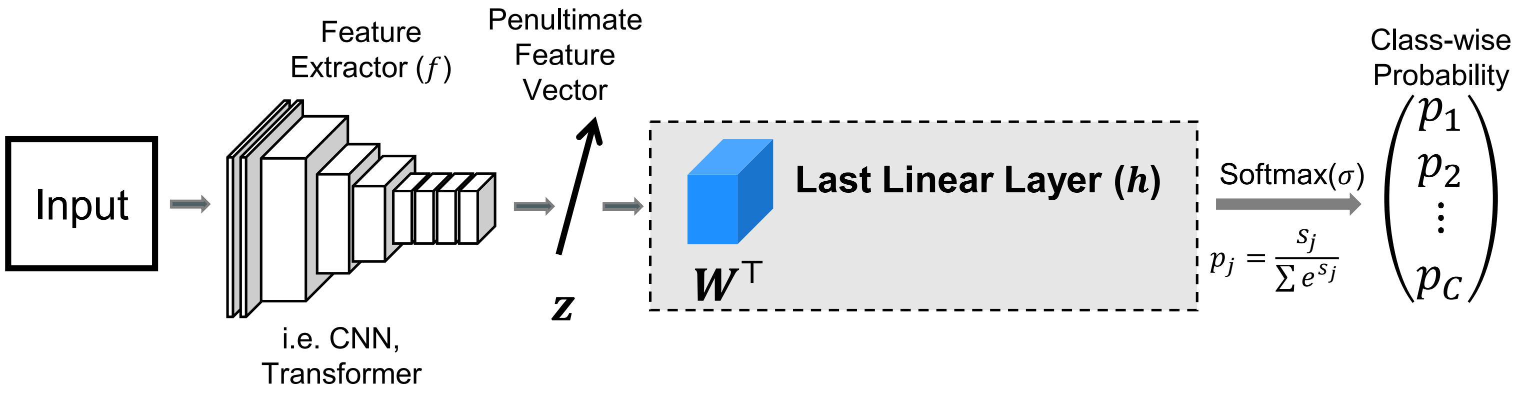
Tilt and Average : Geometric Adjustment of the Last Layer for Recalibration

Gyusang Cho

Chan-Hyun Youn

@ ICML 2024

Confidence



Input

Feature Extractor (f)

i.e. CNN, Transformer

Penultimate Feature Vector

$\boldsymbol{z}$

Last Linear Layer ($\boldsymbol{h}$)

$\boldsymbol{W}^{\top}$

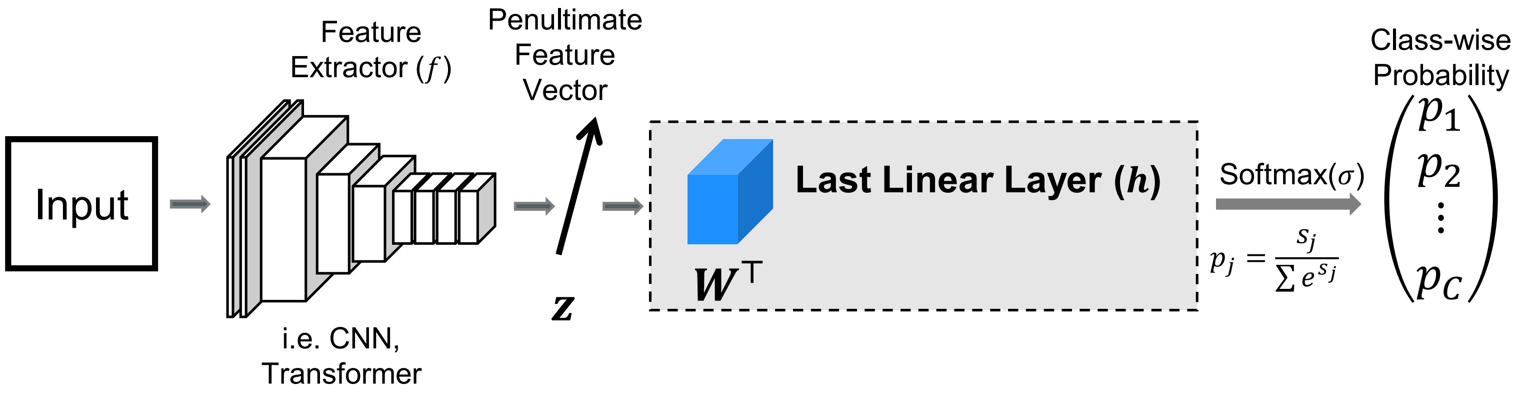
Softmax(σ)

$p_j = \frac{s_j}{\sum e^{s_j}}$

Class-wise Probability

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix}$$

“Uncalibrated” Confidence



Input

Feature Extractor (f)

i.e. CNN, Transformer

Penultimate Feature Vector

$\boldsymbol{z}$

Last Linear Layer ($\boldsymbol{h}$)

$\boldsymbol{W}^{\top}$

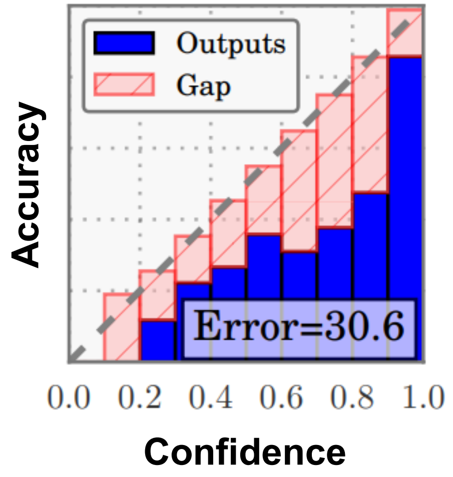
Softmax(σ)

$p_j = \frac{s_j}{\sum e^{s_j}}$

Class-wise Probability

$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix}$

[Guo et al-17’]



“Uncalibrated”

$\sum_{Bins}$ |Accuracy − Confidence| is big

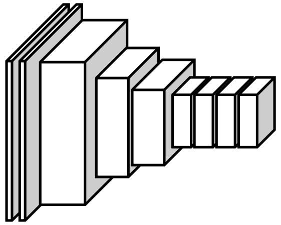
- Neural Network tends to produce overconfident predictions.

- For the predictions to be reliable and trustworthy, “model calibration” is studied to **reflect reliable confidence estimates** to quantify the uncertainty of the prediction.

Previous Approaches for Recalibration Problem

Probability with Original Linear Layer

Input →



Feature Extractor (f)

i.e. CNN, Transformer

→ Penultimate Feature Vector $\boldsymbol{z}$ →

Last Linear Layer (h)

FC-layer Weight $\boldsymbol{W} \in \mathbb{R}^{n\times C}$

Bias

Logit/ Class-wise Score

$$\boldsymbol{W}^\top \quad \begin{array}{cc} \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C,1} & w_{C,2} & \cdots & w_{C,n} \end{pmatrix} & \begin{matrix} \mathbf{w_1} \\ \mathbf{w_2} \\ \\ \mathbf{w_C} \end{matrix} \end{array} \boldsymbol{z} + \begin{pmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_C \end{pmatrix} = \begin{pmatrix} \mathrm{s}_1 \\ \mathrm{s}_2 \\ \vdots \\ \mathrm{s}_C \end{pmatrix}$$

$$\boldsymbol{s} = \boldsymbol{W}^\top \boldsymbol{z} + b$$

Softmax(σ)

$$p_j = \frac{s_j}{\sum e^{s_j}}$$

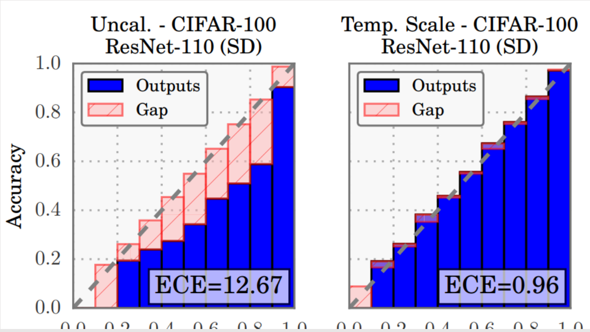
Class-wise Probability

$$\mathbf{p} \quad \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix}$$

→

Calibration Map

$$f(\mathbf{p}) = \mathbf{p}^*, f\colon \Delta^{C-1} \rightarrow \Delta^{C-1}$$



$\mathbf{p}$ $\mathbf{p}^*$

"Uncalibrated" "Calibrated"

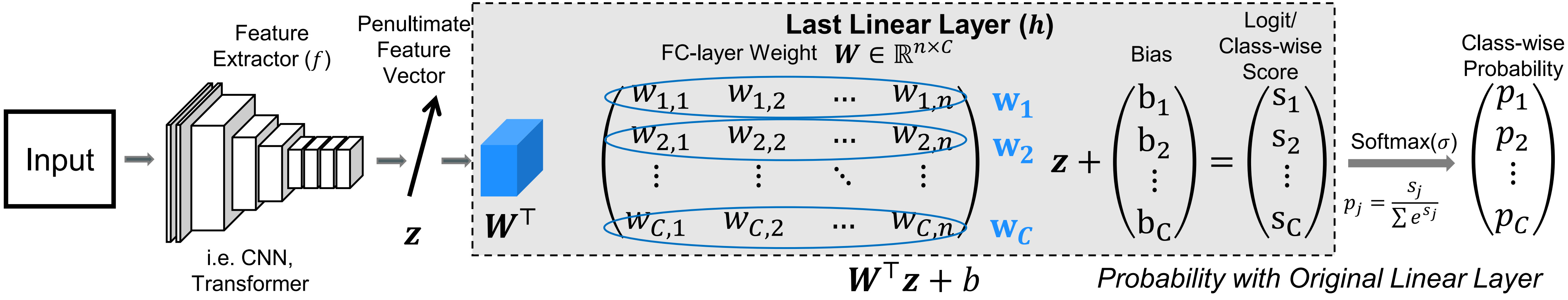
[Guo et al-17']

→

"Calibrated Probability"

