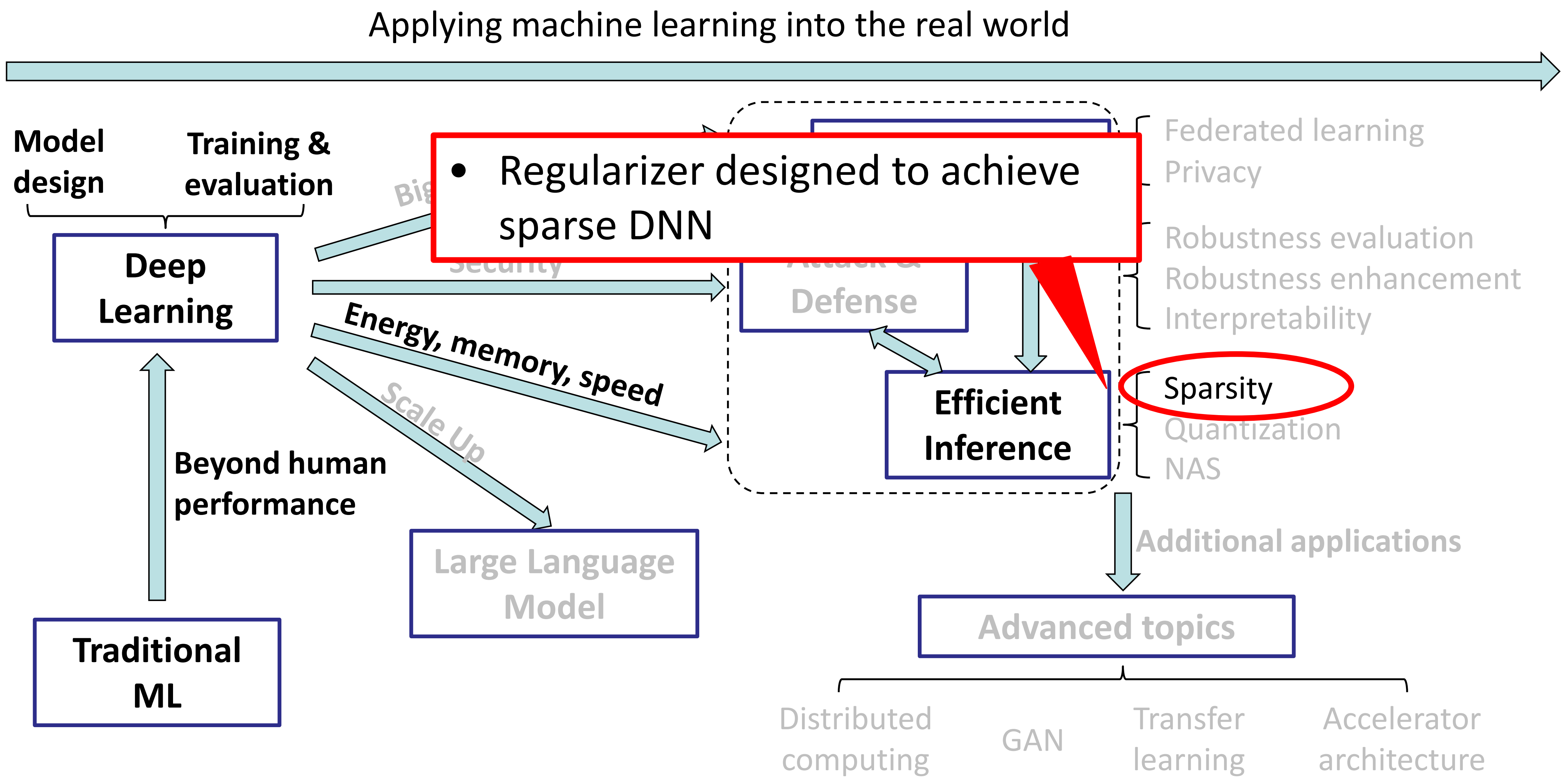
ECE 661 COMP ENG ML & DEEP NEURAL NETS

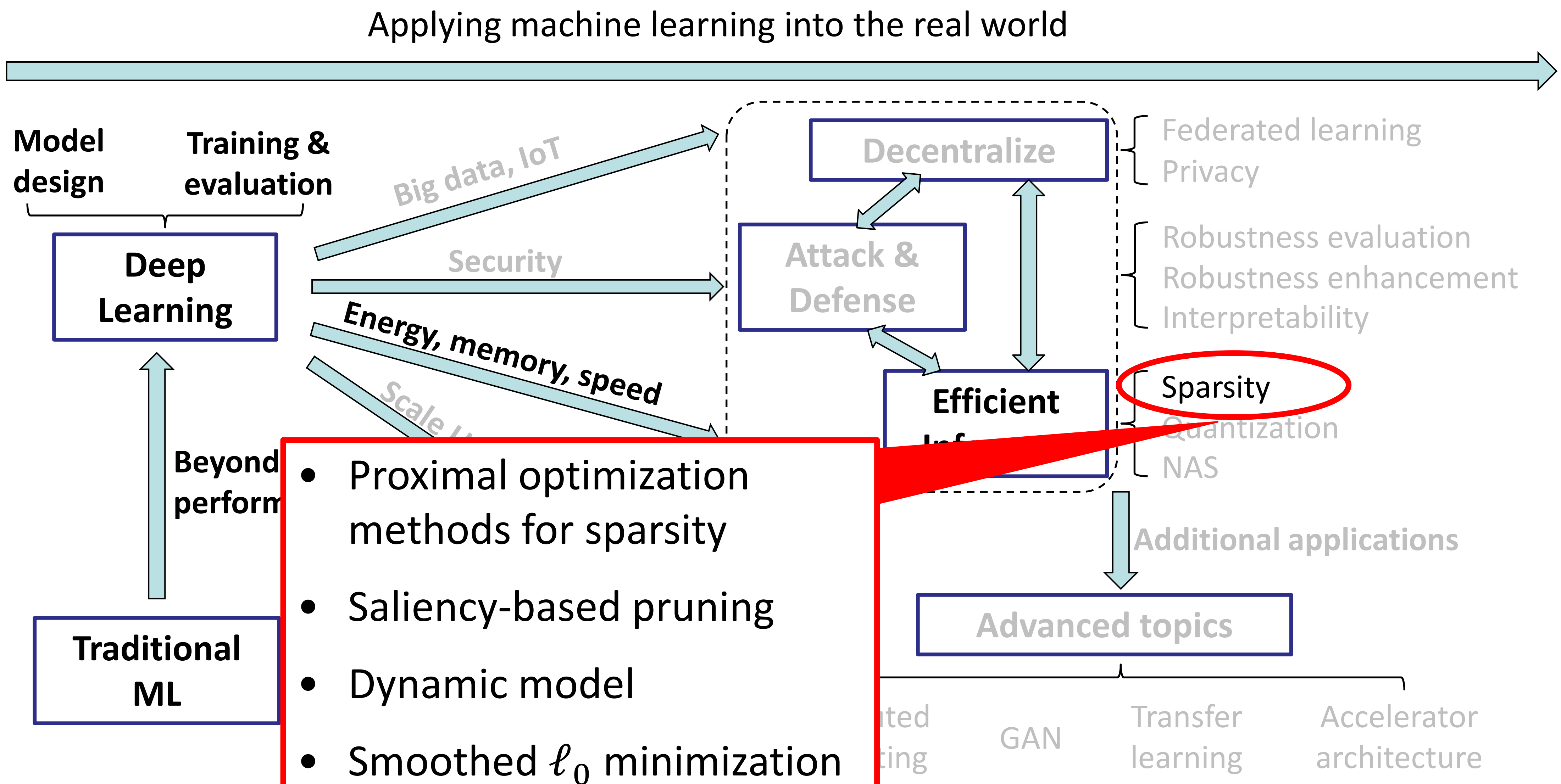
16. OPTIMIZATION-BASED METHODS FOR SPARSE DNN

