other methods

Waterbirds groups	Teacher	DeTT	SimKD	DEDIER
(waterbird, water bg)	94.3	92.6 ± 0.70	89.4 ± 0.06	94.1 ± 0.86
(landbird, land bg)	91.6	90.0 ± 0.06	92.1 ± 0.46	$\boxed{89.8}$ ± 0.46
(waterbird, land bg)	91.7	$\boxed{88.3}$ ± 0.81	$\boxed{71.4}$ ± 2.15	92.1 ± 0.40
(landbird, water bg)	$\boxed{91.4}$	88.8 ± 1.70	84.6 ± 0.79	90.6 ± 0.67
CelebA groups				
(blond, female)	94.3	92.6 ± 1.14	92.2 ± 0.46	92.7 ± 1.48
(non-blond, male)	92.9	92.3 ± 0.58	93.0 ± 0.46	93.2 ± 0.42
(non-blond, female)	92.1	93.0 ± 0.78	93.2 ± 0.35	93.1 ± 0.52
(blond, male)	$\boxed{90.0}$	$\boxed{89.5}$ ± 0.71	$\boxed{89.0}$ ± 0.35	$\boxed{89.6}$ ± 1.96

Adaptive to the dataset



KD with Early Exists

Algorithm 1 The DEDIER approach: learning student $\mathcal{S}$, given dataset $\mathcal{D} = (x_i, y_i) \mid i \in (1 \cdots N)$ and teacher $\mathcal{T}$.

Hyperparameters: Distillation loss fraction λ, depth d of early readout, parameters α and β for calculating the weights.

$\mathcal{S} = h(g_d(\cdot))$: break student down into early layers g_d of depth d and remaining deeper layers h.
Let $h_{aux}(\cdot)$ be the auxiliary network
We augment dataset D with weights as $\mathcal{D}_w \leftarrow (x_i, y_i, wt_i) \mid i \in (1 \cdots n)$, where $(wt_1 \cdots wt_N) \leftarrow (1 \cdots 1)_n$

for each $e \in \{1 \cdots E\}$ **do**
    Train student model $\mathcal{S}$ using the loss described in Eq. 5.
    if $e\%L == 0$ **then**:
        Train $h_{aux}(g_d(\cdot))$ for R epochs on dataset D using standard cross-entropy.
        for each $(x, y, wt) \in \mathcal{D}_w$ **do**
            $y^{(\text{aux})} = h_{aux}(g_d(x))$
            Update wt according to Eq. 4.
        end for
    end if
end for

Section Overview

Fixing overconfidence in Dynamic Neural Networks

[Meronen et al., 2023]

- To decrease computational cost, we do not want to run network for more layers that required for the specific task
- To be able to know where to stop, the model needs good uncertainty estimates
- The paper aims to improve uncertainty estimates for a model

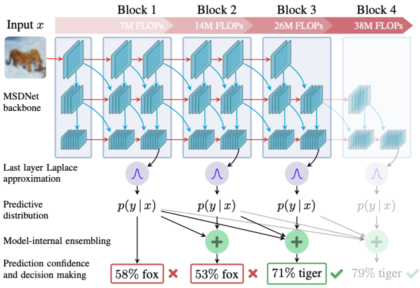


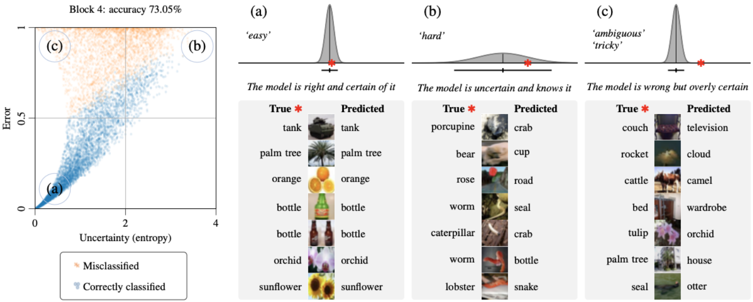
Figure: Increasing depth of dynamic neural network