$\mathbf{p}^*$

Revisiting Confidence


Input
Feature Extractor (f)
i.e. CNN, Transformer
Penultimate Feature Vector
$\boldsymbol{z}$
$\boldsymbol{W}^\top$
Last Linear Layer (h)
FC-layer Weight $\boldsymbol{W} \in \mathbb{R}^{n\times C}$
$w_{1,1}$ $w_{1,2}$ $\cdots$ $w_{1,n}$
$w_{2,1}$ $w_{2,2}$ $\cdots$ $w_{2,n}$
$w_{C,1}$ $w_{C,2}$ $\cdots$ $w_{C,n}$
$\mathbf{w_1}$
$\mathbf{w_2}$
$\mathbf{w_C}$
$\boldsymbol{z} +$
Bias
b_1 b_2 b_C
$=$
Logit/ Class-wise Score
s_1 s_2 s_C
Softmax(σ)
$p_j = \frac{s_j}{\sum e^{s_j}}$
Class-wise Probability
p_1 p_2 p_C
$\boldsymbol{W}^\top \boldsymbol{z} + b$

Probability with Original Linear Layer

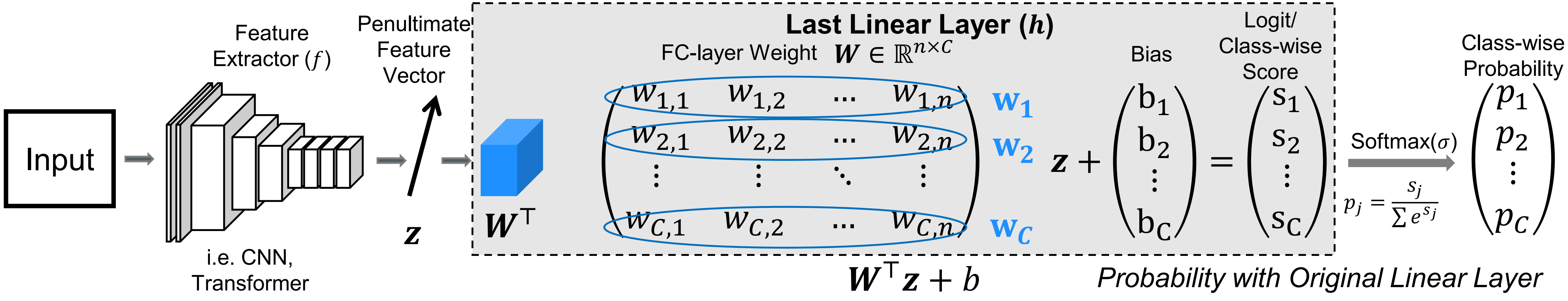
$$\mathbf{s} = \boldsymbol{W}^\top \boldsymbol{z} + \mathbf{b}$$

$$\begin{array}{cc} & \boldsymbol{W} \\ \begin{array}{c} \mathbf{w_1} \\ \mathbf{w_2} \\ \\ \mathbf{w_C} \end{array} & \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C,1} & w_{C,2} & \cdots & w_{C,n} \end{pmatrix} \end{array} \begin{array}{c} \boldsymbol{z} \\ \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} \end{array} + \begin{array}{c} \mathbf{b} \\ \begin{pmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_C \end{pmatrix} \end{array} = \begin{array}{c} \mathbf{s} \\ \begin{pmatrix} \mathrm{s}_1 \\ \mathrm{s}_2 \\ \vdots \\ \mathrm{s}_C \end{pmatrix} \end{array} \xrightarrow[p_j = \frac{e^{s_j}}{\sum e^{s_j}}]{\text{Softmax}} \begin{array}{c} \text{Class-wise Probability} \\ \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix} \end{array} \xrightarrow[\max_{i\in[C]} p_i]{\text{Max}}$$

Confidence = $\max_{i\in[C]} p_i$

Score for class i : $s_i = \underset{\textit{Row vector}}{\mathbf{w_i}} \cdot \boldsymbol{z} + b_i = ||\mathbf{w_i}||\;||\boldsymbol{z}||\cos\angle(\mathbf{w_i}, \boldsymbol{z}) + b_i$

Revisiting Confidence



Input

Feature Extractor (f)

i.e. CNN, Transformer

Penultimate Feature Vector

z

Last Linear Layer (h)

FC-layer Weight $W \in \mathbb{R}^{n\times C}$

$W^\top$

Bias

Logit/ Class-wise Score

Softmax(σ)

Class-wise Probability

$$\begin{array}{c} \\ \mathbf{w_1} \\ \mathbf{w_2} \\ \\ \mathbf{w_C} \end{array} \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C,1} & w_{C,2} & \cdots & w_{C,n} \end{pmatrix} z + \begin{pmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_C \end{pmatrix} = \begin{pmatrix} \mathrm{s}_1 \\ \mathrm{s}_2 \\ \vdots \\ \mathrm{s}_C \end{pmatrix} \xrightarrow[p_j = \frac{s_j}{\sum e^{s_j}}]{\text{Softmax}(\sigma)} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix}$$

$W^\top z + b$

Probability with Original Linear Layer

$$\mathbf{s} = W^\top z + \mathbf{b}$$

$$\begin{array}{c} \\ \mathbf{w_1} \\ \mathbf{w_2} \\ \\ \mathbf{w_C} \end{array} \overset{W}{\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C,1} & w_{C,2} & \cdots & w_{C,n} \end{pmatrix}} \overset{z}{\begin{pmatrix} pf_1 \\ pf_2 \\ \vdots \\ pf_n \end{pmatrix}} + \overset{\mathbf{b}}{\begin{pmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \vdots \\ \mathrm{b}_C \end{pmatrix}} = \overset{\mathbf{s}}{\begin{pmatrix} \mathrm{s}_1 \\ \mathrm{s}_2 \\ \vdots \\ \mathrm{s}_C \end{pmatrix}} \xrightarrow[p_j = \frac{e^{s_j}}{\sum e^{s_j}}]{\text{Softmax}} \overset{\text{Class-wise Probability}}{\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix}} \xrightarrow[\max\limits_{i\in[C]} p_i]{\text{Max}}$$

Confidence = $\max\limits_{i\in[C]} p_i$

Score for class i : $s_i = \boxed{\mathbf{w_i}} \cdot z + b_i = ||\mathbf{w_i}||\ ||z|| \cos\angle(\mathbf{w_i}, z) + b_i$

≜ *Class vector*

Vector

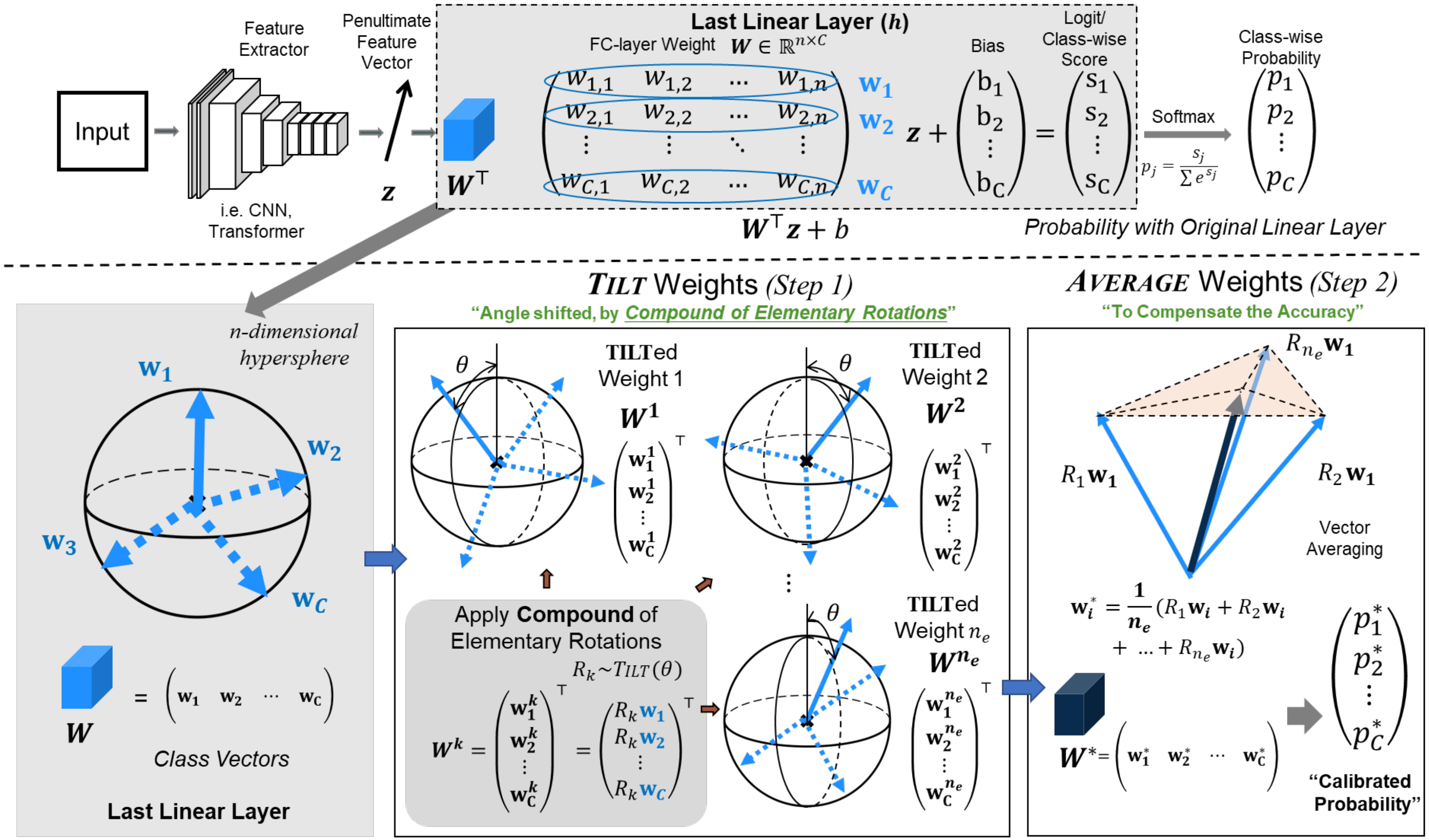
Magnitude of row vectors

Angles of row vectors

We may manipulate the two aspects : Magnitude, Angle

Method Overview : Tilt and Average (TNA)

High Dimensional Geometry for tuning Last Linear Layer!