HAI "HELEN" LI, SPRING 2025

Previously


Applying machine learning into the real world
Model design
Training & evaluation
Deep Learning
Traditional ML
Beyond human performance
Energy, memory, speed
Scale Up
Large Language Model
Security
Defense
Efficient Inference
Regularizer designed to achieve sparse DNN
Federated learning
Privacy
Robustness evaluation
Robustness enhancement
Interpretability
Sparsity
Quantization
NAS
Additional applications
Advanced topics
Distributed computing
GAN
Transfer learning
Accelerator architecture

This lecture



- Proximal optimization methods for sparsity
- Saliency-based pruning
- Dynamic model
- Smoothed ℓ_0 minimization

Recall: Regularized training

- In a typical objective of regularized training of DNN, regularizer $R(\cdot)$ is applied to each parameter θ_i, then added to the overall objective with strength λ

$$min_{\theta}\mathcal{L}(\theta) + \lambda \sum_{i} R(\theta_i)$$

- If the regularizer is differentiable, we can directly use SGD on the overall objective, but the performance will be sensitive to the choice of λ
  - We tend to choose a larger λ if we want the regularizer to have a stronger effect
  - But large strength will make the gradient of R dominate the overall gradient, affecting the minimization of $\mathcal{L}(\theta)$

Smoothing the regularized optimization

$$min_\theta \mathcal{L}(\theta) + \lambda \sum_i R(\theta_i)$$

- Proximal gradient update
  - Consider $min_\theta \mathcal{L}(\theta)$ as main objective, modify the resulted $\theta^{(l)}$ at each step slightly to minimize the regularizer term

Proximal gradient update

- Consider a general case $min_\theta \mathcal{L}(\theta) + \lambda R(\theta)$, at step l
- Main objective: Optimize $\mathcal{L}(\theta)$

$$\hat{\theta}^{(l)} = \theta^{(l-1)} - \alpha \nabla_\theta \mathcal{L}\left(\theta^{(l-1)}\right)$$

- Proximal operator: Optimize $R(\theta)$
  - Try minimize $R(\cdot)$ in the proximity of $\hat{\theta}^{(l)}$

$$\theta^{(l)} \coloneqq prox_{\lambda R}\left(\hat{\theta}^{(l)}\right) = argmin_z \left(\lambda R(z) + \frac{1}{2}\left\|z - \hat{\theta}^{(l)}\right\|_2^2\right)$$

- The added proximity term will allow **smoother convergence** of the overall objective

You may refer to https://web.stanford.edu/~boyd/papers/pdf/prox_algs.pdf or ECE 741 Compressed Sensing for more background, analysis and application of the proximal optimization methods.

Special case: ℓ_0 as indicator function

$$\hat{\theta} \coloneqq argmin_\theta \mathcal{L}(\theta)\ s.t. \left|\left|\theta\right|\right|_0 \leq n$$

- Convert the constraint into an indicator function
  - Let $g(\theta) = \begin{cases} 0, \left|\left|\theta\right|\right|_0 \leq n \\ +\infty, \ \ Otherwise \end{cases}$, we can convert the optimization objective into $\hat{\theta} \coloneqq argmin_\theta \mathcal{L}(\theta) + g(\theta)$
- Main objective: Gradient descent to optimize $\mathcal{L}(\theta)$

$$\hat{\theta}^{(l)} = \theta^{(l-1)} - \alpha \nabla_\theta \mathcal{L}\left(\theta^{(l-1)}\right)$$

- Proximal operator: Optimize $g(\theta)$

$$\theta^{(l)} = argmin_z \left(g(z) + \frac{1}{2}\left\|z - \hat{\theta}^{(l)}\right\|_2^2\right)$$

$$= argmin_z \left\|z - \hat{\theta}^{(l)}\right\|_2 \ s.t. \left\|z\right\|_0 \leq n$$

Projection step for iterative pruning

PGD is a special case of proximal update with indicator function

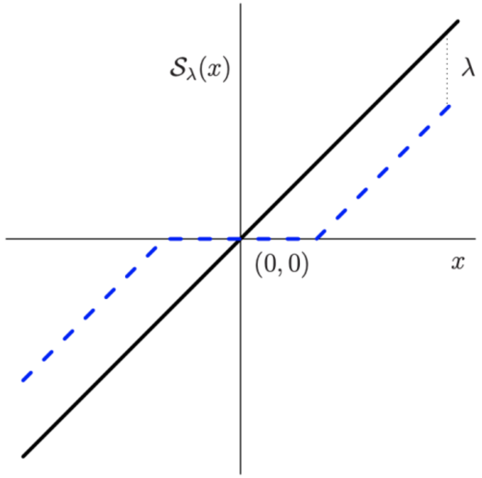
Special case: Revisit ℓ_1 regularization

$$min_\theta \mathcal{L}(\theta) + \lambda \sum_i |\theta_i|$$

- Previously we considered direct SGD optimization of the regularized objective
- We can use the proximal update to better understand how ℓ_1 works
- Proximal operator of ℓ_1: (known as soft thresholding)

$$prox_{\lambda\ell_1}(\theta_i) = argmin_z \left(\lambda|z| + \frac{1}{2}\|z - \theta_i\|_2^2 \right)$$

$$= \begin{cases} \theta_i - \lambda, & \theta_i > \lambda \\ 0, & |\theta_i| \leq \lambda \\ \theta_i + \lambda, & \theta_i < -\lambda \end{cases}$$



Tibshirani, Robert, Martin Wainwright, and Trevor Hastie. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

Recall: Pros & Cons of ℓ_1

$$prox_{\lambda\ell_1}(\theta_i) = \begin{cases} \theta_i - \lambda, & \theta_i > \lambda \\ 0, & |\theta_i| \leq \lambda \\ \theta_i + \lambda, & \theta_i < -\lambda \end{cases}$$

- Pros:
  - Simple, effectively leading to sparsity
  - Pruning effect controlled by λ
- Cons:
  - **Biased**: the absolute value of large element always shrinks by λ, leading to loss of variance
- Improvement: change how large elements are handled

