Improving Normalization with the James-Stein Estimator

Seyedalireza Khoshsirat Chandra Kambhamettu
Video/Image Modeling and Synthesis (VIMS) Lab, University of Delaware
{alireza, chandrak}@udel.edu

Abstract

Stein's paradox holds considerable sway in high-dimensional statistics, highlighting that the sample mean, traditionally considered the de facto estimator, might not be the most efficacious in higher dimensions. To address this, the James-Stein estimator proposes an enhancement by steering the sample means toward a more centralized mean vector. In this paper, first, we establish that normalization layers in deep learning use inadmissible estimators for mean and variance. Next, we introduce a novel method to employ the James-Stein estimator to improve the estimation of mean and variance within normalization layers. We evaluate our method on different computer vision tasks: image classification, semantic segmentation, and 3D object classification. Through these evaluations, it is evident that our improved normalization layers consistently yield superior accuracy across all tasks without extra computational burden. Moreover, recognizing that a plethora of shrinkage estimators surpass the traditional estimator in performance, we study two other prominent shrinkage estimators: Ridge and LASSO. Additionally, we provide visual representations to intuitively demonstrate the impact of shrinkage on the estimated layer statistics. Finally, we study the effect of regularization and batch size on our modified batch normalization. The studies show that our method is less sensitive to batch size and regularization, improving accuracy under various setups.

1. Introduction

Deep neural networks have an influential role in many applications, especially computer vision. One milestone improvement in these networks was the addition of normalization layers [21, 29, 34]. Since then, a plethora of research has tried to improve normalization layers through various means, including better estimating the layer statistics. In this paper, we take a statistical approach to estimate layer statistics and introduce an improved way of estimating the mean and variance in normalization layers. The rest of this section is devoted to introducing prerequisite statistical concepts and normalization layers.

Estimators. An estimator refers to a method used to calculate an approximation of a specific value from the data observed. Therefore, one should differentiate between the estimator itself, the targeted value of interest, and the resulting approximation [37].

Shrinkage. In statistics, shrinkage is the reduction of the effects of sampling noise. In regression analysis, a fitted relationship appears to perform worse on a new dataset than on the dataset used for fitting [11, 56]. More specifically, the value of the coefficient of determination 'shrinks.'

Shrinkage Estimators. A shrinkage estimator is an estimator that explicitly or implicitly uses the effects of shrinkage. In loose terms, a naive or basic estimate is improved by combining it with other information. This term refers to the notion that the improved estimate is pushed closer to the value provided by the 'other information' than the basic estimate [24, 56]. In this sense, shrinkage is used to regularize the estimation process. In terms of mean squared error (MSE), many standard estimators can be improved by shrinking them towards zero or some other value. In other words, the improvement in the estimate from the corresponding reduction in the width of the confidence interval is likely to outweigh the worsening of the estimate instructed by biasing the estimate towards zero [23].

Admissibility. Assume x is distributed according to $p(x|\theta)$, with θ being a member of the set Θ. Consider $\hat{\theta}$ as an estimator for θ and let $R(\hat{\theta}, \theta)$ represent the risk of using $\hat{\theta}$, which is calculated based on a specific loss function. The risk is defined by $R(\hat{\theta}, \theta) = \mathbb{E}[\ell(\hat{\theta}, \theta)]$, where ℓ signifies the loss function and the expectation is taken over x drawn from $p(x|\theta)$, with $\hat{\theta}$ being a derived function of x. An estimator is deemed 'inadmissible' if there is another estimator, $\tilde{\theta}$, which performs better; that is, $R(\tilde{\theta}, \theta) \leq R(\hat{\theta}, \theta)$ for all θ within Θ, and there's at least one θ for which the inequality is strict. If no such dominating estimator exists, then $\hat{\theta}$ is considered 'admissible' [6].

James-Stein. Calculating the mean of a multivariate normal distribution stands as a key issue in the field of statistics. Typically, the sample mean is used, which also happens to be the maximum-likelihood estimator. However, the

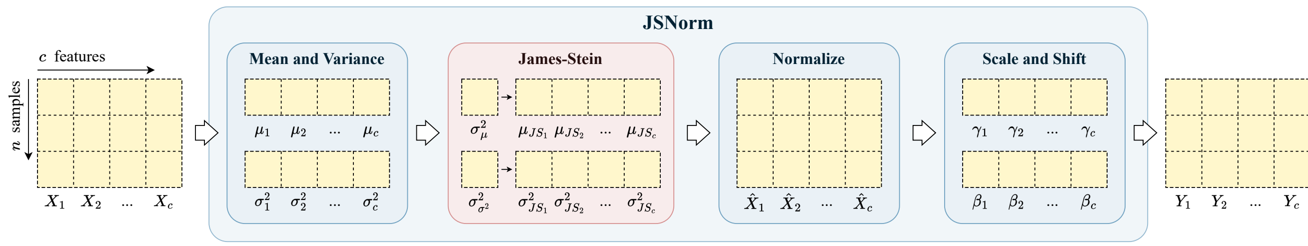

Figure 1. The overall structure of our proposed JSNorm for batch normalization with the key components involved at each step. JSNorm integrates the James-Stein estimator into normalization layers to refine the originally estimated statistics. This design choice ensures that the computational overhead associated with JSNorm is minimal and practically negligible.