Last Linear Layer (h)

Probability with Original Linear Layer

TILT* Weights *(Step 1)

"Angle shifted, by ***Compound of Elementary Rotations***"

AVERAGE* Weights *(Step 2)

"To Compensate the Accuracy"

Input
Feature Extractor
i.e. CNN, Transformer
Penultimate Feature Vector
z
$W^\top$
FC-layer Weight $W \in \mathbb{R}^{n \times C}$
$\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{C,1} & w_{C,2} & \cdots & w_{C,n} \end{pmatrix}$
$\mathbf{w}_1$
$\mathbf{w}_2$
$\mathbf{w}_C$
$z +$
Bias
$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_C \end{pmatrix}$
$=$
Logit/ Class-wise Score
$\begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_C \end{pmatrix}$
Softmax
$p_j = \frac{s_j}{\sum e^{s_j}}$
Class-wise Probability
$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_C \end{pmatrix}$
$W^\top z + b$
n-dimensional hypersphere
$\mathbf{w}_1$
$\mathbf{w}_2$
$\mathbf{w}_3$
$\mathbf{w}_C$
W
$= (\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_C)$
Class Vectors
Last Linear Layer
θ
TILTed Weight 1
W^1
$\begin{pmatrix} \mathbf{w}_1^1 \\ \mathbf{w}_2^1 \\ \vdots \\ \mathbf{w}_C^1 \end{pmatrix}^\top$
TILTed Weight 2
W^2
$\begin{pmatrix} \mathbf{w}_1^2 \\ \mathbf{w}_2^2 \\ \vdots \\ \mathbf{w}_C^2 \end{pmatrix}^\top$
⋮
TILTed Weight n_e
W^{n_e}
$\begin{pmatrix} \mathbf{w}_1^{n_e} \\ \mathbf{w}_2^{n_e} \\ \vdots \\ \mathbf{w}_C^{n_e} \end{pmatrix}^\top$
Apply **Compound** of Elementary Rotations
$R_k \sim \textit{TILT}(\theta)$
$W^k = \begin{pmatrix} \mathbf{w}_1^k \\ \mathbf{w}_2^k \\ \vdots \\ \mathbf{w}_C^k \end{pmatrix}^\top = \begin{pmatrix} R_k \mathbf{w}_1 \\ R_k \mathbf{w}_2 \\ \vdots \\ R_k \mathbf{w}_C \end{pmatrix}^\top$
$R_{n_e}\mathbf{w}_1$
$R_1 \mathbf{w}_1$
$R_2 \mathbf{w}_1$
Vector Averaging
$\mathbf{w}_i^* = \frac{1}{n_e}(R_1\mathbf{w}_i + R_2\mathbf{w}_i + \ldots + R_{n_e}\mathbf{w}_i)$
$W^* = (\mathbf{w}_1^* \ \mathbf{w}_2^* \ \cdots \ \mathbf{w}_C^*)$
$\begin{pmatrix} p_1^* \\ p_2^* \\ \vdots \\ p_C^* \end{pmatrix}$
"Calibrated Probability"

How do we manipulate “Angle”?

Idea : ***TILT*** with Rotation Transformation

Q) How do we TILT the class vectors in high-dimensional space?

How do we manipulate “Angle”?

Idea : ***TILT*** with Rotation Transformation

Q) How do we TILT the class vectors in high-dimensional space?

> ***(Euler’s Rotation Theorem)*** *When a sphere is moved around its cenere it is always possible to find a diameter whose direction in the displaced position is the same as in the initial position.*
> *In other words, every rotation in 3-dimensional space can be written as,*
>
> $$R = R_x(\alpha)R_y(\beta)R_z(\gamma).$$
>
> $$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}, R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, R_z(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix}$$

$$R = R_1(\theta_{t_1})R_2(\theta_{t_2})\ldots R_{n_r}(\theta_{t_{n_r}}),$$

$$R_i(\theta_{t_i}) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta_{t_i} & \cdots & \sin\theta_{t_i} & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta_{t_i} & \cdots & \cos\theta_{t_i} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

We do this in n-dimensional space **(Givens Rotation)**

How do we manipulate “Angle”?

Idea : ***TILT*** with Rotation Transformation

Q) How do we control the intensity of TILT?

How do we manipulate “Angle”?

Idea : *TILT* with Rotation Transformation

Q) How do we control the intensity of TILT?

Definition. (**mean Rotation over Classes**) *Given the original weight* $\mathbf{W}$ *and a rotation matrix* R, *the mean Rotation over Classes*(mRC) *is defined as,*

$$mRC(\mathbf{W}, R) = \frac{1}{C}\sum_{i=1}^{C} \arccos \frac{< \mathbf{w}_i, R\mathbf{w}_i >}{||\mathbf{w}_i|| ||R\mathbf{w}_i||}$$

$$R = R_1(\theta_{t_1})R_2(\theta_{t_2})...R_{n_r}(\theta_{t_{n_r}}),$$

$$R_i(\theta_{t_i}) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta_{t_i} & \cdots & \sin\theta_{t_i} & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta_{t_i} & \cdots & \cos\theta_{t_i} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

How do we manipulate “Angle”?

Idea : *TILT* with Rotation Transformation

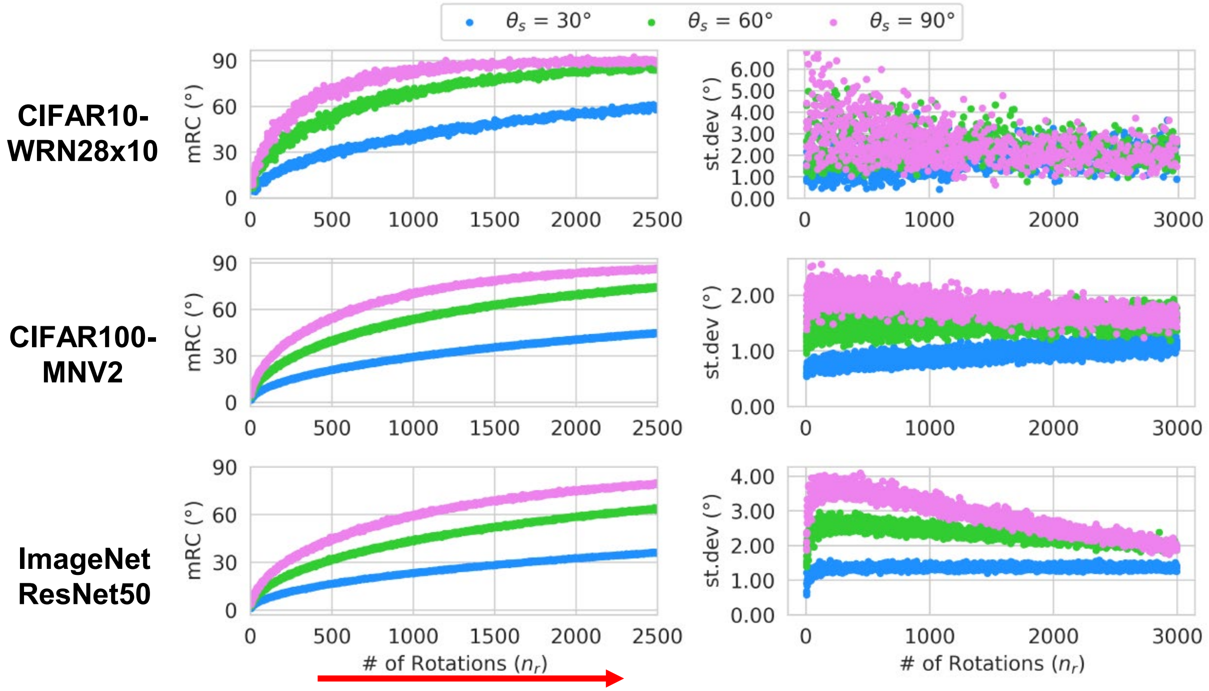
Q) How do we control the intensity of TILT?

Definition. (**mean Rotation over Classes**) *Given the original weight* $\mathbf{W}$ *and a rotation matrix* R, *the mean Rotation over Classes*(mRC) *is defined as,*

$$mRC(\mathbf{W}, R) = \frac{1}{C}\sum_{i=1}^{C} \arccos \frac{< \mathbf{w}_i, R\mathbf{w}_i >}{||\mathbf{w}_i||||R\mathbf{w}_i||}$$

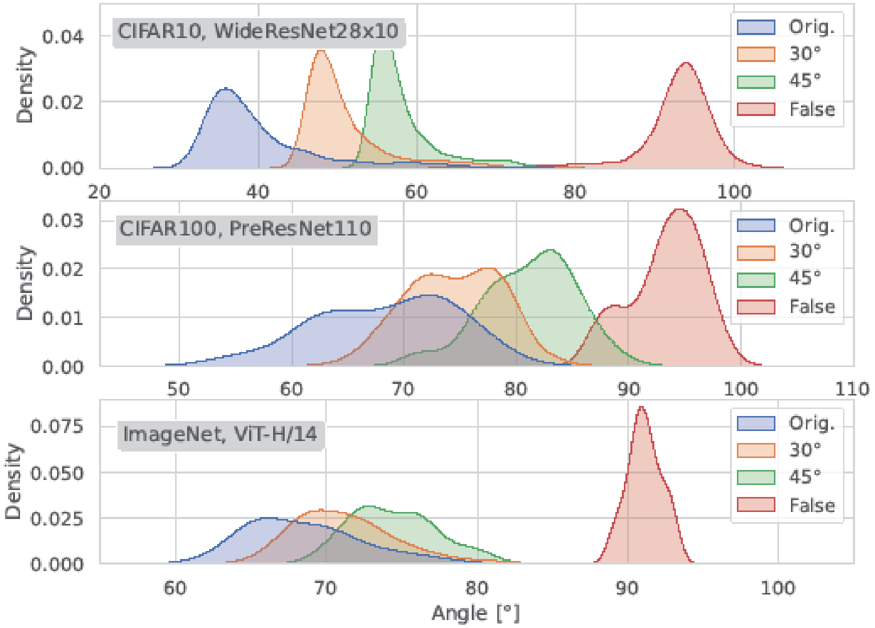
$$R = R_1(\theta_{t_1})R_2(\theta_{t_2})...R_{n_r}(\theta_{t_{n_r}}),$$

$$R_i(\theta_{t_i}) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta_{t_i} & \cdots & \sin\theta_{t_i} & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta_{t_i} & \cdots & \cos\theta_{t_i} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$



mRC reaches 90° as n_r increases

cf) Near Orthogonal Theorem : Two arbitrary vectors are likely to be orthogonal



Angle Shifting Effect

$\angle(\boldsymbol{w}_i, \boldsymbol{z})$ density plot for each dataset

How do we manipulate “Angle”?