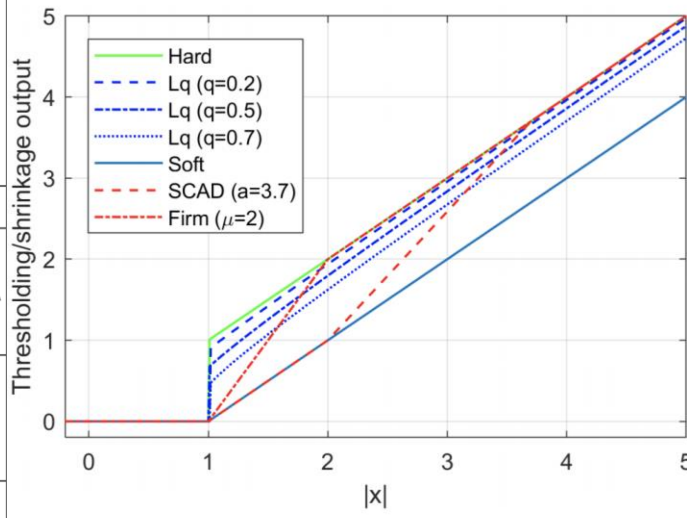
Tibshirani, Robert, Martin Wainwright, and Trevor Hastie. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

Improvements from ℓ_1

- There are a lot of ways to (partially) solve the “bias” problem of ℓ_1, here we show some famous choices used in compressed sensing research

TABLE I: Regularization penalties and the corresponding proximity operator ($\lambda > 0$ is a thresholding parameter).

Penalty name	Penalty formulation	Proximity operator
(iii) ℓ_q-norm [6], [7]	$P_\lambda(x) = \lambda\|x\|^q,\ 0 < q < 1$	$\mathrm{prox}_{P_\lambda}(t) = \begin{cases} 0, & \|t\| < \tau \\ \{0, \mathrm{sign}(t)\beta\}, & \|t\| = \tau \\ \mathrm{sign}(t)y, & \|t\| > \tau \end{cases}$ where $\beta = [2\lambda(1-q)]^{1/(2-q)}$, $\tau = \beta + \lambda q \beta^{q-1}$, $h(y) = \lambda q y^{q-1} + y - \|t\| = 0$ and $y \in [\beta, \|t\|]$
(iv) q-shrinkage [10]	N/A ($q < 1$)	$\mathrm{prox}_{P_\lambda}(t) = \mathrm{sign}(t)\max\left\{\|t\| - \lambda^{2-q}t^{q-1}, 0\right\}$
(v) SCAD [11]	$P_\lambda(x) = \begin{cases} \lambda\|x\|, & \|x\| < \lambda \\ \frac{2a\lambda\|x\| - x^2 - \lambda^2}{2(a-1)}, & \lambda \le \|x\| < a\lambda \\ (a+1)\lambda^2/2, & \|x\| \ge a\lambda \end{cases}$	$\mathrm{prox}_{P_\lambda}(t) = \begin{cases} \mathrm{sign}(t)\max\{\|t\| - \lambda, 0\}, & \|t\| \le 2\lambda \\ \frac{(a-1)t - \mathrm{sign}(t)a\lambda}{a-2}, & 2\lambda < \|t\| \le a\lambda \\ \boxed{t,} & \|t\| > a\lambda \end{cases}$
(vi) MCP [12]	$P_{\lambda,\gamma}(x) = \lambda \int_0^{\|x\|} \max(1 - t/(\gamma\lambda), 0)dt$, where $\gamma > 0$	$\mathrm{prox}_{P_{\lambda,\gamma}}(t) = \begin{cases} 0, & \|t\| \le \lambda \\ \frac{\mathrm{sign}(t)(\|t\| - \lambda)}{1 - 1/\gamma}, & \lambda < \|t\| \le \gamma\lambda \\ \boxed{t,} & \|t\| > \gamma\lambda \end{cases}$
(vii) Firm thresholding [13]	$P_{\lambda,\mu}(x) = \begin{cases} \lambda[\|x\| - x^2/(2\mu)], & \|t\| \le \mu \\ \lambda\mu/2, & \|t\| \ge \mu \end{cases}$ where $\mu > \lambda$	$\mathrm{prox}_{P_{\lambda,\mu}}(t) = \begin{cases} 0, & \|t\| \le \lambda \\ \frac{\mathrm{sign}(t)(\|t\| - \lambda)\mu}{\mu - \lambda}, & \lambda \le \|t\| \le \mu \\ \boxed{t,} & \|t\| \ge \mu \end{cases}$



Wen, Fei, et al. "A survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning." *IEEE Access* 6 (2018): 69883-69906.

Further modification: Trimmed ℓ_1

- All previous variants of the ℓ_1 norm, including ℓ_1 itself, require a fixed strength/threshold as the hyperparameter, where a larger threshold leads to higher sparsity
- Recall iterative pruning, the pruning threshold is determined by the sparsity level we want to achieve
- We can combine the ideas by adding ℓ_1 penalty to a certain percentage of the smallest weights, while leaving larger weights unpenalized
- This is named "**Trimmed $\boldsymbol{\ell_1}$**", which allows sparsity exploration with a fixed upper bound

Formulation of the Trimmed ℓ_1

$$\underset{\boldsymbol{\theta}\in\Omega,\, \boldsymbol{w}\in[0,1]^p}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) + \lambda_n \sum_{j=1}^{p} w_j|\theta_j|$$

$$\text{s.t.} \ \ \mathbf{1}^\top \boldsymbol{w} \geq p - h\,. \qquad (1)$$

- p: dimension of parameter θ
- h: the number of nonzero parameters we want to keep
- “Gate variable” w: learnable ℓ_1 strength on each element (with constraint)
- The combination of the minimization objective and the constraint on the w enforces that the ℓ_1 penalty will be added on the smallest $p - h$ parameters at convergence

Yun, Jihun, et al. "Trimming the $\ell_1 $ Regularizer: Statistical Analysis, Optimization, and Applications to Deep Learning." *International Conference on Machine Learning*. 2019.

Optimization of Trimmed ℓ_1

$$\underset{\boldsymbol{\theta}\in\Omega,\, \boldsymbol{w}\in[0,1]^p}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) + \lambda_n \sum_{j=1}^{p} w_j|\theta_j|$$

$$\text{s.t.} \quad \mathbf{1}^\top \boldsymbol{w} \geq p - h. \qquad (1)$$

Objective of w

- Constrained optimization -> PGD

- PGD step for w

$$\boldsymbol{w}^{k+1} \leftarrow \underbrace{\text{proj}_{\mathcal{S}}}_{\text{Projection}}[\underbrace{\boldsymbol{w}^k - \tau \boldsymbol{r}(\boldsymbol{\theta}^k)}_{\text{Gradient descent}}]$$

$\mathcal{S}$ encodes the constraints $0 \leq w_i \leq 1$, $\mathbf{1}^T\boldsymbol{w} = p - h$.

Yun, Jihun, et al. "Trimming the $\ell_1 $ Regularizer: Statistical Analysis, Optimization, and Applications to Deep Learning." *International Conference on Machine Learning*. 2019.