Background

- Investigates image classification under budget restrictions $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{n_{\text{train}}}$
- Budget B (FLOPs) must be distributed across a batch for highest possible accuracy
- With n_{block} intermediate classifiers, the predictive distribution: $p_k(\hat{\mathbf{y}}_i \mid \mathbf{x}_i),\ k = 1, 2, \ldots, n_{\text{block}}$
- Feature representation on the last linear layer $\phi_{i,k} = f_k(\mathbf{x}_i)$ with parameters $\theta_k = \{\mathbf{W}_k, \mathbf{b}_k\}$
- Prediction of layer k: $p_k(\hat{\mathbf{y}}_i \mid \mathbf{x}_i) = \text{softmax}(\hat{\mathbf{z}}_{i,k}),\ \text{where } \hat{\mathbf{z}}_{i,k} = \mathbf{W}_k \phi_{i,k} + \mathbf{b}_k$

Aleatoric and epistemic uncertainty

- Aleatoric uncertainty is related to randomness intrinsic to the task at hand and cannot be reduced.
- Epistemic uncertainty is related to our knowledge of the task and can be reduced by learning more about the task $\rightarrow$ more data



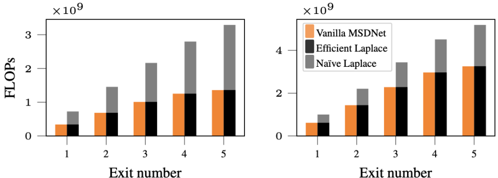
Bayesian treatment of parameters

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}}) = \frac{p(\mathcal{D}_{\text{train}} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(\mathcal{D}_{\text{train}}, \boldsymbol{\theta})\, d\boldsymbol{\theta}} = \frac{[\text{likelihood}] \times [\text{prior}]}{[\text{model evidence}]}$$

- Posterior distribution over the model parameters is intractable in deep learning
- Laplace approximation (second order Taylor expansion)
- MAP estimate can be found by maximising the unnormalised posterior:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}_{\text{train}} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$$

$$p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}}) \approx \mathcal{N}(\hat{\boldsymbol{\theta}}, \mathbf{H}^{-1})$$



Method

- Last layer Laplace approximation for each intermediate classifier of the DNN
- Final prediction:

$$\hat{\mathbf{y}}_i = \frac{1}{n_{\mathrm{MC}}} \sum_{l=1}^{n_{\mathrm{MC}}} \mathrm{softmax}(\hat{\mathbf{z}}_i^{(l)})$$

- Laplace implementation has cost:

$$\mathrm{FLOPs}_{\mathrm{efficient}} = 2cn_{\mathrm{MC}} + 2p^2 + 5p + 2$$

  (c: number of classes, p: feature dimensionality, n: number of MC samples)

- Gaussian distribution:

$$p(\hat{\mathbf{z}}_i \mid \mathbf{x}_i) = \mathcal{N}(\hat{\mathbf{W}}_{\mathrm{MAP}}^{\top} \hat{\phi}_i, (\hat{\phi}_i^{\top} \mathbf{V} \hat{\phi}_i)\mathbf{U})$$

$$\mathbf{V}^{-1} \otimes \mathbf{U}^{-1} = \mathbf{H}^{-1}$$

- Samples:

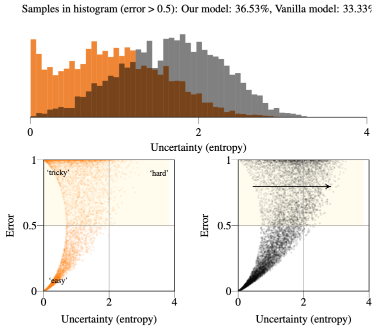
$$\hat{\mathbf{z}}_i^{(l)} = \hat{\mathbf{W}}_{\mathrm{MAP}}^{\top} \hat{\phi}_i + (\hat{\phi}_i^{\top} \mathbf{V} \hat{\phi}_i)^{\frac{1}{2}} (\mathbf{L}\mathbf{g}^{(l)})$$

  $\mathbf{g}^{(l)} \sim \mathcal{N}(0, \mathbf{I})$ and $\mathbf{L}$ is the Cholesky factor of $\mathbf{U}$

- Temperature scaling is recommended for well-calibrated predictions

Uncertainty

- The uncertainty should be high for the model to be able to recognize these samples as ‘tricky’, and continue their evaluation to the next block.
- For paper model these samples have a high uncertainty, while the vanilla MSDNet is overconfident.