Idea : *TILT* with Rotation Transformation

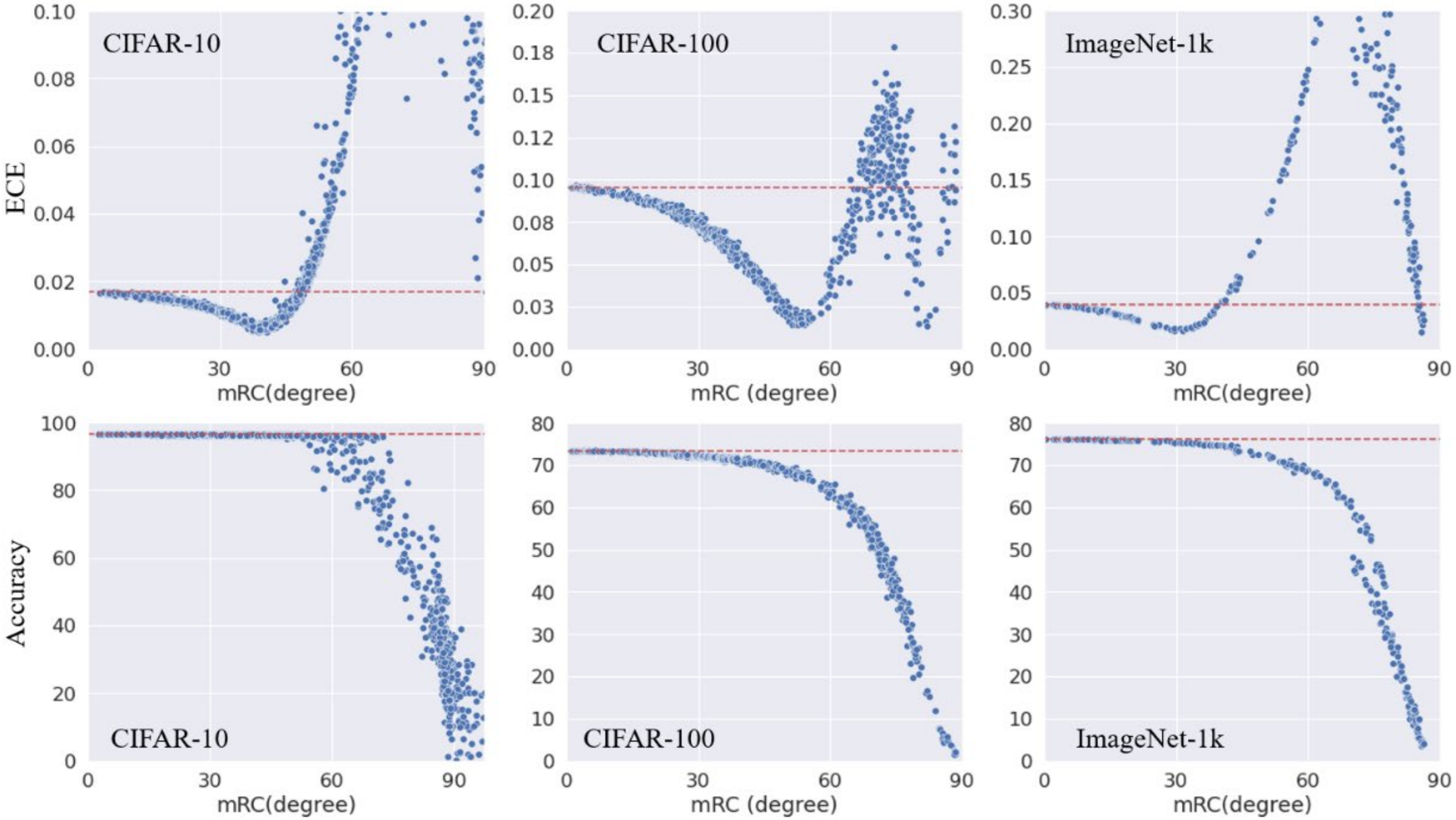
Theorem 3.2 (**Class-wise effect of TILT.**) *Let there be an original weight $\boldsymbol{W}$ and rotation matrix R. Also, let ψ_i to be $\angle(\boldsymbol{w}_i, \boldsymbol{z})$. Suppose the rotation matrix R rotates the i-th class vector with θ, $\angle(\boldsymbol{w}_i, R\boldsymbol{w}_i\,) = \theta$. We further assume that inequality $0^\circ < \theta < \psi_i < 90^\circ$ holds for penultimate feature $\boldsymbol{z}$. Lastly, we assume equal probability on the possible rotations of R. That is, let V be the set of vectors rotated from a vector u by all possible rotation matrices R, that rotates with angle θ, then for $\forall\, v_1, v_2 \in V$, $\mathbb{P}\,(Ru = v_1) = \mathbb{P}(Ru = v_2)$. Let $\mathbb{M}$ be the mode of $\Delta_{\boldsymbol{z},i}$. Then the equation below holds,*

$$\mathbb{M}\left[\Delta_{\boldsymbol{z},i}\right] = \arccos(\cos\psi_i \cos\theta) - \psi_i$$

Proposition 3.3 (**Confidence Relaxation.**) *For an input sample X and the corresponding penultimate feature $\boldsymbol{z}$, we assume the equalities below hold across all the classes in addition to the assumptions made in Theorem 3.2, $\forall i \in [C], \angle(\boldsymbol{w}_i, R\boldsymbol{w}_i) = \theta$ and $\Delta_{\boldsymbol{z},i} = arccos(cos\,\psi_i\, cos\,\theta) - \psi_i$, and element of bias vector be $b_{k_1} = b_{k_2}, \forall k_1, k_2 \in [C]$. Then the tilted weight $\boldsymbol{W}' = RW$ has a smaller confidence estimate for sample X.*

Check for the proof in our paper.

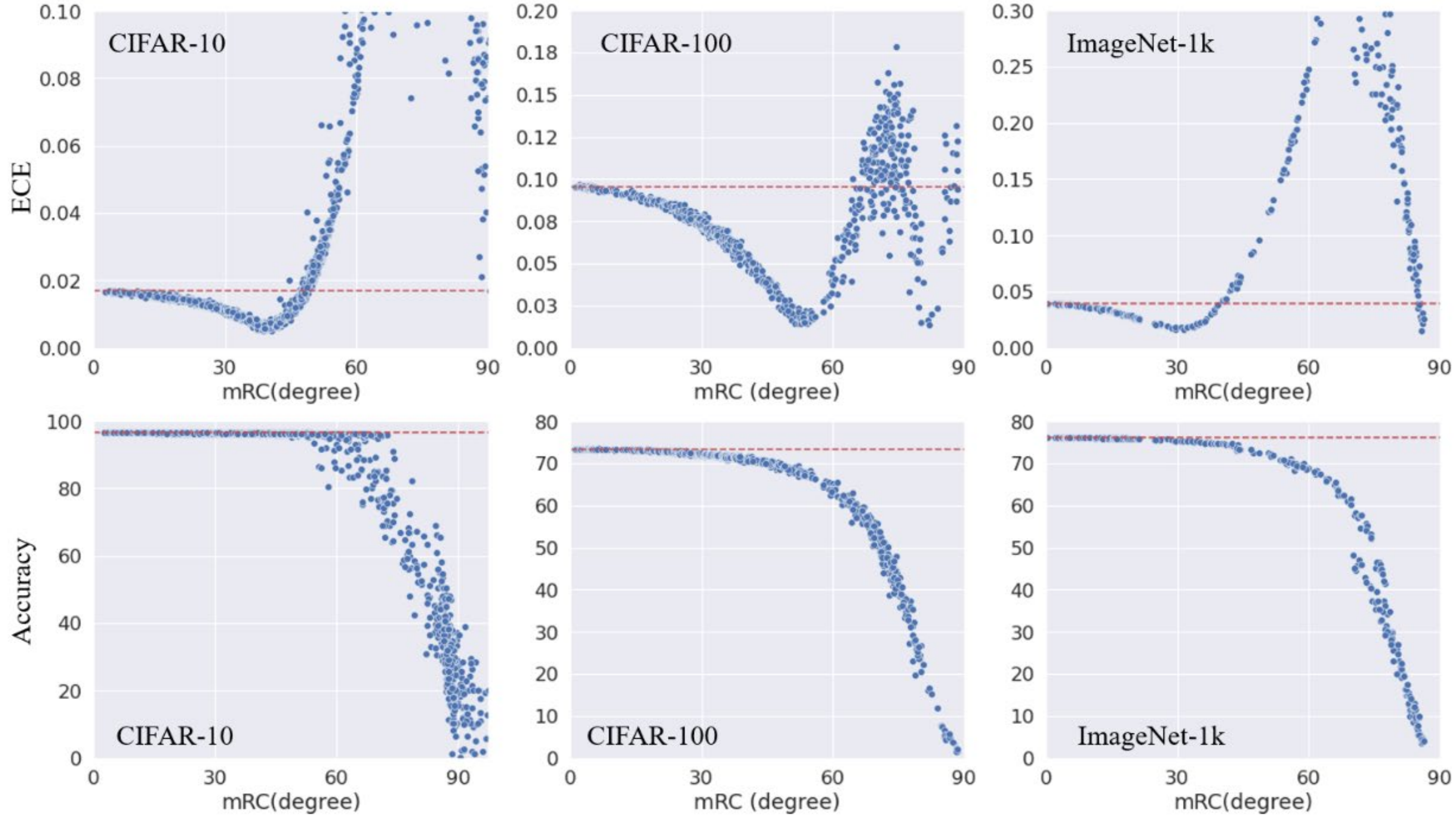
Confidence is relaxed, but….