Optimization of Trimmed ℓ_1

$$\underset{\boldsymbol{\theta}\in\Omega,\,\boldsymbol{w}\in[0,1]^p}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) + \lambda_n \sum_{j=1}^{p} w_j|\theta_j|$$

Objective of θ

$$\text{s.t.} \quad \mathbf{1}^\top \boldsymbol{w} \geq p - h\,. \qquad (1)$$

- Regularized optimization -> Proximal gradient
  - Proximal operator for the regularizer:

$$\text{prox}_{\eta\lambda\mathcal{R}(\cdot\boldsymbol{w}^{k+1})}(\boldsymbol{z}) := \arg\min_{\boldsymbol{\theta}} \frac{1}{2\eta\lambda} \parallel \boldsymbol{\theta} - \boldsymbol{z} \parallel^2 + \textstyle\sum_{j=1}^{p} w_j^{k+1}|\theta_j|$$

  - Have a form of ℓ_1, solution is the soft thresholding operator
- Proximal update step for θ

$$\underbrace{\boldsymbol{\theta}^{k+1} \leftarrow \text{prox}_{\eta\lambda\mathcal{R}(\cdot,\boldsymbol{w}^{k+1})}}_{\text{Proximal (soft thresholding)}}\underbrace{[\boldsymbol{\theta}^k - \eta\nabla\mathcal{L}(\boldsymbol{\theta}^k)]}_{\text{Gradient descent}}$$

Yun, Jihun, et al. "Trimming the $\ell_1 $ Regularizer: Statistical Analysis, Optimization, and Applications to Deep Learning." *International Conference on Machine Learning*. 2019.

Optimization of Trimmed ℓ_1

$$\underset{\boldsymbol{\theta}\in\Omega,\, \boldsymbol{w}\in[0,1]^p}{\text{minimize}} \quad \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) + \lambda_n \sum_{j=1}^{p} w_j|\theta_j|$$

$$\text{s.t.} \quad \mathbf{1}^\top \boldsymbol{w} \geq p - h\,. \qquad (1)$$

- Two operators:
  - Projection for w: $\text{proj}_{\mathcal{S}}(\boldsymbol{z}) := \arg\min_{\boldsymbol{w}\in\mathcal{S}} \frac{1}{2} \parallel \boldsymbol{z} - \boldsymbol{w} \parallel^2$
  - Proximal for θ: $\text{prox}_{\eta\lambda\mathcal{R}(\cdot\boldsymbol{w}^{k+1})}(\boldsymbol{z}) := \arg\min_{\boldsymbol{\theta}} \frac{1}{2\eta\lambda} \parallel \boldsymbol{\theta} - \boldsymbol{z} \parallel^2 + \sum_{j=1}^{p} w_j^{k+1}|\theta_j|$
- Putting everything together

Algorithm 1 Block Coordinate Descent for (1)

Input: λ, η, and τ.
Initialize: $\boldsymbol{\theta}^0$, $\boldsymbol{w}^0$, and $k = 0$.
while not converged **do**
  $\boldsymbol{w}^{k+1} \leftarrow \text{proj}_{\mathcal{S}}[\boldsymbol{w}^k - \tau \boldsymbol{r}(\boldsymbol{\theta}^k)]$
  $\boldsymbol{\theta}^{k+1} \leftarrow \text{prox}_{\eta\lambda\mathcal{R}(\cdot,\boldsymbol{w}^{k+1})}[\boldsymbol{\theta}^k - \eta\nabla\mathcal{L}(\boldsymbol{\theta}^k)]$
  $k \leftarrow k + 1$
end while
Output: $\boldsymbol{\theta}^k$, $\boldsymbol{w}^k$.

Yun, Jihun, et al. "Trimming the $\ell_1 $ Regularizer: Statistical Analysis, Optimization, and Applications to Deep Learning." *International Conference on Machine Learning*. 2019.

Summary of proximal operator

- Proximal operator makes regularized optimization smoother by introducing the proximity term
  - L0 -> projected gradient descent (hard thresholding)
  - L1 -> Soft thresholding
- Soft thresholding reveals the “bias” problem of L1, partially solved by SCAD & MCP etc.
- Trimmed L1 combines hard and soft thresholding for better sparsity inducing performance on DNN models

Moving beyond absolute values

- Previous projection/proximal methods use the absolute value of the weights as pruning/penalization criterion
- What if a small weight element is also useful?
  - Small changes to activation may cause major differences in the final output (i.e. adversarial example)
- Measuring the importance of the weight element
  - How loss changes when setting a weight element to zero

$$L(x, y, \mathbf{\Theta}_{w \leftarrow 0}) = L(x, y, \mathbf{\Theta}) - \frac{\partial L(x, y, \mathbf{\Theta})}{\partial w}(0 - w) + o(w^2)$$

Taylor expansion

$$|L(x, y, \mathbf{\Theta}_{w \leftarrow 0}) - L(x, y, \mathbf{\Theta})| = \boxed{\left|\frac{\partial L(x, y, \mathbf{\Theta})}{\partial w} w\right| = T(x, y, w)}$$

Ignore high-order term

Saliency

Ding, Xiaohan, et al. "Global sparse momentum sgd for pruning very deep neural networks." *Advances in Neural Information Processing Systems*. 2019.

Global Sparse Momentum (GSM) SGD

- Momentum SGD with weight decay for weights with high saliency
- **No gradient update** for low-saliency elements, only do weight decay.

$$B_{m,n}^{(k)} = \begin{cases} 1 & \text{if } T(x, y, W_{m,n}^{(k)}) \geq \text{the } Q\text{-th greatest value in } T(x, y, \boldsymbol{\Theta}^{(k)}) \ , \\ 0 & \text{otherwise} \ . \end{cases}$$

Weight decay

Gradient

Momentum $\boldsymbol{Z}^{(k+1)} \leftarrow \beta \boldsymbol{Z}^{(k)} + \eta \boldsymbol{W}^{(k)} + \boldsymbol{B}^{(k)} \circ \frac{\partial L(x, y, \boldsymbol{\Theta})}{\partial \boldsymbol{W}^{(k)}} ,$

Weight update $\boldsymbol{W}^{(k+1)} \leftarrow \boldsymbol{W}^{(k)} - \alpha \boldsymbol{Z}^{(k+1)} ,$

- Global ranking across all layers
  - Saliency is more comparable across layers, than abs value
  - Enable more flexible sparsity allocation

Ding, Xiaohan, et al. "Global sparse momentum sgd for pruning very deep neural networks." *Advances in Neural Information Processing Systems*. 2019.

Pruning results

Table 1: Pruning results on MNIST.

Model	Result	Base Top1	Pruned Top1	Origin / Remain Params	Compress Ratio	Non-zero Ratio
LeNet-300	Han et al. [21]	98.36	98.41	267K / 22K	12.1×	8.23%
LeNet-300	L-OBS [13]	98.24	98.18	267K / 18.6K	14.2×	7%
LeNet-300	Zhang et al. [56]	98.4	98.4	267K / 11.6K	23.0×	4.34%
LeNet-300	DNS [18]	97.72	98.01	267K / 4.8K	55.6×	1.79%
LeNet-300	**GSM**	**98.19**	**98.18**	**267K / 4.4K**	**60.0×**	**1.66%**
LeNet-5	Han et al. [21]	99.20	99.23	431K / 36K	11.9×	8.35%
LeNet-5	L-OBS [13]	98.73	98.73	431K / 3.0K	14.1×	7%
LeNet-5	Srinivas et al. [47]	99.20	99.19	431K / 22K	19.5×	5.10%
LeNet-5	Zhang et al. [56]	99.2	99.2	431K / 6.05K	71.2×	1.40%
LeNet-5	DNS [18]	99.09	99.09	431K / 4.0K	107.7×	0.92%
LeNet-5	**GSM**	**99.21**	**99.22**	**431K / 3.4K**	**125.0×**	**0.80%**
LeNet-5	**GSM**	**99.21**	**99.06**	**431K / 1.4K**	**300.0×**	**0.33%**

Table 2: Pruning results on CIFAR-10.