James-Stein (JS) estimator, which is known to be biased, is used for estimating the mean of c correlated Gaussian distributed random vectors whose means are not known. The development of this estimator unfolded over two major papers; the initial version was introduced by Charles Stein in 1956 [ ], leading to the surprising discovery that the standard mean estimate is admissible when $c \leq 2$, but becomes inadmissible for $c \geq 3$. This work suggested an enhancement that involves shrinking the sample means towards a central vector of means, a concept often cited as Stein's example or paradox [57].

Normalization Layers. The primary objective of normalization techniques is to enhance training stability and facilitate the design of network architectures. While various normalization layers are available [19, 35, 43, 51, 58], this paper specifically focuses on batch normalization and layer normalization. These two normalization techniques are widely utilized in computer vision networks and serve as the most common choices within the field.

Batch Normalization (BatchNorm or BN) [21] is a key deep learning technique that enhances computer vision applications by normalizing feature-maps using mean and variance computed over batches. This aids optimization [ ], promotes convergence in deep networks [ ], and reduces the number of iterations to converge, improving performance [ ].

Layer Normalization (LayerNorm or LN) [ ] is a different approach that addresses training issues in Recurrent Neural Networks [ ]. It normalizes statistics within a single sample, making it independent of batch size, and was specifically designed to work with the variable statistics of recurrent neurons. It is a critical component in transformer networks [33, 53].

Contributions. Our contributions in this study are threefold:

- **Identification of Inadmissible Estimators:** We establish that widely-used normalization layers in deep learning networks employ inadmissible estimators for calculating mean and variance. This insight draws attention to potential limitations in the current practice.
- **Introduction of Improved Normalization Methods:** We present an innovative approach wherein the James-Stein estimator is adapted to more accurately estimate the mean and variance in normalization layers. This is a significant contribution, as it bridges the gap between classical statistical methods and contemporary deep learning techniques.
- **Empirical Validation Across Domains:** Through extensive experimentation, we substantiate the efficacy of our proposed method across three distinct computer vision tasks. Additionally, we conduct a series of analyses to ascertain the robustness and performance enhancements attributed to our approach under varying configurations.

2. Related Work

The James-Stein estimator has motivated a rich literature on the theme of "shrinkage" in statistics. Just a small selection of examples include LASSO [ ], ridge regression [ ], the Ledoit-Wolf covariance estimator [ ] and Elastic Net [ ]. Excellent textbook treatments of the concepts behind Stein's paradox and James-Stein shrinkage include [12, 14].

Before batch normalization became prevalent, various types of normalization layers were already being utilized in deep neural networks. Local Response Normalization (LRN) featured in AlexNet [29] and was adopted by subsequent models [17, 28, 45, 65]. LRN normalizes the values around a given pixel within a specified neighborhood.

Batch Normalization (BN) [21], introduced later, applies a more comprehensive form of normalization across the entire batch of data and suggests applying this technique across all layers of the network. In the wake of BN, other normalization approaches [2, 25, 27, 51] were developed that do not rely on the batch dimension. For example, Layer Normalization (LN) [2] adjusts the data across the channel dimension, while Instance Normalization (IN) [51] carries out a batch normalization-like process for each individual instance. In a different approach from normalizing