Each point is a randomly generated alternate classification-head with different mRC.

Accuracy collapses as mRC increases.

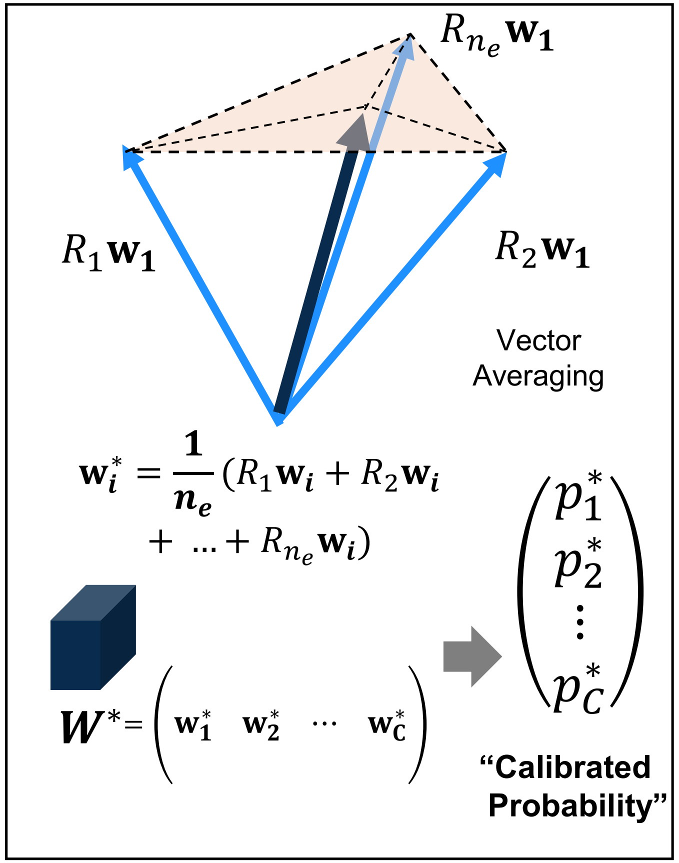
Confidence is relaxed, but….



Each point is a randomly generated alternate classification-head with different mRC.

Accuracy collapses as mRC increases.

AVERAGE;
Vector Averaging on equally tilted weights



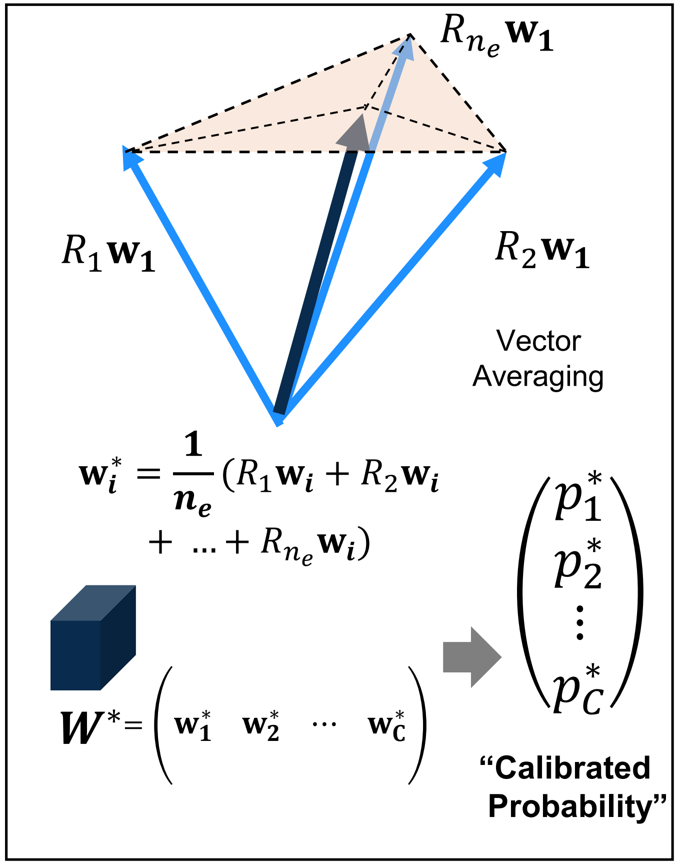
$$\mathbf{w}_i^* = \frac{\mathbf{1}}{\boldsymbol{n_e}}(R_1\mathbf{w_i} + R_2\mathbf{w_i} + \dots + R_{n_e}\mathbf{w_i})$$

$$\boldsymbol{W}^* = \begin{pmatrix} \mathrm{w}_1^* & \mathrm{w}_2^* & \cdots & \mathrm{w}_C^* \end{pmatrix} \rightarrow \begin{pmatrix} p_1^* \\ p_2^* \\ \vdots \\ p_C^* \end{pmatrix}$$

"Calibrated Probability"

Compensate Accuracy with AVERAGE

AVERAGE;
Vector Averaging on
equally tilted weights



$$\mathbf{w}_i^* = \frac{\mathbf{1}}{\boldsymbol{n_e}}(R_1\mathbf{w}_i + R_2\mathbf{w}_i + \ldots + R_{n_e}\mathbf{w}_i)$$

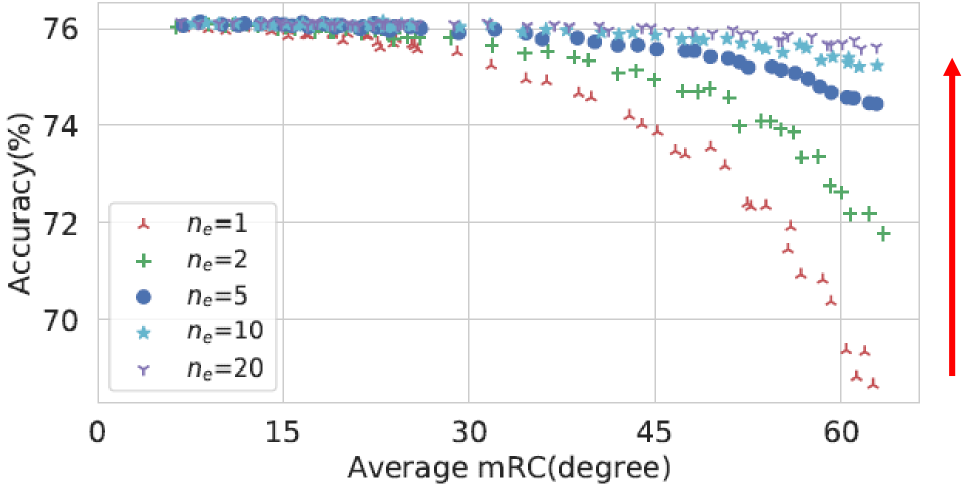
$$\boldsymbol{W}^* = \begin{pmatrix} \mathbf{w}_1^* & \mathbf{w}_2^* & \cdots & \mathbf{w}_C^* \end{pmatrix}$$

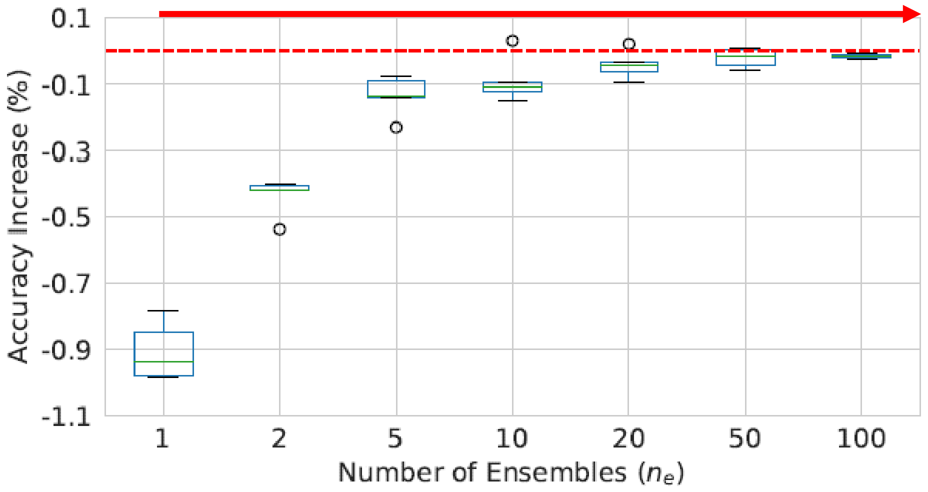
$$\begin{pmatrix} p_1^* \\ p_2^* \\ \vdots \\ p_C^* \end{pmatrix}$$

“Calibrated Probability”

Insight; Geometric Mean is applied on softmax members $\exp(s_j^*)$ when ignoring bias

$$\exp(s_j^*) = \exp(\mathbf{w}_\mathbf{i}^* \cdot \boldsymbol{z}) = \exp\left(\frac{\mathbf{1}}{\boldsymbol{n_e}}(R_1\mathbf{w}_i + R_2\mathbf{w}_i + \ldots + R_{n_e}\mathbf{w}_i) \cdot \boldsymbol{z}\right)$$