Model	Result	Base Top1	Pruned Top1	Origin / Remain Params	Compress Ratio	Non-zero Ratio
ResNet-56	GSM	94.05	94.10	852K / 127K	6.6×	15.0%
ResNet-56	GSM	94.05	93.80	852K / 85K	10.0×	10.0%
DenseNet-40	GSM	93.86	94.07	1002K / 150K	6.6×	15.0%
DenseNet-40	GSM	93.86	94.02	1002K / 125K	8.0×	12.5%
DenseNet-40	GSM	93.86	93.90	1002K / 100K	10.0×	10.0%

Ding, Xiaohan, et al. "Global sparse momentum sgd for pruning very deep neural networks." *Advances in Neural Information Processing Systems*. 2019.

Direct parameter selection

- All previously mentioned methods are aiming for making more parameters closer to zero

- Can we directly select the parameters to be pruned?
  - This is a process for parameter selection, or gating (masking)

$$\theta_j = \tilde{\theta}_j z_j, z_j \in \{0,1\}, \tilde{\theta}_j \neq 0, \left||\theta|\right|_0 = \sum z_j$$

  - To make it trainable with gradient-based optimization, we need a differentiable, nonlinear selection (gating) function
    - Stochastic gate with hard concrete distribution
    - Continuous sparsification

Stochastic binary gates

$$\theta_j = \tilde{\theta}_j z_j, z_j \in \{0,1\}, \tilde{\theta}_j \neq 0, \left|\left|\theta\right|\right|_0 = \sum z_j$$

- Here we consider the stochastic case, where the state of the gate is z_j determined by a Bernoulli random variable

$$q\left(z_j \middle| \pi_j\right) = Bern\left(\pi_j\right)$$

- The ℓ_0 regularization can be converted to

$$\tilde{\theta}^*, \pi^* = argmin_{\tilde{\theta},\pi} \mathcal{L}\left(\tilde{\theta} \cdot z\right) + \lambda \sum \pi_j$$

The Bernoulli distribution is still discrete, cannot pass gradient from z to π directly. Further smoothing is required for optimization, details in the reading material

Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through $ L_0 $ Regularization." *arXiv preprint arXiv:1712.01312* (2017).

Continuous Sparsification

$$\theta_j = \tilde{\theta}_j z_j, z_j \in \{0,1\}, \tilde{\theta}_j \neq 0, ||\theta||_0 = \sum z_j$$

- Stochastic gate method requires multiple steps of smoothing and estimation, leading to large variance and poor scalability to larger models
- Rethink the problem we are facing
  - Training requires the gate to be continuous and differentiable
  - Sparsity requires gate to be binary for the **final model**
- No need to have binary gate throughout training process, can keep it continuous, while gradually moving towards binary as training progresses

Savarese, P., Silva, H., & Maire, M. (2019). Winning the lottery with continuous sparsification. *arXiv preprint arXiv:1912.04427*.
Yuan, X., Savarese, P., & Maire, M. (2020). Growing Efficient Deep Networks by Structured Continuous Sparsification. *arXiv preprint arXiv:2007.15353*.

Continuous Sparsification

- A function converting continuous input towards binary output -> Sigmoid with temperature

$$\sigma(\beta s) = \frac{1}{1 + e^{-\beta s}}, \beta \in [1, \infty)$$

- $\sigma(\beta s)$ is continuously differentiable, yet go towards binary with $\beta \to \infty$, satisfying our need for the mask
- Using $\sigma(\beta s)$ as mask:
  - Original $L = L(\theta) + \lambda \left|\left|\theta\right|\right|_0 = L\left(\tilde{\theta}_j z_j\right) + \lambda \sum z_j, z_j \in \{0,1\}$
  - Converted $L_\beta = L\left(\tilde{\theta}_j \sigma\left(\beta s_j\right)\right) + \lambda \sum \sigma\left(\beta s_j\right), s_j \in \mathbb{R}$
- Increasing β from 1 to ∞ during training, leading to sparse model in the end

Summary of ℓ_0 regularization

- ℓ_0 regularization is an effective method for model pruning, as it directly measures the sparsity, without shrinking larger values, and can result in exact zero parameters
- The hard-concrete distribution provides a particle way of approximating and optimizing the ℓ_0 regularization, but suffers from high variance in estimating the gradient, and is hard to scale to larger problems
- Continuous sparsification makes the minimization of ℓ_0 regularization more stable

Moving beyond pruning

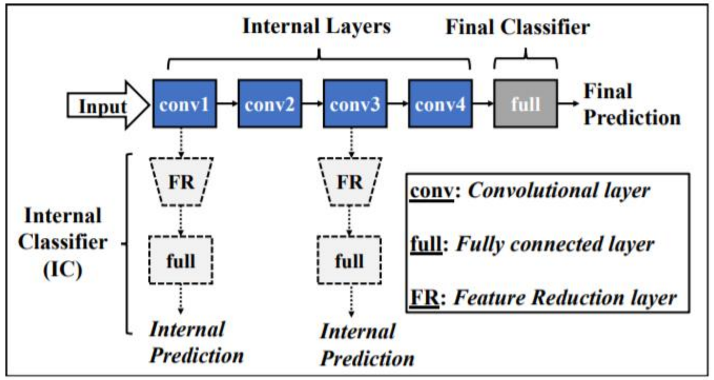
- Pruning aims for removing part of the model
- Often leads to loss in model capacity -> loss in accuracy, leading to efficiency-accuracy tradeoff
- Can we get the best of both worlds?
  - Models are designed in this way for a reason, need the capacity to learn the entire dataset
  - But may not need that large capacity to learn easy data

(a) Red wine (b) Volcano

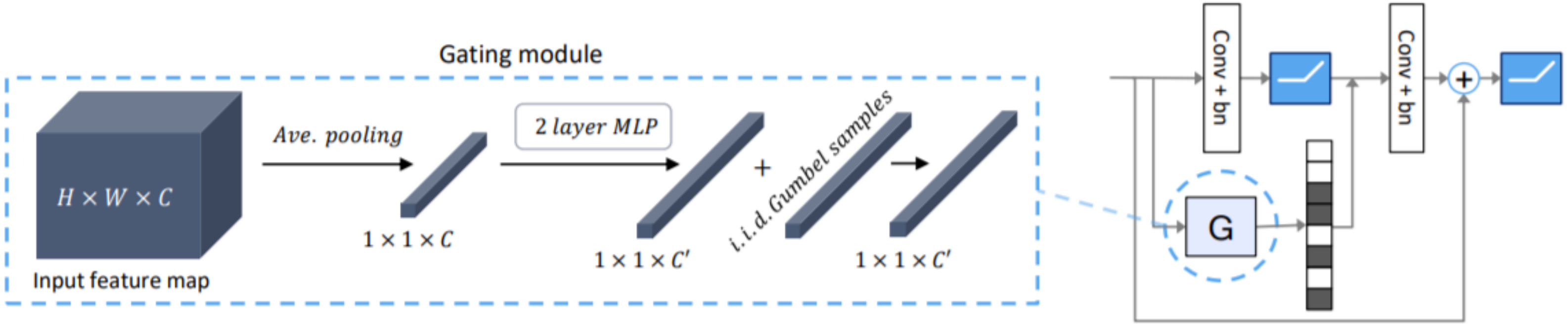
Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., & Weinberger, K. Q. (2017). Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*.