Batch Normalization	Layer Normalization	Description
$\mu_{\mathcal{B}} = \frac{1}{n \times h \times w} \sum_{n,h,w} x_{i,j,k}$	$\mu_{\mathcal{B}} = \frac{1}{h \times w} \sum_{h,w} x_{i,j}$	Calculating the corresponding mean
$\sigma^2_{\mathcal{B}} = \frac{1}{n \times h \times w} \sum_{n,h,w} (x_{i,j,k} - \mu_{\mathcal{B}})^2$	$\sigma^2_{\mathcal{B}} = \frac{1}{h \times w} \sum_{h,w} (x_{i,j} - \mu_{\mathcal{B}})^2$	Calculating the corresponding variance
$\mu_{\mu_{\mathcal{B}}} = \frac{1}{c} \sum_c \mu_{\mathcal{B}i}$	$\mu_{\mu_{\mathcal{B}}} = \frac{1}{c} \sum_c \mu_{\mathcal{B}i}$	Mean of the estimated means
$\sigma^2_{\mu_{\mathcal{B}}} = \frac{1}{c} \sum_c (\mu_{\mathcal{B}i} - \mu_{\mu_{\mathcal{B}}})^2$	$\sigma^2_{\mu_{\mathcal{B}}} = \frac{1}{c} \sum_c (\mu_{\mathcal{B}i} - \mu_{\mu_{\mathcal{B}}})^2$	Variance of the estimated means
$\mu_{\mathcal{JS}} = \left(1 - \frac{(c-2)\sigma^2_{\mu_{\mathcal{B}}}}{\|\mu_{\mathcal{B}}\|_2^2}\right) \mu_{\mathcal{B}}$	$\mu_{\mathcal{JS}} = \left(1 - \frac{(c-2)\sigma^2_{\mu_{\mathcal{B}}}}{\|\mu_{\mathcal{B}}\|_2^2}\right) \mu_{\mathcal{B}}$	JS estimation of the mean
$\mu_{\sigma^2_{\mathcal{B}}} = \frac{1}{c} \sum_c \sigma^2_{\mathcal{B}i}$	$\mu_{\sigma^2_{\mathcal{B}}} = \frac{1}{c} \sum_c \sigma^2_{\mathcal{B}i}$	Mean of the estimated variances
$\sigma^2_{\sigma^2_{\mathcal{B}}} = \frac{1}{c} \sum_c (\sigma^2_{\mathcal{B}i} - \mu_{\sigma^2_{\mathcal{B}}})^2$	$\sigma^2_{\sigma^2_{\mathcal{B}}} = \frac{1}{c} \sum_c (\sigma^2_{\mathcal{B}i} - \mu_{\sigma^2_{\mathcal{B}}})^2$	Variance of the estimated variances
$\sigma^2_{\mathcal{JS}} = \left(1 - \frac{(c-2)\sigma^2_{\sigma^2_{\mathcal{B}}}}{\|\sigma^2_{\mathcal{B}}\|_2^2}\right) \sigma^2_{\mathcal{B}}$	$\sigma^2_{\mathcal{JS}} = \left(1 - \frac{(c-2)\sigma^2_{\sigma^2_{\mathcal{B}}}}{\|\sigma^2_{\mathcal{B}}\|_2^2}\right) \sigma^2_{\mathcal{B}}$	JS estimation of the variance
$\hat{x} = \frac{x - \mu_{\mathcal{JS}}}{\sqrt{\sigma^2_{\mathcal{JS}} + \epsilon}}$	$\hat{x} = \frac{x - \mu_{\mathcal{JS}}}{\sqrt{\sigma^2_{\mathcal{JS}} + \epsilon}}$	Standardization using the JS estimations
$y = \gamma \hat{x} + \beta$	$y = \gamma \hat{x} + \beta$	Scale and shift

Table 1. Our proposed batch normalization and layer normalization which integrate the James-Stein estimator. Given an input feature-map $x \in \mathbb{R}^{n \times c \times h \times w}$, for batch normalization, each channel (c) is processed separately, and for layer normalization, each sample (n). The operations are done in order, from top to bottom.

data, Weight Normalization [43] focuses on normalizing the weights of the filters within the network. Group Normalization (GN) [58] generalizes LN, dividing the neurons into groups and standardizing the layer input within the neurons of each group for each sample independently.

A diverse set of works aiming at improving normalization layers exist. Decorrelated Batch Normalization [20] employs whitening techniques to solve the so-called "stochastic axis swapping" problem. In [60], a simple version of the layer normalization is introduced that removes the bias and gain parameters to cope with over-fitting. Batch group normalization [66] extends the grouping mechanism of GN from being over only channels dimension to being over both channels and batch dimensions. MoBN [43] is a mean-only batch normalization that performs centering only along the batch dimension. It performs well when combined with weight normalization. A scaling-only batch normalization is proposed in [62] that performs well in training with a small batch size.

Although the James-Stein estimator has not directly been used in deep learning, some methods use this estimator alongside deep learning. In [1], the James-Stein estimator is used for feature extraction before feeding data to a deep neural network. This method combines Pinsker's theorem with James-Stein to leverage the advantages of non-parametric regression to use deep learning in limited data setups. C-SURE [7] is a novel shrinkage estimator based on the James-Stein estimator for the complex-valued manifold. It is incorporated in a complex-valued classifier network for estimating the statistical mean of the per-class wFM features.

Our research stands at the forefront by being the first study to investigate the admissibility of estimators within normalization layers in deep learning. Furthermore, we break new ground by introducing an innovative methodology to address this issue, thereby pioneering a potentially transformative approach in the field.

3. Method

In this section, we begin by providing a concise overview of batch normalization [21] and layer normalization [2] techniques. We then delve into our underlying motivation for proposing modified versions of these normalization methods. Finally, we present our novel approaches to normalization.

Batch Normalization. The batch normalization layer performs feature standardization within each batch by normalizing each feature individually, followed by learning a shared weight and bias for the entire batch. Formally, given the input to a batch normalization layer $\mathbf{X} \in \mathbb{R}^{n \times c}$,

Method	Original	JSNorm
ResNet-18 [15]	69.7	**71.0**±0.3 (+1.3)
ResNet-50 [15]	76.1	**77.2**±0.3 (+1.1)
ResNet-152 [15]	77.9	**78.9**±0.2 (+1.0)
EfficientNet-B1 [48]	79.2	**80.2**±0.2 (+1.0)
EfficientNet-B3 [48]	81.7	**82.6**±0.1 (+0.9)
EfficientNet-B5 [48]	83.7	**84.6**±0.1 (+0.9)
GENet-light [31]	75.7	**77.2**±0.3 (+1.5)
SwinV2-T, win8x8 [32]	81.8	**82.8**±0.2 (+1.0)
SwinV2-T, win16x16 [32]	82.8	**83.7**±0.2 (+0.9)
SwinV2-S, win8x8 [32]	83.7	**84.6**±0.1 (+0.9)