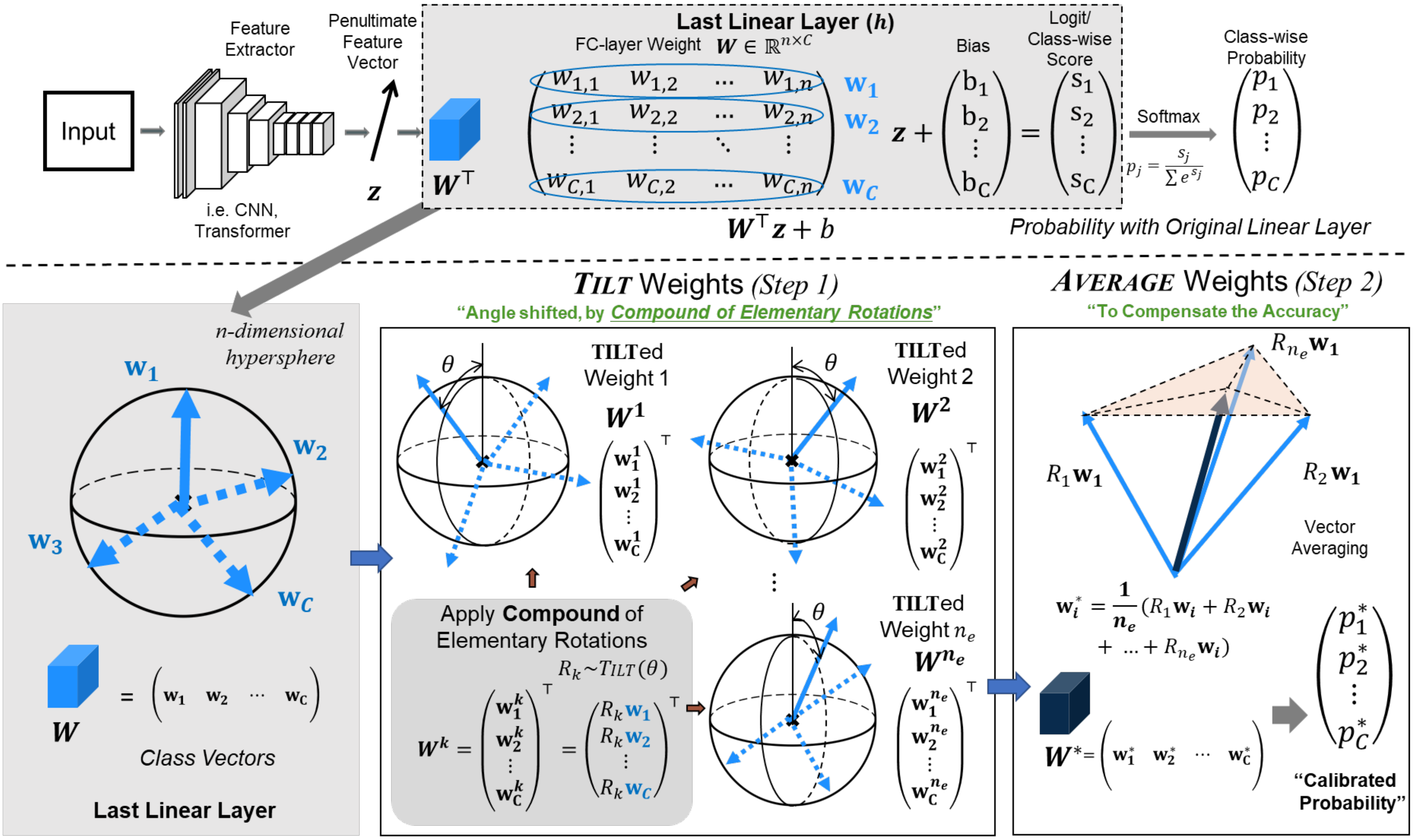




Accuracy is compensated as n_e increases

Method Overview : Tilt and Average (TNA)

High Dimensional Geometry for tuning Last Linear Layer!



Input
Feature Extractor
i.e. CNN, Transformer
Penultimate Feature Vector
z
Last Linear Layer (h)
FC-layer Weight $W \in \mathbb{R}^{n \times C}$
$W^\top$
$w_{1,1}$ $w_{1,2}$ … $w_{1,n}$
$w_{2,1}$ $w_{2,2}$ … $w_{2,n}$
$w_{C,1}$ $w_{C,2}$ … $w_{C,n}$
$\mathbf{w}_1$ $\mathbf{w}_2$ $\mathbf{w}_C$
Bias
Logit/ Class-wise Score
$z + (b_1, b_2, \ldots, b_C) = (s_1, s_2, \ldots, s_C)$
$W^\top z + b$
Softmax
$p_j = \frac{s_j}{\sum e^{s_j}}$
Class-wise Probability
p_1 p_2 ⋮ p_C
Probability with Original Linear Layer
n-dimensional hypersphere
$\mathbf{w}_1$ $\mathbf{w}_2$ $\mathbf{w}_3$ $\mathbf{w}_C$
$W = (\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_C)$
Class Vectors
Last Linear Layer
TILT Weights (Step 1)
"Angle shifted, by Compound of Elementary Rotations"
θ
TILTed Weight 1
W^1
$(\mathbf{w}_1^1, \mathbf{w}_2^1, \ldots, \mathbf{w}_C^1)^\top$
TILTed Weight 2
W^2
$(\mathbf{w}_1^2, \mathbf{w}_2^2, \ldots, \mathbf{w}_C^2)^\top$
TILTed Weight n_e
W^{n_e}
$(\mathbf{w}_1^{n_e}, \mathbf{w}_2^{n_e}, \ldots, \mathbf{w}_C^{n_e})^\top$
Apply Compound of Elementary Rotations
$R_k \sim \text{TILT}(\theta)$
$W^k = (\mathbf{w}_1^k, \mathbf{w}_2^k, \ldots, \mathbf{w}_C^k)^\top = (R_k\mathbf{w}_1, R_k\mathbf{w}_2, \ldots, R_k\mathbf{w}_C)^\top$
AVERAGE Weights (Step 2)
"To Compensate the Accuracy"
$R_{n_e}\mathbf{w}_1$
$R_1\mathbf{w}_1$
$R_2\mathbf{w}_1$
Vector Averaging
$\mathbf{w}_i^* = \frac{1}{n_e}(R_1\mathbf{w}_i + R_2\mathbf{w}_i + \ldots + R_{n_e}\mathbf{w}_i)$
$W^* = (\mathbf{w}_1^* \ \mathbf{w}_2^* \ \cdots \ \mathbf{w}_C^*)$
p_1^* p_2^* ⋮ p_C^*
"Calibrated Probability"

Evaluation

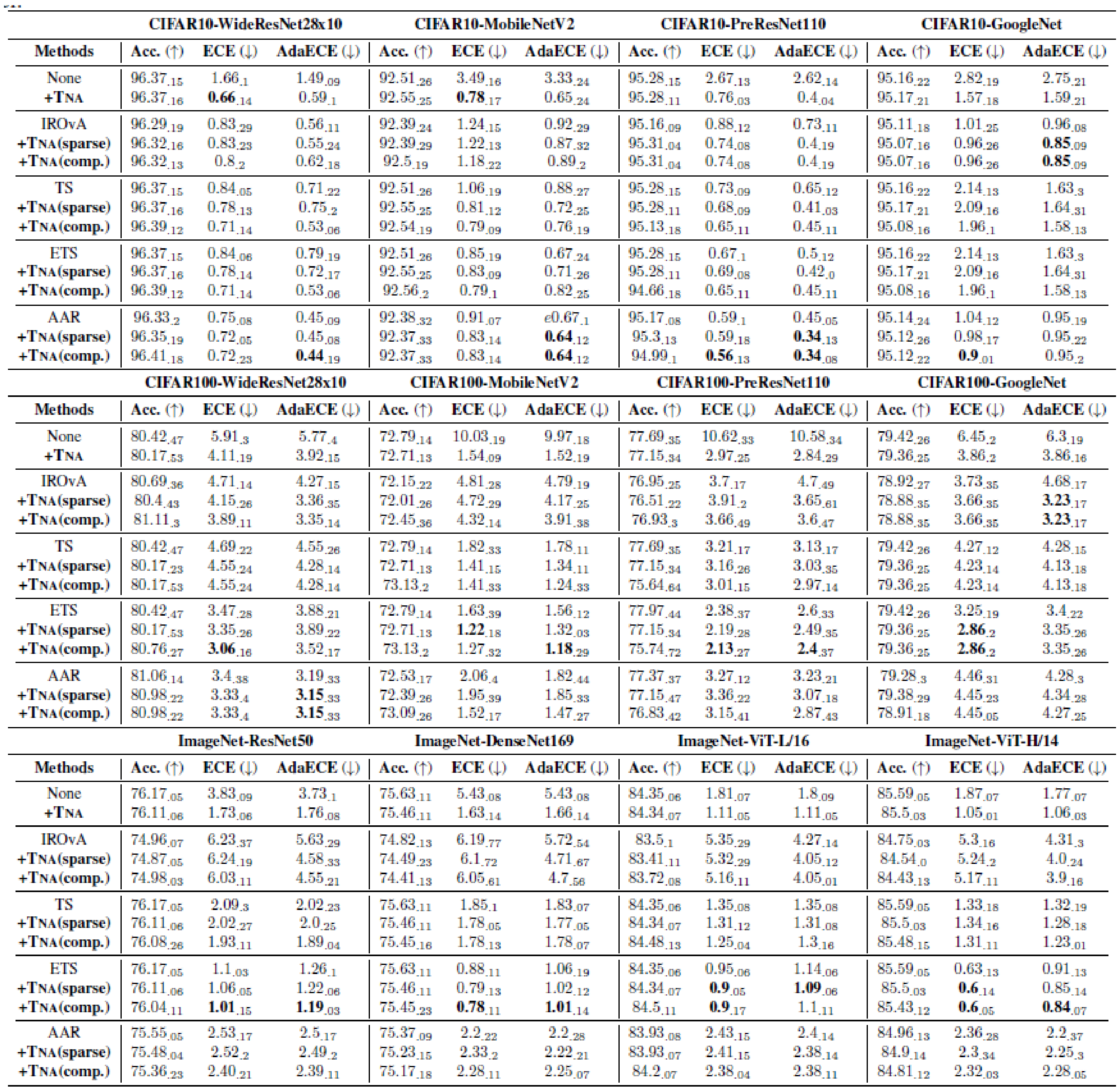
Methods

- TNA(sparse.) : Optimize TNA first (Find Angle), and then apply the method, takes few seconds on a single GPU
- TNA(comp.) : Grid Search over the best set for all possible (W',f), takes few minutes on a single GPU

Metrics

- ECE(Expected Calibration Error): estimates calibration error by *equal-interval-binning* scheme.
- AdaECE(Adaptive Expected Calibation Error): estimates calibration error by *uniform-mass-binning* scheme.