Processing easy data with less effort

- Make the model input-adaptive
  - Multi-exit: make classification with intermediate layer feature if a certain condition is met



  - Dynamic execution: dynamically changing (gating) model architecture for inputs with different features

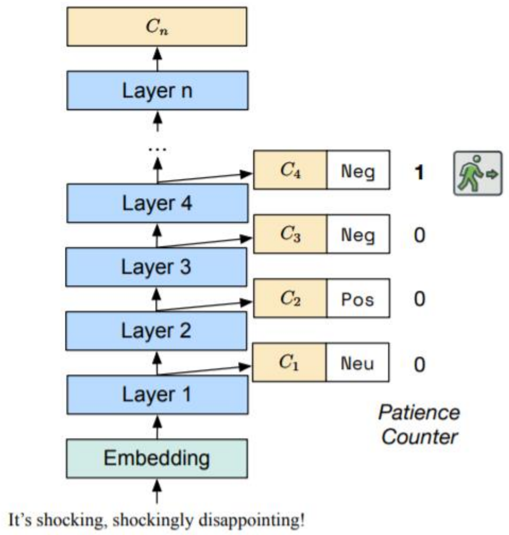


Early exit model

- Model training objective
  - Weighted sum of multi-exits' losses
  - Weighing earlier exit more will encourage earlier exit, but may hurt overall accuracy
- Early exit criteria
  - Entropy threshold
  - Output consistency

$$L_{\text{branchynet}}(\hat{\boldsymbol{y}}, \boldsymbol{y}; \theta) = \sum_{n=1}^{N} w_n L(\hat{\boldsymbol{y}}_{\text{exit}_n}, \boldsymbol{y}; \theta)$$

$$\text{entropy}(\boldsymbol{y}) = \sum_{c \in \mathcal{C}} y_c \log y_c$$

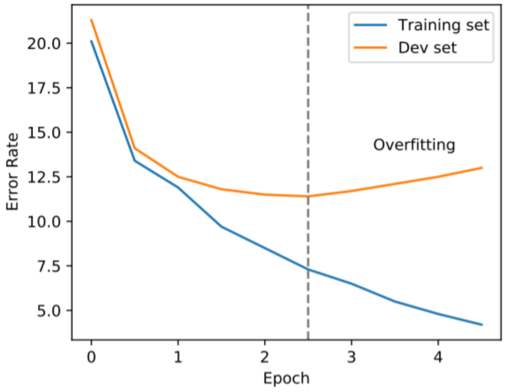


Teerapittayanon, S., McDanel, B., & Kung, H. T. (2016, December). Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)* (pp. 2464-2469). IEEE.

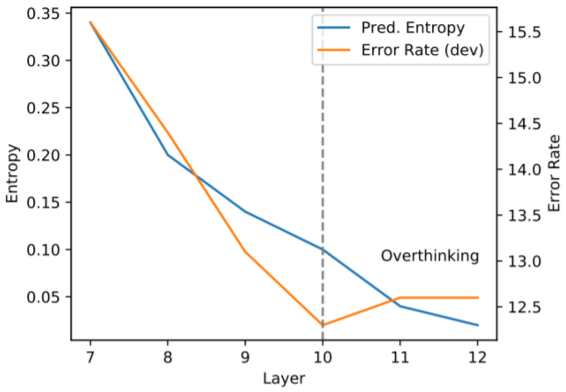
Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., & Wei, F. (2020). Bert loses patience: Fast and robust inference with early exit. *arXiv preprint arXiv:2006.04152*.

Effectiveness of early exiting

- Overcoming overfitting and overthinking



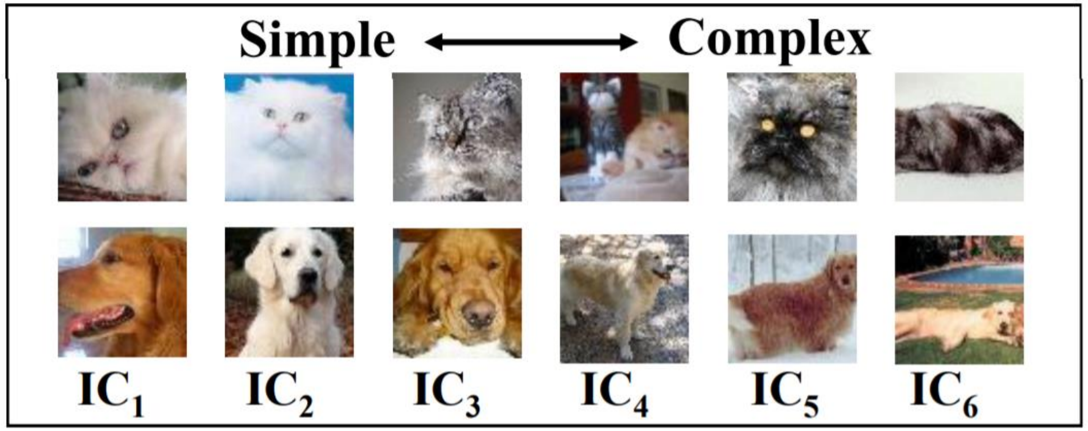
(a) Overfitting in training



(b) Overthinking in inference

Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., & Wei, F. (2020). Bert loses patience: Fast and robust inference with early exit. *arXiv preprint arXiv:2006.04152*.

- Identify easy data at early stage



Kaya, Y., Hong, S., & Dumitras, T. (2019, May). Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning* (pp. 3301-3310). PMLR.