Table 2. Comparison of top-1 accuracy on ImageNet with mean±std in five random evaluations. Notably, all the networks were trained without any fine-tuning of hyper-parameters.

where n denotes the batch size, and c is the feature dimension, batch normalization layer first normalizes each feature $i \in \{1 \dots c\}$:

$$\hat{x}_i = \frac{x_i - \mathbb{E}[x_i]}{\sqrt{Var[x_i]}}, \tag{1}$$

where the expectation and variance are computed over the training data set. Next, for each feature dimension, a pair of parameters γ_i and β_i are used to scale and shift the normalized value:

$$y_i = \gamma_i \hat{x}_i + \beta_i. \tag{2}$$

These parameters are learned during the training.

Layer Normalization. Similar to batch normalization, layer normalization begins by standardizing the input. However, unlike batch normalization, layer normalization operates independently on each sample, without relying on the batch size. As a result, the standardization process occurs individually for each sample.

Therefore, for each sample $i \in \{1 \dots n\}$:

$$\hat{x}_i = \frac{x_i - \mu_{x_i}}{\sqrt{Var[x_i]}}. \tag{3}$$

Similar to batch normalization, it is followed by learning a pair of parameters γ_i and β_i:

$$y_i = \gamma_i \hat{x}_i + \beta_i. \tag{4}$$

Motivation. One notable concern pertaining to Equations 1 and 3 lies in the estimation of the mean and variance. The conventional approach suggests independently calculating the mean and variance using "usual estimators". For batch normalization:

$$\mathbb{E}[x_i] = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}, \tag{5}$$

$$Var[x_i] = \frac{1}{n}\sum_{j=1}^{n} (x_{i,j} - \mathbb{E}[x_i])^2, \tag{6}$$

Method	Original	JSNorm
HRNetV2 [54]	81.6	**83.0**±0.4 (+1.4)
HRNetV2+OCR [64]	83.0	**84.2**±0.2 (+1.2)
HRNetV2+OCR† [64]	84.2	**85.3**±0.1 (+1.1)
EfficientPS† [36]	84.2	**85.3**±0.2 (+1.1)
Lawin [61]	84.4	**85.4**±0.1 (+1.0)

Table 3. Evaluation of mIoU scores on the Cityscapes test set. We use † to mark methods pre-trained on Mapillary Vistas dataset [38].

and for layer normalization:

$$\mu_{x_i} = \frac{1}{c}\sum_{j=1}^{c} x_{i,j}, \tag{7}$$

$$Var[x_i] = \frac{1}{c}\sum_{j=1}^{c} (x_{i,j} - \mu_{x_i})^2, \tag{8}$$

are the estimators.

Given that all the features contribute to a shared loss function, according to Stein's paradox [47], these estimators are inadmissible when $c \geq 3$. Notably, in computer vision networks, it is consistently observed that $c \geq 3$. To address this, we propose a novel method to adopt admissible shrinkage estimators, which effectively enhance the estimation of the mean and variance in both normalization layers.

James-Stein. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_c\}$ with unknown means $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_c\}$ and estimates $\hat{\boldsymbol{\theta}} = \{\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_c\}$. The basic formula for the James-Stein estimator is:

$$\hat{\boldsymbol{\theta}}_{JS} = \hat{\boldsymbol{\theta}} + s(\mu_{\hat{\boldsymbol{\theta}}} - \hat{\boldsymbol{\theta}}), \tag{9}$$

where $\mu_{\hat{\boldsymbol{\theta}}} - \hat{\boldsymbol{\theta}}$ is the difference between the total mean (average of averages) and each individual estimated mean, and s is a shrinking factor. Among the numerous perspectives that motivate the James-Stein estimator, the empirical Bayes perspective [10] is exquisite. Taking a Gaussian prior on the unknown means leads us to the following formula [22, 23]:

$$\hat{\boldsymbol{\theta}}_{JS} = \left(1 - \frac{(c-2)\sigma^2}{\|\hat{\boldsymbol{\theta}} - \mathbf{v}\|_2^2}\right)(\hat{\boldsymbol{\theta}} - \mathbf{v}) + \mathbf{v}, \tag{10}$$

where $\|\cdot\|_2$ denotes the L_2 norm of the argument, σ^2 is the variance, $\mathbf{v}$ is an arbitrary fixed vector that shows the shrinkage direction, and $c \geq 3$. Setting $\mathbf{v} = \mathbf{0}$ results the following:

$$\hat{\boldsymbol{\theta}}_{JS} = \left(1 - \frac{(c-2)\sigma^2}{\|\hat{\boldsymbol{\theta}}\|_2^2}\right)\hat{\boldsymbol{\theta}}. \tag{11}$$

The above estimator shrinks the estimates towards the origin $\mathbf{0}$.