(a) Vanilla MSDNet (ACC: 69.08%) (b) Our model (ACC: 70.90%)

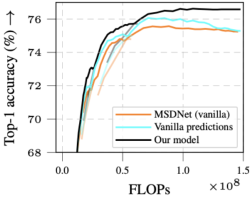
Ensamble prediction

$$p_k^{\text{ens}}(\hat{\mathbf{y}}_i \mid \mathbf{x}_i) = \frac{1}{\sum_{l=1}^{k} w_l} \sum_{m=1}^{k} w_m \, p_m(\hat{\mathbf{y}}_i \mid \mathbf{x}_i)$$

Weights w are the computational costs in FLOPs up to classifier m

- Early exiting decisions based on model predicted confidence (referred to as MIE)
- Thresholds for exiting are calculated on the validation set. Different for every layer (not included in paper)

Results



	$(n_{\text{train}}, d, c, n_{\text{batch}})$	CIFAR-100 (50000, 3072, 100, 64)			
		Top-1 ACC ↑	Top-5 ACC ↑	NLPD ↓	ECE ↓
Small	MSDNet (vanilla)	69.25	90.48	1.498	0.182
	+ Laplace T_{opt} σ_{opt}	69.06 −0.19	90.58 +0.10	1.208 −0.289	0.073 −0.109
	+ MIE	**69.97** +0.72	90.88 +0.40	1.218 −0.280	0.080 −0.102
	+ MIE Laplace T_{opt} σ_{opt}	69.84 +0.59	**91.09** +0.61	**1.133** −0.364	**0.017** −0.165
Medium	MSDNet (vanilla)	74.12	91.94	1.549	0.190
	+ Laplace T_{opt} σ_{opt}	73.92 −0.20	92.01 +0.06	1.070 −0.479	0.083 −0.107
	+ MIE	**75.03** +0.91	92.97 +1.03	1.011 −0.538	0.050 −0.140
	+ MIE Laplace T_{opt} σ_{opt}	74.99 +0.86	**93.23** +1.29	**0.944** −0.605	**0.026** −0.164
Large	MSDNet (vanilla)	75.36	92.78	1.475	0.178
	+ Laplace T_{opt} σ_{opt}	75.32 −0.05	92.83 +0.05	0.996 −0.479	0.075 −0.103
	+ MIE	76.32 +0.95	93.50 +0.72	0.949 −0.525	0.061 −0.117
	+ MIE Laplace T_{opt} σ_{opt}	**76.34** +0.98	**93.84** +1.05	**0.885** −0.590	**0.025** −0.152

Figure: Enter Caption

Section Overview

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024]
  AVCS

[Jazbec et al., 2024] propose using **A**nytime**V**alid**C**onfidence **S**equences(AVCS) for uncertainity estimation in Early Exit Networks

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024]
  AVCS
- AVCS has time uniform and non - asymptotic guarantees

$$\mathcal{C}_t = (l_t, r_t) \subseteq \mathbb{R}$$

$$\mathcal{L}(W_{1:T}, U_{1:T}; \mathcal{D}) = \sum_{n=1}^{N} \frac{1}{T} \sum_{t=1}^{T} \ell(y_n, f(x_n; W_t, U_{1:t}))$$

$$\text{size}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} |\mathcal{C}_t(x_n)|$$

$$\text{coverage}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} [y_n \in \mathcal{C}_t(x_n)]$$

$$\mathcal{N}(t) = |\bigcap_{s \leq t} \mathcal{C}_s| / |\mathcal{C}_t|$$

$$\mathbb{P}(\forall t,\, \theta^* \in \mathcal{C}_t) \geq 1 - \alpha$$

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024]
  AVCS
- AVCS has time uniform and non - asymptotic guarantees
- Martingales And Ville's theorem

$$\mathbb{E}_{x_{t+1}}\left[R_{t+1}(\theta^*) \mid x_1, \ldots, x_t\right] = R_t(\theta^*)$$

$$\mathbb{P}(\exists t : R_t(\theta^*) \geq 1/\alpha) \leq \alpha$$

$$\mathcal{C}_t := \{\theta : R_t(\theta) \leq 1/\alpha\}$$

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024] **AVCS**
- AVCS has time uniform and non - asymptotic guarantees
- Martingales And Ville's theorem
- Predictive-likelihood ratio