Dynamic model architecture

- Recall in CNN lectures: CNN utilize multiple channels to capture different features in each layer
- The model needs to capture all possible features to fit a dataset well
- Each input may only utilize part of the feature
- Can let the model dynamically decide which channel to use based on the input

Typical gating module

- Gating module decides which channel to choose
- For each layer, the input information is captured from the output feature of previous layer
- Gating module perform a cheap computation based on previous layer’s output
  - Global average pooling + FC
  - L0 reg on the gate to encourage efficiency

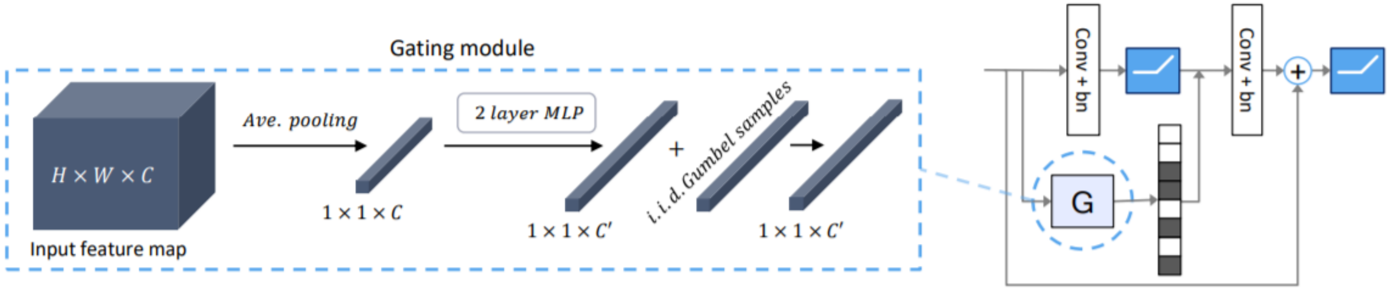
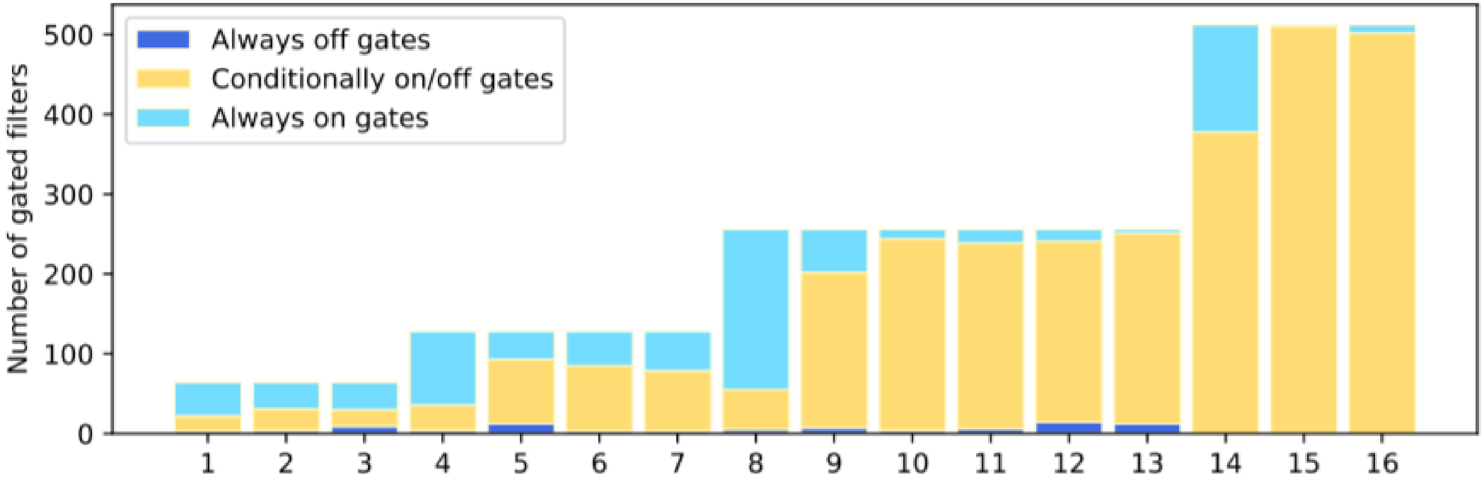


Figure 2: Illustration of our channel gated ResNet block and the gating module.

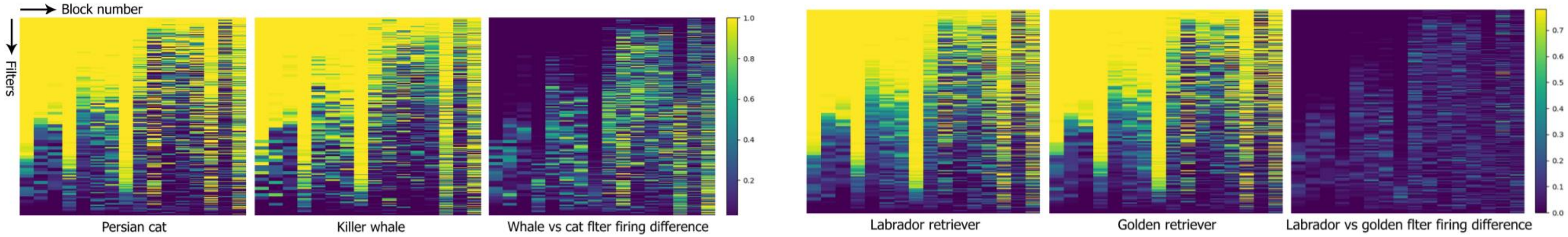
Bejnordi, B. E., Blankevoort, T., & Welling, M. (2019). Batch-shaping for learning conditional channel gated networks. *arXiv preprint arXiv:1907.06627*.

What is learnt in the gate?

- Gate activation pattern: few gates are totally useless, while most of them are used occasionally



- Similar classes activates similar channels



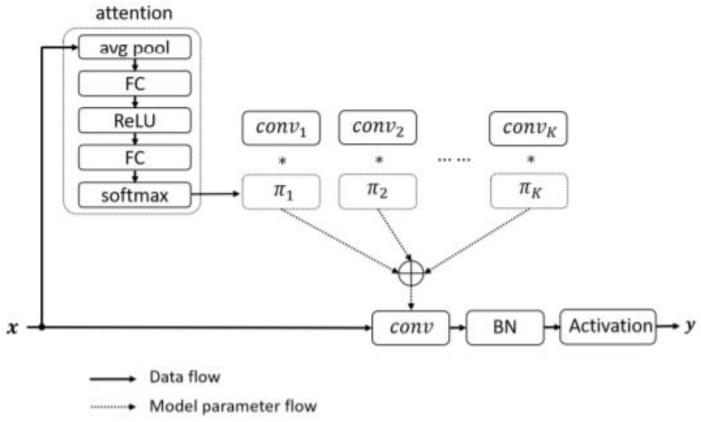
Bejnordi, B. E., Blankevoort, T., & Welling, M. (2019). Batch-shaping for learning conditional channel gated networks. *arXiv preprint arXiv:1907.06627*.