Lower the better!



	CIFAR10-WideResNet28x10			CIFAR10-MobileNetV2			CIFAR10-PreResNet110			CIFAR10-GoogleNet		
Methods	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**
None	$96.37_{.15}$	$1.66_{.1}$	$1.49_{.09}$	$92.51_{.26}$	$3.49_{.16}$	$3.33_{.24}$	$95.28_{.15}$	$2.67_{.13}$	$2.62_{.14}$	$95.16_{.22}$	$2.82_{.19}$	$2.75_{.21}$
+TNA	$96.37_{.16}$	$\mathbf{0.66}_{.14}$	$0.59_{.1}$	$92.55_{.25}$	$\mathbf{0.78}_{.17}$	$0.65_{.24}$	$95.28_{.11}$	$0.76_{.03}$	$0.4_{.04}$	$95.17_{.21}$	$1.57_{.18}$	$1.59_{.21}$
IROvA	$96.29_{.19}$	$0.83_{.29}$	$0.56_{.11}$	$92.39_{.24}$	$1.24_{.15}$	$0.92_{.29}$	$95.16_{.09}$	$0.88_{.12}$	$0.73_{.11}$	$95.11_{.18}$	$1.01_{.25}$	$0.96_{.08}$
+TNA(sparse)	$96.32_{.16}$	$0.83_{.23}$	$0.55_{.24}$	$92.39_{.29}$	$1.22_{.13}$	$0.87_{.32}$	$95.31_{.04}$	$0.74_{.08}$	$0.4_{.19}$	$95.07_{.16}$	$0.96_{.26}$	$\mathbf{0.85}_{.09}$
+TNA(comp.)	$96.32_{.13}$	$0.8_{.2}$	$0.62_{.18}$	$92.5_{.19}$	$1.18_{.22}$	$0.89_{.2}$	$95.31_{.04}$	$0.74_{.08}$	$0.4_{.19}$	$95.07_{.16}$	$0.96_{.26}$	$\mathbf{0.85}_{.09}$
TS	$96.37_{.15}$	$0.84_{.05}$	$0.71_{.22}$	$92.51_{.26}$	$1.06_{.19}$	$0.88_{.27}$	$95.28_{.15}$	$0.73_{.09}$	$0.65_{.12}$	$95.16_{.22}$	$2.14_{.13}$	$1.63_{.3}$
+TNA(sparse)	$96.37_{.16}$	$0.78_{.13}$	$0.75_{.2}$	$92.55_{.25}$	$0.81_{.12}$	$0.72_{.25}$	$95.28_{.11}$	$0.68_{.09}$	$0.41_{.03}$	$95.17_{.21}$	$2.09_{.16}$	$1.64_{.31}$
+TNA(comp.)	$96.39_{.12}$	$0.71_{.14}$	$0.53_{.06}$	$92.54_{.19}$	$0.79_{.09}$	$0.76_{.19}$	$95.13_{.18}$	$0.65_{.11}$	$0.45_{.11}$	$95.08_{.16}$	$1.96_{.1}$	$1.58_{.13}$
ETS	$96.37_{.15}$	$0.84_{.06}$	$0.79_{.19}$	$92.51_{.26}$	$0.85_{.19}$	$0.67_{.24}$	$95.28_{.15}$	$0.67_{.1}$	$0.5_{.12}$	$95.16_{.22}$	$2.14_{.13}$	$1.63_{.3}$
+TNA(sparse)	$96.37_{.16}$	$0.78_{.14}$	$0.72_{.17}$	$92.55_{.25}$	$0.83_{.09}$	$0.71_{.26}$	$95.28_{.11}$	$0.69_{.08}$	$0.42_{.0}$	$95.17_{.21}$	$2.09_{.16}$	$1.64_{.31}$
+TNA(comp.)	$96.39_{.12}$	$0.71_{.14}$	$0.53_{.06}$	$92.56_{.2}$	$0.79_{.1}$	$0.82_{.25}$	$94.66_{.18}$	$0.65_{.11}$	$0.45_{.11}$	$95.08_{.16}$	$1.96_{.1}$	$1.58_{.13}$
AAR	$96.33_{.2}$	$0.75_{.08}$	$0.45_{.09}$	$92.38_{.32}$	$0.91_{.07}$	$e0.67_{.1}$	$95.17_{.08}$	$0.59_{.1}$	$0.45_{.05}$	$95.14_{.24}$	$1.04_{.12}$	$0.95_{.19}$
+TNA(sparse)	$96.35_{.19}$	$0.72_{.05}$	$0.45_{.08}$	$92.37_{.33}$	$0.83_{.14}$	$\mathbf{0.64}_{.12}$	$95.3_{.13}$	$0.59_{.18}$	$\mathbf{0.34}_{.13}$	$95.12_{.26}$	$0.98_{.17}$	$0.95_{.22}$
+TNA(comp.)	$96.41_{.18}$	$0.72_{.23}$	$\mathbf{0.44}_{.19}$	$92.37_{.33}$	$0.83_{.14}$	$\mathbf{0.64}_{.12}$	$94.99_{.1}$	$\mathbf{0.56}_{.13}$	$\mathbf{0.34}_{.08}$	$95.12_{.22}$	$\mathbf{0.9}_{.01}$	$0.95_{.2}$

	CIFAR100-WideResNet28x10			CIFAR100-MobileNetV2			CIFAR100-PreResNet110			CIFAR100-GoogleNet		
Methods	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**
None	$80.42_{.47}$	$5.91_{.3}$	$5.77_{.4}$	$72.79_{.14}$	$10.03_{.19}$	$9.97_{.18}$	$77.69_{.35}$	$10.62_{.33}$	$10.58_{.34}$	$79.42_{.26}$	$6.45_{.2}$	$6.3_{.19}$
+TNA	$80.17_{.53}$	$4.11_{.19}$	$3.92_{.15}$	$72.71_{.13}$	$1.54_{.09}$	$1.52_{.19}$	$77.15_{.34}$	$2.97_{.25}$	$2.84_{.29}$	$79.36_{.25}$	$3.86_{.2}$	$3.86_{.16}$
IROvA	$80.69_{.36}$	$4.71_{.14}$	$4.27_{.15}$	$72.15_{.22}$	$4.81_{.28}$	$4.79_{.19}$	$76.95_{.25}$	$3.7_{.17}$	$4.7_{.49}$	$78.92_{.27}$	$3.73_{.35}$	$4.68_{.17}$
+TNA(sparse)	$80.4_{.43}$	$4.15_{.26}$	$3.36_{.35}$	$72.01_{.26}$	$4.72_{.29}$	$4.17_{.25}$	$76.51_{.22}$	$3.91_{.2}$	$3.65_{.61}$	$78.88_{.35}$	$3.66_{.35}$	$\mathbf{3.23}_{.17}$
+TNA(comp.)	$81.11_{.3}$	$3.89_{.11}$	$3.35_{.14}$	$72.45_{.36}$	$4.32_{.14}$	$3.91_{.38}$	$76.93_{.3}$	$3.66_{.49}$	$3.6_{.47}$	$78.88_{.35}$	$3.66_{.35}$	$\mathbf{3.23}_{.17}$
TS	$80.42_{.47}$	$4.69_{.22}$	$4.55_{.26}$	$72.79_{.14}$	$1.82_{.33}$	$1.78_{.11}$	$77.69_{.35}$	$3.21_{.17}$	$3.13_{.17}$	$79.42_{.26}$	$4.27_{.12}$	$4.28_{.15}$
+TNA(sparse)	$80.17_{.23}$	$4.55_{.24}$	$4.28_{.14}$	$72.71_{.13}$	$1.41_{.15}$	$1.34_{.11}$	$77.15_{.34}$	$3.16_{.26}$	$3.03_{.35}$	$79.36_{.25}$	$4.23_{.14}$	$4.13_{.18}$
+TNA(comp.)	$80.17_{.53}$	$4.55_{.24}$	$4.28_{.14}$	$73.13_{.2}$	$1.41_{.33}$	$1.24_{.33}$	$75.64_{.64}$	$3.01_{.15}$	$2.97_{.14}$	$79.36_{.25}$	$4.23_{.14}$	$4.13_{.18}$
ETS	$80.42_{.47}$	$3.47_{.28}$	$3.88_{.21}$	$72.79_{.14}$	$1.63_{.39}$	$1.56_{.12}$	$77.97_{.44}$	$2.38_{.37}$	$2.6_{.33}$	$79.42_{.26}$	$3.25_{.19}$	$3.4_{.22}$
+TNA(sparse)	$80.17_{.53}$	$3.35_{.26}$	$3.89_{.22}$	$72.71_{.13}$	$\mathbf{1.22}_{.18}$	$1.32_{.03}$	$77.15_{.34}$	$2.19_{.28}$	$2.49_{.35}$	$79.36_{.25}$	$\mathbf{2.86}_{.2}$	$3.35_{.26}$
+TNA(comp.)	$80.76_{.27}$	$\mathbf{3.06}_{.16}$	$3.52_{.17}$	$73.13_{.2}$	$1.27_{.32}$	$\mathbf{1.18}_{.29}$	$75.74_{.72}$	$\mathbf{2.13}_{.27}$	$\mathbf{2.4}_{.37}$	$79.36_{.25}$	$\mathbf{2.86}_{.2}$	$3.35_{.26}$
AAR	$81.06_{.14}$	$3.4_{.38}$	$3.19_{.33}$	$72.53_{.17}$	$2.06_{.4}$	$1.82_{.44}$	$77.37_{.37}$	$3.27_{.12}$	$3.23_{.21}$	$79.28_{.3}$	$4.46_{.31}$	$4.28_{.3}$
+TNA(sparse)	$80.98_{.22}$	$3.33_{.4}$	$\mathbf{3.15}_{.33}$	$72.39_{.26}$	$1.95_{.39}$	$1.85_{.33}$	$77.15_{.47}$	$3.36_{.22}$	$3.07_{.18}$	$79.38_{.29}$	$4.45_{.23}$	$4.34_{.28}$
+TNA(comp.)	$80.98_{.22}$	$3.33_{.4}$	$\mathbf{3.15}_{.33}$	$73.09_{.26}$	$1.52_{.17}$	$1.47_{.27}$	$76.83_{.42}$	$3.15_{.41}$	$2.87_{.43}$	$78.91_{.18}$	$4.45_{.05}$	$4.27_{.25}$