Method	ScanObjectNN Original	ScanObjectNN JSNorm	ModelNet40 Original	ModelNet40 JSNorm
PointNet [39]	68.2	**70.0**±0.4 (+1.8)	89.2	**90.8**±0.3 (+1.6)
PointNet++ [40]	77.9	**79.5**±0.3 (+1.6)	91.9	**93.5**±0.4 (+1.6)
DGCNN [55]	81.9	**83.3**±0.1 (+1.4)	93.5	**94.8**±0.2 (+1.3)
Point-BERT [63]	83.1	**84.3**±0.2 (+1.2)	93.8	**94.8**±0.1 (+1.0)
PointNeXt-S [41]	87.7	**88.8**±0.1 (+1.1)	93.2	**94.2**±0.1 (+1.0)

Table 4. Evaluation of classification accuracy across two 3D datasets: ScanObjectNN and ModelNet40.

Commonly in transformer networks for computer vision [32, 33, 53], the statistics for layer normalization are calculated on the patch size dimensions h, and w. We take advantage of this design and use the James-Stein estimator such that our layer normalization stays independent of batch size, and each sample is processed independently.

In this paper, we employ Equation 11 to avoid the 'mean shift' problem [4, 5]. To integrate this equation into normalization layers, we substitute the $\hat{\boldsymbol{\theta}}$ in Equation 11 with the estimated mean and variance obtained through the original method. By applying the James-Stein estimator to the estimated statistics from the original method, the additional processing required is minimal and can be considered negligible. Consequently, we utilize the James-Stein estimator for both the mean and variance in the normalization layers. For batch normalization, $\mathbb{E}[x]$ and $Var[x]$ are vectors of length c (for the whole batch). Therefore, they can directly be used in place of $\hat{\boldsymbol{\theta}}$. For layer normalization, μ_x and $Var[x]$ are in the form of $\mathbb{R}^{n\times c}$, and since in layer normalization each sample should be processed independently, each vector from the second dimension c is separately used in place of $\hat{\boldsymbol{\theta}}$. Table 1 shows the detailed definition of our proposed normalization layers, while Figure 1 depicts the overview of JSNorm for batch normalization.

3.1. Gaussian Prior

Employing a variant of the James-Stein estimator that assumes a Gaussian prior might raise queries regarding the general applicability of our methodology. This concern can be addressed through two key observations:

- **Alignment with Normal Distribution in Normalization Layers:** Normalization layers aim to standardize the features-maps within a layer and make them resemble a standard Gaussian distribution with a mean of zero and a standard deviation of one. Thus, adopting a Gaussian prior aligns with the intrinsic characteristics of the feature-maps and does not substantially deviate their distribution within the network. Moreover, numerous approaches to developing normalization-free networks [4, 5] incorporate Gaussian weight initialization and standardization to promote a Gaussian-like distribution for feature-maps. This underscores the practical advantages of adopting a Gaussian prior.
- **Distinction Between Feature-Maps and Input Distributions:** It is crucial to acknowledge that the distribution of the feature-maps within the neural network need not mirror that of the input data. The succession of transformations applied by the network layers often results in an evolution of the input data distribution. As the network trains, it learns to change data representation in a manner useful for the specific task. Therefore, the efficacy of our method is not strictly tethered to the distribution of inputs.

4. Experiments

We conduct a comprehensive evaluation of our proposed method across various computer vision tasks. For each task, we utilize well-established state-of-the-art networks with readily available implementations. The sole modification we introduce involves replacing the normalization layers with our proposed batch normalization or layer normalization counterparts. As a result, all other hyper-parameters and training configurations remain consistent with those outlined in the original papers.

4.1. Image Classification

Our evaluation of the proposed method for image classification involves using the ImageNet dataset [42], which consists of 1.28 million training images and 50,000 validation images across 1,000 classes. We train the networks from scratch and report the top-1 accuracy achieved. The results are presented and compared in Table 2, where we assess the performance of our modified batch normalization on various model sizes from the ResNet [15], EfficientNet [48], and GENet [31] families, as well as the SwinV2 [32] model with layer normalization.

Across the ResNet models, we observe a maximum accuracy improvement of 1.3% for ResNet-18, while the larger models show slightly lesser improvement. This trend holds true for other network architectures as well. For GENet-light, we achieve a maximum accuracy improvement of 1.5%, and EfficientNet-B5 demonstrates a minimum accuracy improvement of 0.9%. While it is true that the gains are more pronounced for ResNet-18 or GENet-light, it is important to note that our method also yields steady improvements on larger networks, demonstrating its broad applicability. It is generally accepted that larger networks, which already attain higher accuracy levels and reach performance saturation, exhibit less incremental improvement when more regularizers or data augmentation techniques are introduced. This behavior is consistent with that of the JSNorm, which functions as a regularizer and is inherently similar to other regularizers. The smaller improvements observed in larger networks can be attributed to the presence of existing regularizers, which can limit the additional boost provided by introducing another regularizer. This is one of the motivations for performing the Regularization Effect ablation study (Section 5.2).