Variant of dynamic model

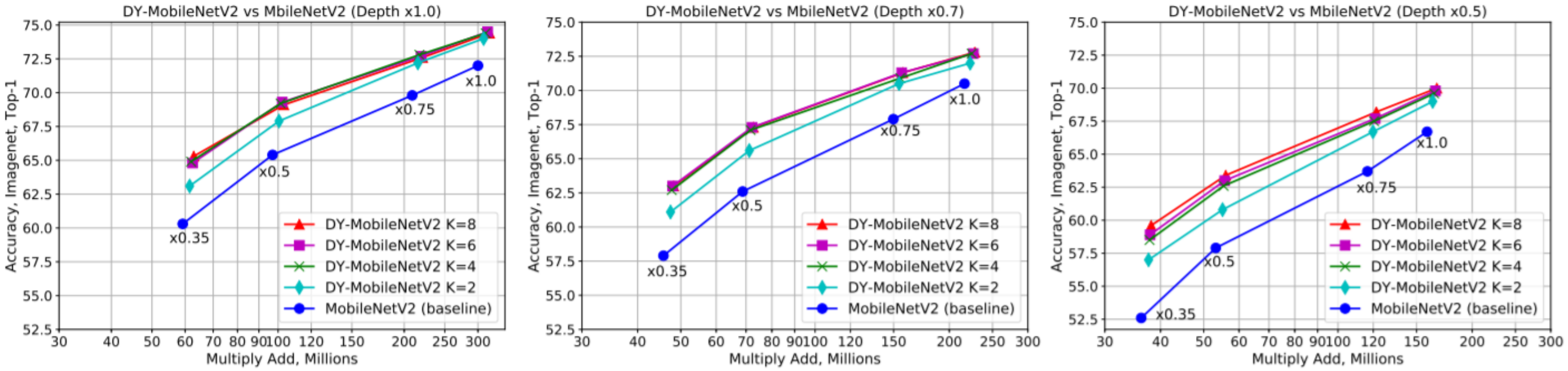
- Dynamic models make inference cost smaller under the same architecture
- Similarly, it will enable the use of a larger model with similar inference cost -> Overparameterization
- Overparameterized model use more parameters in the training process, while dynamically combining them back to the original model size at inference
  - Can learn more diverse features
  - Can converge to lower loss in complex tasks

Case study: Dynamic convolution

- Have multiple trainable kernels in each layer
- Use a weighted sum of all kernels for forward pass, weights generate dynamically based on previous layer's feature



Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020). Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 11030-11039).



In this lecture, we learned:

- Proximal optimization methods
  - Soft thresholding and beyond
  - Trimmed L1
- Alternative pruning criteria
  - Saliency-based pruning
- Continuous smoothing for ℓ_0 optimization
- Dynamic NN model

- See cited papers for details on each technique

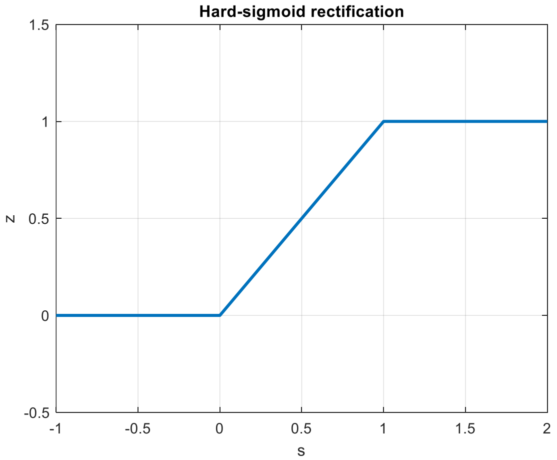
Smoothing the discrete gate (reading)

$$\tilde{\theta}^*, \pi^* = argmin_{\tilde{\theta},\pi} \mathcal{L}(\tilde{\theta} \cdot z) + \lambda \sum \pi_j$$

- The Bernoulli distribution is still discrete, cannot pass gradient from z to π directly
- Consider a random variable s defined by $s \sim q(s|\phi)$, we can smooth z by letting $z = \min(1, \max(0, s))$
  - This is named "hard-sigmoid rectification"
- Recall that $\pi_j = p_r(z \neq 0)$, with

$Q(\cdot)$ being the CDF of s we have:

$$\pi_j = 1 - Q(s_j \leq 0|\phi)$$



Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through $ L_0 $ Regularization." *arXiv preprint arXiv:1712.01312* (2017).

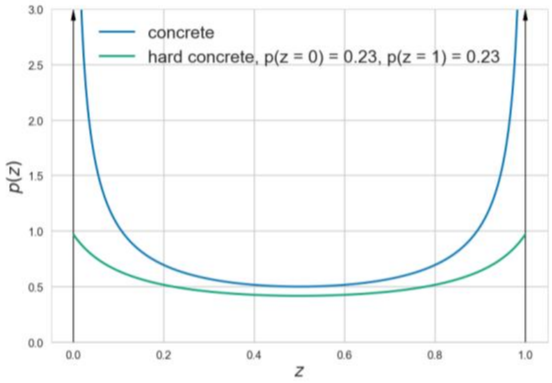
Hard concrete distribution

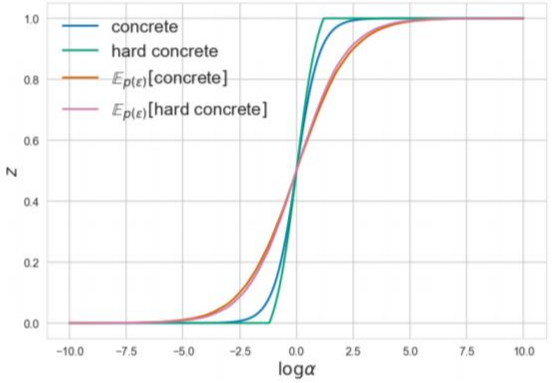
$$\tilde{\theta}^*, \pi^* = argmin_{\tilde{\theta},\pi}\mathcal{L}(\tilde{\theta} \cdot z) + \lambda \sum \pi_j$$

- Now z and π_j are related to s with some differentiable function, only thing left is to define $q(s|\phi)$
- Let s follow a concrete distribution with $\phi = \log \alpha$

$$u \sim \mathcal{U}(0,1), \quad s = \text{Sigmoid}\big((\log u - \log(1-u) + \log\alpha)/\beta\big), \quad \bar{s} = s(\zeta - \gamma) + \gamma$$

- $z = \min(1, \max(0, s))$ will be following a hard-concrete distribution





Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through $ L_0 $ Regularization." *arXiv preprint arXiv:1712.01312* (2017).

Hard concrete ℓ_0 regularization

- The ℓ_0 norm can be relaxed with the hard-concrete distribution as

$$\mathcal{L}_C = \sum_{j=1}^{|\theta|} \left(1 - Q_{\bar{s}_j}(0|\phi)\right) = \sum_{j=1}^{|\theta|} \mathrm{Sigmoid}\left(\log \alpha_j - \beta \log \frac{-\gamma}{\zeta}\right).$$

$$\hat{\mathbf{z}} = \min(\mathbf{1}, \max(\mathbf{0}, \mathrm{Sigmoid}(\log \boldsymbol{\alpha})(\zeta - \gamma) + \gamma))$$

$$\tilde{\theta}^*, \alpha^* = argmin_{\tilde{\theta},\alpha} \mathcal{L}\left(\tilde{\theta} \cdot \hat{z}\right) + \lambda \mathcal{L}_C$$

- Here α is a trainable variable, same size as $\tilde{\theta}$
- The combined loss is differentiable
- $0 < \beta < 1$ is a hyperparameter for the distribution
  - $\beta \to 0$, z will have a high probability to be 0 or 1, closer approximation to original Bernoulli
  - $\beta \to 1$, more density between 0 and 1, a flatter density function leading to easier optimization