	ImageNet-ResNet50			ImageNet-DenseNet169			ImageNet-ViT-L/16			ImageNet-ViT-H/14		
Methods	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**	**Acc. (↑)**	**ECE (↓)**	**AdaECE (↓)**
None	$76.17_{.05}$	$3.83_{.09}$	$3.73_{.1}$	$75.63_{.11}$	$5.43_{.08}$	$5.43_{.08}$	$84.35_{.06}$	$1.81_{.07}$	$1.8_{.09}$	$85.59_{.05}$	$1.87_{.07}$	$1.77_{.07}$
+TNA	$76.11_{.06}$	$1.73_{.06}$	$1.76_{.08}$	$75.46_{.11}$	$1.63_{.14}$	$1.66_{.14}$	$84.34_{.07}$	$1.11_{.05}$	$1.11_{.05}$	$85.5_{.03}$	$1.05_{.01}$	$1.06_{.03}$
IROvA	$74.96_{.07}$	$6.23_{.37}$	$5.63_{.29}$	$74.82_{.13}$	$6.19_{.77}$	$5.72_{.54}$	$83.5_{.1}$	$5.35_{.29}$	$4.27_{.14}$	$84.75_{.03}$	$5.3_{.16}$	$4.31_{.3}$
+TNA(sparse)	$74.87_{.05}$	$6.24_{.19}$	$4.58_{.33}$	$74.49_{.23}$	$6.1_{.72}$	$4.71_{.67}$	$83.41_{.11}$	$5.32_{.29}$	$4.05_{.12}$	$84.54_{.0}$	$5.24_{.2}$	$4.0_{.24}$
+TNA(comp.)	$74.98_{.03}$	$6.03_{.11}$	$4.55_{.21}$	$74.41_{.13}$	$6.05_{.61}$	$4.7_{.56}$	$83.72_{.08}$	$5.16_{.11}$	$4.05_{.01}$	$84.43_{.13}$	$5.17_{.11}$	$3.9_{.16}$
TS	$76.17_{.05}$	$2.09_{.3}$	$2.02_{.23}$	$75.63_{.11}$	$1.85_{.1}$	$1.83_{.07}$	$84.35_{.06}$	$1.35_{.08}$	$1.35_{.08}$	$85.59_{.05}$	$1.33_{.18}$	$1.32_{.19}$
+TNA(sparse)	$76.11_{.06}$	$2.02_{.27}$	$2.0_{.25}$	$75.46_{.11}$	$1.78_{.05}$	$1.77_{.05}$	$84.34_{.07}$	$1.31_{.12}$	$1.31_{.08}$	$85.5_{.03}$	$1.34_{.16}$	$1.28_{.18}$
+TNA(comp.)	$76.08_{.26}$	$1.93_{.11}$	$1.89_{.04}$	$75.45_{.16}$	$1.78_{.13}$	$1.78_{.07}$	$84.48_{.13}$	$1.25_{.04}$	$1.3_{.16}$	$85.48_{.15}$	$1.31_{.11}$	$1.23_{.01}$
ETS	$76.17_{.05}$	$1.1_{.03}$	$1.26_{.1}$	$75.63_{.11}$	$0.88_{.11}$	$1.06_{.19}$	$84.35_{.06}$	$0.95_{.06}$	$1.14_{.06}$	$85.59_{.05}$	$0.63_{.13}$	$0.91_{.13}$
+TNA(sparse)	$76.11_{.06}$	$1.06_{.05}$	$1.22_{.06}$	$75.46_{.11}$	$0.79_{.13}$	$1.02_{.12}$	$84.34_{.07}$	$\mathbf{0.9}_{.05}$	$\mathbf{1.09}_{.06}$	$85.5_{.03}$	$\mathbf{0.6}_{.14}$	$0.85_{.14}$
+TNA(comp.)	$76.04_{.11}$	$\mathbf{1.01}_{.15}$	$\mathbf{1.19}_{.03}$	$75.45_{.23}$	$\mathbf{0.78}_{.11}$	$\mathbf{1.01}_{.14}$	$84.5_{.11}$	$\mathbf{0.9}_{.17}$	$1.1_{.11}$	$85.43_{.12}$	$\mathbf{0.6}_{.05}$	$\mathbf{0.84}_{.07}$
AAR	$75.55_{.05}$	$2.53_{.17}$	$2.5_{.17}$	$75.37_{.09}$	$2.2_{.22}$	$2.2_{.28}$	$83.93_{.08}$	$2.43_{.15}$	$2.4_{.14}$	$84.96_{.13}$	$2.36_{.28}$	$2.2_{.37}$
+TNA(sparse)	$75.48_{.04}$	$2.52_{.2}$	$2.49_{.2}$	$75.23_{.15}$	$2.33_{.2}$	$2.22_{.21}$	$83.93_{.07}$	$2.41_{.15}$	$2.38_{.14}$	$84.9_{.14}$	$2.3_{.34}$	$2.25_{.3}$
+TNA(comp.)	$75.36_{.23}$	$2.40_{.21}$	$2.39_{.11}$	$75.17_{.18}$	$2.28_{.11}$	$2.25_{.07}$	$84.2_{.07}$	$2.38_{.04}$	$2.38_{.11}$	$84.81_{.12}$	$2.32_{.03}$	$2.28_{.05}$

Discussion & Ablation Study

- Geometrical Point : Temperature Scaling(TS)[Guo et al-17’] of adjusting T can be interpreted as tuning **Magnitude.**

  To our knowledge, our paper is the first to provide the method to tune the **Angle.**

$$\text{Confidence} = \max_{\mathrm{j}} \frac{e^{s_j}}{\Sigma e^{s_j}} \fallingdotseq \frac{e^{\boldsymbol{w}_j \cdot z}}{\Sigma e^{\boldsymbol{w}_j \cdot z}}$$

$$\text{TS Confidence} = \max_{\mathrm{j}} \frac{e^{s_j/T}}{\Sigma e^{s_j/T}} \fallingdotseq \frac{e^{\frac{\boldsymbol{w}_j}{T} \cdot z}}{\Sigma e^{\frac{\boldsymbol{w}_j}{T} \cdot z}}$$

Discussion & Ablation Study

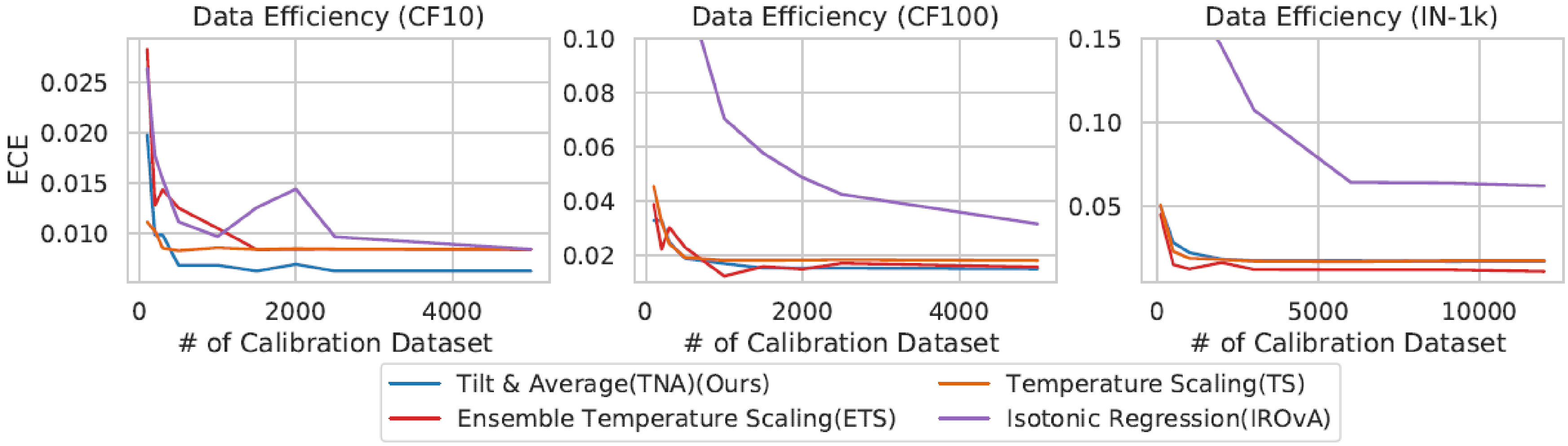
- Geometrical Point : Temperature Scaling(TS)[Guo et al-17'] of adjusting T can be interpreted as tuning **Magnitude.**

  To our knowledge, our paper is the first to provide the method to tune the **Angle.**

  $$\text{Confidence} = \max_{\mathrm{j}} \frac{e^{s_j}}{\Sigma e^{s_j}} \fallingdotseq \frac{e^{w_j \cdot z}}{\Sigma e^{w_j \cdot z}}$$

  $$\text{TS Confidence} = \max_{\mathrm{j}} \frac{e^{s_j/T}}{\Sigma e^{s_j/T}} \fallingdotseq \frac{e^{\frac{w_j}{T} \cdot z}}{\Sigma e^{\frac{w_j}{T} \cdot z}}$$

- Data Efficiency : TNA is **as data efficient as** Temperature Scaling[Guo et al-17'], since the method optimizes a single parameter $\theta(mRC)$, similar to that of TS, in which we also optimize a single parameter T(temperature).



Thank you!

Please Check our paper for the details :