The findings in this section highlight the compatibility of our proposed JSNorm with both convolutional and transformer networks, indicating its effectiveness in improving model performance.

4.2. Semantic Segmentation

The Cityscapes dataset [8] serves as our primary evaluation dataset for semantic segmentation, consisting of 5,000 high-quality street images with pixel-level annotations. These finely annotated images are divided into subsets of 2,975 for training, 500 for validation, and 1,525 for testing. Additionally, the dataset includes an additional 20,000 coarsely annotated images. It contains 30 classes, with 19 classes used for performance assessment.

Our experiments involve the use of HRNetV2 [54] and its augmented version HRNetV2+OCR [64], as well as EfficientPS [36] and Lawin [61]. HRNetV2 is a fully convolutional network that maintains high-resolution representations throughout the network. EfficientPS employs EfficientNet as a backbone for semantic and panoptic segmentation. Meanwhile, Lawin is a multi-scale transformer network that utilizes a window attention mechanism. In our evaluation, we replace the normalization layers in the aforementioned networks with our proposed normalization layer and compare the results.

Table 3 presents the mean intersection over union (mIoU) measure for these models on the Cityscapes dataset. Our improved batch normalization demonstrates notable enhancements for HRNetV2, achieving an improvement of 1.4%. Additionally, we observe a minimum improvement of 1.0%, which boosts Lawin's accuracy to reach 85.4% on this dataset. These results showcase the efficacy of our proposed normalization approach in improving the segmentation performance across different models.

4.3. 3D Object Classification

A 3D point cloud, which comprises an unordered collection of 3D points, necessitates distinct network architectures compared to those tailored for 2D images. This, in turn, creates an alternative platform for evaluation. We have conducted experiments utilizing two datasets tailored for 3D object classification. The first dataset, ScanObjectNN [52], is derived from real 3D scenes, and its inherent complexity, augmented by occlusions and noise, poses substantial challenges for prevailing 3D classification methodologies. ScanObjectNN encompasses 2,309 training and 581 testing point clouds, distributed across 15 object classes. The second dataset, ModelNet40 [59], is widely recognized in the realm of 3D object classification and comprises synthetic object point clouds. The dataset contains 12,311 CAD-generated meshes categorized into 40 classes, and is partitioned into 9,843 training and 2,468 testing samples. We assessed the efficacy of our proposed technique on five models, four incorporating batch normalization - PointNet [39], PointNet++ [40], DGCNN [55], PointNeXt [41] - and one employing layer normalization, namely Point-BERT [63].

Table 4 presents the classification accuracy for both ScanObjectNN and ModelNet40 datasets. Remarkably, by employing our enhanced JSNorm, PointNet attains a substantial improvement in classification accuracy, increasing by 1.8% on the ScanObjectNN dataset. Even at the lower end of the spectrum, Point-BERT on ModelNet40 exhibits a noteworthy enhancement in accuracy by 1.0%. These findings underscore the versatility of our proposed approach, indicating that its application extends beyond 2D image networks and is adaptable to a diverse array of data formats and network architectures.

Method	Original	Ridge	LASSO
ResNet-18 [15]	69.7	70.1 (+0.4)	70.1 (+0.4)
ResNet-152 [15]	77.9	78.0 (+0.1)	78.1 (+0.2)
EfficientNet-B1 [48]	79.2	79.4 (+0.2)	79.5 (+0.3)
EfficientNet-B5 [48]	83.7	83.7 (+0.0)	83.8 (+0.1)
GENet-light [31]	75.7	76.0 (+0.3)	76.0 (+0.3)
SwinV2-T [32]	81.8	81.9 (+0.1)	81.9 (+0.1)
SwinV2-S [32]	83.7	83.7 (+0.0)	83.7 (+0.0)

Table 5. Comparative analysis of two widely used shrinkage estimators: Ridge and LASSO. The figures represent the Top-1 accuracy on the ImageNet dataset.

5. Extra Studies

In addition to our primary study, we conduct an additional study that compares two alternative shrinkage estimators. Furthermore, we carry out three ablation studies to gain insights into the impact of regularization, shrinkage, and batch size on the performance of our proposed method.

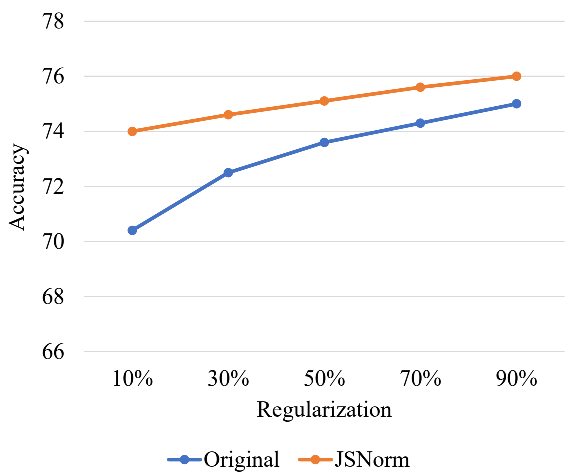

Figure 2. Comparative analysis of our enhanced batch normalization performance across various regularization intensities. JSNorm not only boosts accuracy but also exhibits increased robustness, particularly under lower regularization.