Idea is to create a sequence of martingales for each exit layer and then use Ville's theorem to construct AVCS.

$$R_t^*(y) = \prod_{l=1}^{t} \frac{p_l(y|x^*, \mathcal{D})}{p(y|x^*, W_l)}, W_l \sim p(W_l|D^*)$$

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024] **AVCS**
- AVCS has time uniform and non - asymptotic guarantees
- Martingales And Ville's theorem
- Predictive-likelihood ratio
- AVCS for Regression

$$y \sim \mathcal{N}\left(y; h_t(x)^T W_t, \sigma_t^2\right),$$
$$W_t \sim \mathcal{N}(W_t; \hat{W}_t, \sigma_{w,t}^2 \mathbb{I}_H)$$
$$p(W_t|\mathcal{D}) = \mathcal{N}(W_t; \mu_t, \Sigma_t),$$
$$p_t(y|x^*, \mathcal{D}) = \mathcal{N}(y; h_t(x^*)^T \mu_t, v_* + \sigma_t^2)$$

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024]
  AVCS
- AVCS has time uniform and non - asymptotic guarantees
- Martingales And Ville's theorem
- Predictive-likelihood ratio
- AVCS for Regression
- AVCS for Classification

$$p(y|\pi_t) = \mathrm{Cat}(y|\pi_t),$$

$$p(\pi_t|x^*, \mathcal{D}) = \mathrm{Dir}(\pi_t|\alpha_t(x^*; \mathcal{D}))$$

$$p_t(y = y|x^*, \mathcal{D}) = \int p(y = y|\pi_t)\, p(\pi_t|x^*, \mathcal{D})\, d\pi_t$$

$$= \frac{\alpha_{t,y}}{\sum_{y' \in \mathcal{Y}} \alpha_{t,y'}}$$

Early-Exit with Nested Prediction Sets

- [Jazbec et al., 2024] **AVCS**
- AVCS has time uniform and non - asymptotic guarantees
- Martingales And Ville's theorem
- Predictive-likelihood ratio
- AVCS for Regression
- AVCS for Classification
- Post-Hoc Implementation

$$a_t(x) = \mathrm{ReLU}(x, \tau_t)$$

Evaluations

$$\mathcal{C}_t = (l_t, r_t) \subseteq \mathbb{R}$$

$$\mathcal{L}(W_{1:T}, U_{1:T}; \mathcal{D}) =$$

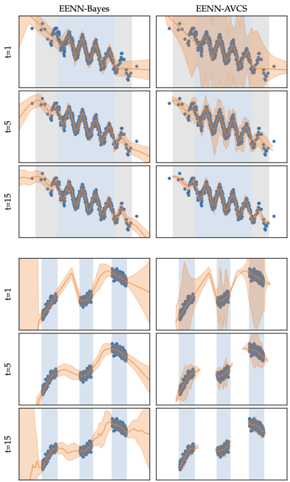
$$\sum_{n=1}^{N} \frac{1}{T} \sum_{t=1}^{T} \ell(y_n, f(x_n; W_t, U_{1:t}))$$

$$\text{size}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} |\mathcal{C}_t(x_n)|$$

$$\text{coverage}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} [y_n \in \mathcal{C}_t(x_n)]$$