5.1. Ridge and LASSO Estimators

In this study, we assess two widely utilized shrinkage estimators, namely Ridge [16] and LASSO [50], to determine whether alternative shrinkage estimators possess the capacity to enhance accuracy. In terms of the Ridge estimator, $\hat{\boldsymbol{\theta}}$ can be estimated via:

$$\hat{\boldsymbol{\theta}}_{Ridge} = \underset{\hat{\boldsymbol{\theta}}}{\text{argmin}} \left[\ell(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}) + \lambda \|\hat{\boldsymbol{\theta}}\|_2^2 \right], \tag{12}$$

and in terms of LASSO:

$$\hat{\boldsymbol{\theta}}_{LASSO} = \underset{\hat{\boldsymbol{\theta}}}{\text{argmin}} \left[\ell(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}) + \lambda \|\hat{\boldsymbol{\theta}}\|_1 \right], \tag{13}$$

where λ is called the regularization parameter and controls the amount of shrinkage. Both the Ridge and LASSO estimators perform regularization of the estimated parameters, with the LASSO estimator offering the added benefit of variable selection, which enhances the interpretability of the model. Both estimators also introduce shrinkage towards $\mathbf{0}$ as part of their regularization process.

We incorporate the Ridge and LASSO estimators as regularization components acting upon the estimated mean and variance within batch normalization, a methodology that can be seamlessly extended to layer normalization as well. It is imperative to highlight that there is no alteration to the original formulas of batch normalization when employing Ridge and LASSO estimators. To integrate these regularization components, we modify the training loss function in the following manner:

$$\ell_{final} = \ell_{original} + \lambda \sum f(\mu_{\mathcal{B}}) + f(\sigma^2_{\mathcal{B}}), \tag{14}$$

Method	32	64	128	256	512	1024
Original	75.1	75.4	75.6	75.7	75.6	75.5
JSNorm	**76.8**	**77.0**	**77.1**	**77.1**	**77.0**	**76.9**
Δ	+1.7	+1.6	+1.5	+1.4	+1.4	+1.4

Table 6. Impact of varying training batch sizes on ImageNet accuracy. Our JS batch normalization improves accuracy and demonstrates superior robustness across different batch sizes.

where $\sum$ is summation over all batch normalization layers and f is the regularization term, $\|\cdot\|_2^2$ for Ridge and $\|\cdot\|_1$ for LASSO. Since the value of $\ell_{original}$ can be very much larger or smaller than $\sum f(\mu_{\mathcal{B}}) + f(\sigma^2_{\mathcal{B}})$ for different tasks and networks, λ needs to be tuned accordingly. To solve this problem, we re-scale λ proportionally to the value of the regularization part and $\ell_{original}$:

$$\lambda = \lambda_{original}\left(\frac{\ell_{original}}{\sum f(\mu_{\mathcal{B}}) + f(\sigma^2_{\mathcal{B}})}\right). \tag{15}$$

This way, $\lambda_{original}$ is the hyper-parameter that should be chosen. Re-scaling λ happens outside the computational graph to prevent it from affecting gradient calculation.

We subject the regularized normalization layers to evaluation within the context of the image classification task. Table 5 presents the top-1 accuracy on the ImageNet dataset for several distinct networks. While Ridge and LASSO contribute to enhanced performance, they do not match the level of improvement achieved by the James-Stein estimator. Given that the James-Stein estimator is predicated on the assumption of a Gaussian distribution underlying the data, this may account for its superior performance relative to Ridge and LASSO. In our experimental assessments, LASSO marginally outperforms Ridge in terms of accuracy, though their performances are largely analogous. The results underscore the capability of various shrinkage estimators to bolster accuracy, albeit not to the same extent as James-Stein.

5.2. Regularization Effect

In the scholarly domain, it is well-established that shrinkage estimators inherently exhibit regularization effects [16, 50]. In this investigation, we illuminate the performance characteristics of our novel JSNorm layers across an array of regularization magnitudes. For this purpose, we deploy three regularization techniques, namely RandAugment [9], stochastic depth [18], and dropout [46]. We initiate the experiment with a baseline configuration using GENet-light [31] and progressively escalate the regularization factors. Guided by the methodologies delineated in [9] and [49], we establish the upper bounds for regularization factors. The combined regularization is represented in percentage to foster clarity and streamline the visualisation.