$$\mathcal{N}(t) = |\bigcap_{s \leq t} \mathcal{C}_s| / |\mathcal{C}_t|$$

$$\mathbb{P}(\forall t,\ \theta^* \in \mathcal{C}_t) \geq 1 - \alpha$$



Evaluations

$$\mathcal{C}_t = (l_t, r_t) \subseteq \mathbb{R}$$

$$\mathcal{L}(W_{1:T}, U_{1:T}; \mathcal{D}) =$$

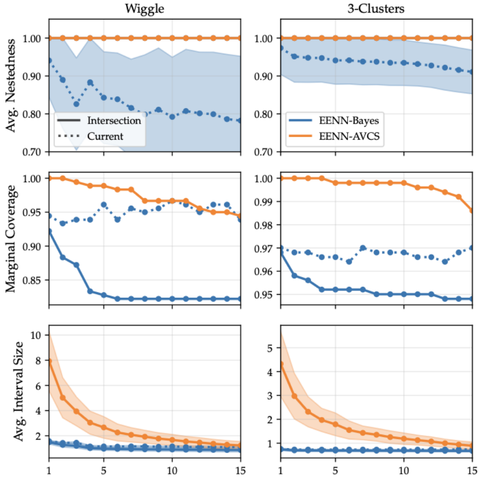
$$\sum_{n=1}^{N} \frac{1}{T} \sum_{t=1}^{T} \ell(y_n, f(x_n; W_t, U_{1:t}))$$

$$\text{size}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} |\mathcal{C}_t(x_n)|$$

$$\text{coverage}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} [y_n \in \mathcal{C}_t(x_n)]$$

$$\mathcal{N}(t) = |\bigcap_{s \leq t} \mathcal{C}_s| / |\mathcal{C}_t|$$

$$\mathbb{P}(\forall t,\ \theta^* \in \mathcal{C}_t) \geq 1 - \alpha$$



Evaluations

$$\mathcal{C}_t = (l_t, r_t) \subseteq \mathbb{R}$$

$$\mathcal{L}(W_{1:T}, U_{1:T}; \mathcal{D}) =$$

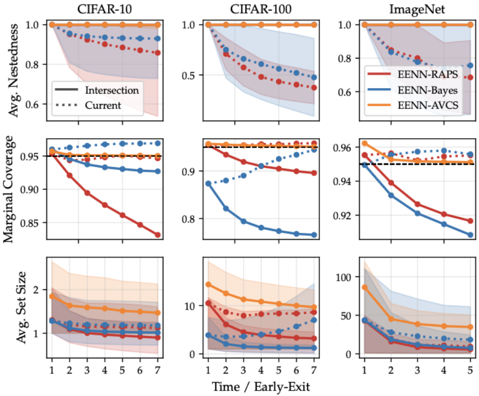
$$\sum_{n=1}^{N} \frac{1}{T} \sum_{t=1}^{T} \ell(y_n, f(x_n; W_t, U_{1:t}))$$

$$\text{size}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} |\mathcal{C}_t(x_n)|$$

$$\text{coverage}(t) := \frac{1}{n_{\text{test}}} \sum_{n=1}^{n_{\text{test}}} [y_n \in \mathcal{C}_t(x_n)]$$

$$\mathcal{N}(t) = |\bigcap_{s \leq t} \mathcal{C}_s| / |\mathcal{C}_t|$$

$$\mathbb{P}(\forall t,\ \theta^* \in \mathcal{C}_t) \geq 1 - \alpha$$



Proposal

- Improving confidence margins in student model makes for more efficient knowledge transfer with DEDIER approach.
  - AVCS is one proposed approach
  - Using last layer Laplace approximation is another proposed approach
- Test the approaches on newer LLMs like Llama 3.2

References

Hinton, G., Vinyals, O., and Dean, J. (2015).
Distilling the Knowledge in a Neural Network.
arXiv:1503.02531 [cs, stat].

Jazbec, M., Forré, P., Mandt, S., Zhang, D., and Nalisnick, E. (2024).
Early-Exit Neural Networks with Nested Prediction Sets.
arXiv:2311.05931 [cs, stat].

Meronen, L., Trapp, M., Pilzer, A., Yang, L., and Solin, A. (2023).
Fixing Overconfidence in Dynamic Neural Networks.
Version Number: 4.

Tiwari, R., Sivasubramanian, D., Mekala, A., Ramakrishnan, G., and Shenoy, P. (2024).
Using Early Readouts to Mediate Featural Bias in Distillation.
In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2626–2635, Waikoloa, HI, USA. IEEE.

Thank You