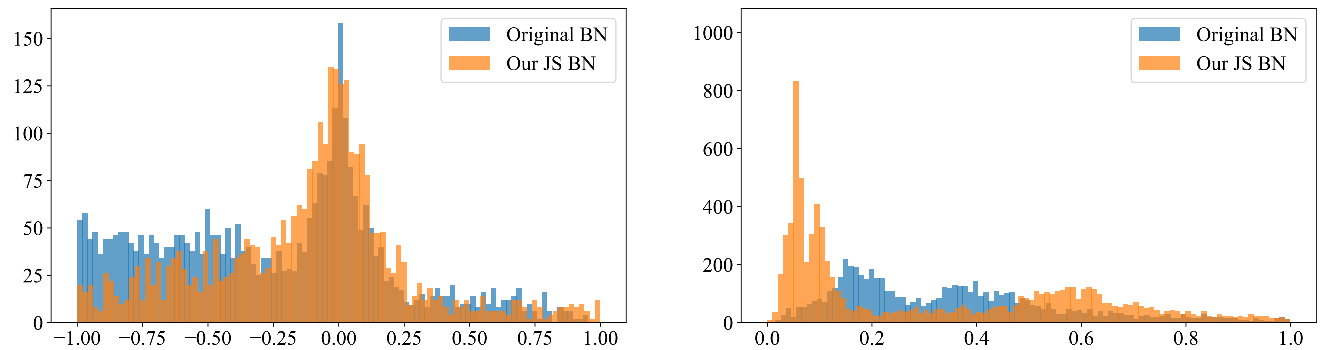

Figure 3. The shrinkage effect of JSNorm on ResNet-18 [15]. The figures illustrate the distribution of running means (left) and running variances (right) in the original batch normalization compared to JSNorm, showcasing the shrinkage effect induced by our JSNorm layer.

We train the network under two scenarios – one employing our enhanced batch normalization and the other without.

Figure 2 presents a graphical representation that compares the two approaches across a range of regularization factors. An intriguing observation is that our method exhibits its most pronounced enhancement in scenarios characterized by low regularization. Nonetheless, even in scenarios with maximal regularization, our method registers noteworthy accuracy improvements. This exemplifies the robustness and stability of our proposition across a spectrum of regularization parameters, thereby outclassing the conventional approach.

5.3. Shrinkage Effect

Shrinkage estimators exert a shrinkage effect on the estimated parameters. We are keen to analyze how this shrinkage influences the normalization layers. To accomplish this, we provide a visual comparison of the distribution of the running statistics within the batch normalization layers of a ResNet-18 model [15]. Two networks are trained, one employing the standard batch normalization and the other utilizing our enhanced batch normalization.

Figure 3 offers a graphical representation of the histograms of all the running means (left subfigure) and variances (right subfigure), both with and without the incorporation of our JSNorm. The subfigures illuminate how our JSNorm layer nudges the distributions toward zero. Pertaining to the running means, the employment of our JSNorm facilitates a distribution that is less skewed and more bell-shaped. As for the running variances, directing the distribution towards zero does not hinder the network's capability to learn layers with elevated variances. In essence, it contributes positively to the diversity of the distribution.

5.4. Batch Size Effect

Batch size plays a pivotal role in the training process of networks that incorporate batch normalization layers. Given that our method enhances the estimation of normalization statistics, it is intriguing to investigate its performance across varying batch sizes. This study is designed to elucidate the behavior of our James-Stein-augmented batch normalization in the context of different batch sizes. To achieve this, we train a GENet-light [31] network on the ImageNet dataset using a range of batch sizes, spanning from 32 to 1024. We employ the linear learning rate scaling rule [13] to adjust to the alterations in batch size.

Table 6 presents the findings of this investigation. Our enhanced batch normalization exhibits optimal performance with smaller batch sizes, attributable to the James-Stein estimator's proficiency in estimating normalization statistics with a limited sample pool. Notably, even with larger batch sizes, our method surpasses the performance of the original batch normalization by a considerable margin. Consequently, our JSNorm demonstrates not only an enhancement in accuracy but also exhibits robustness with respect to batch size variations.

6. Conclusion

Through the lens of Stein's paradox, we illustrated that normalization layers employ inadmissible estimators, resulting in suboptimal estimations of layer statistics. To address this issue, we introduced an innovative technique utilizing the James-Stein estimator, which enhances the accuracy of mean and variance estimations. We evaluated our proposed method rigorously across three distinct computer vision tasks. Our findings demonstrated that the technique not only bolstered the accuracy of both convolutional and transformer networks but did so without incurring additional computational overhead. We performed extra studies to unveil that our approach exhibits robustness and is less susceptible to changes in batch size and regularization, leading to consistent improvements in accuracy under diverse configurations.

References

[1] Marko Angjelichinoski, Mohammadreza Soltani, John Choi, Bijan Pesaran, and Vahid Tarokh. Deep pinsker and james-stein neural networks for decoding motor intentions from limited data. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:1058–1067, 2021. 3

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2, 3

[3] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in neural information processing systems*, 31, 2018. 2

[4] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. *arXiv preprint arXiv:2101.08692*, 2021. 5

[5] Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071. PMLR, 2021. 5

[6] Lawrence D Brown. Admissible estimators, recurrent diffusions, and insoluble boundary value problems. *The Annals of Mathematical Statistics*, 42(3):855–903, 1971. 1

[7] Rudrasis Chakraborty, Yifei Xing, Minxuan Duan, and Stella X Yu. C-sure: Shrinkage estimator and prototype classifier for complex-valued deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 80–81, 2020. 3

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 6

[9] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 7

[10] Bradley Efron and Carl Morris. Data analysis using stein's estimator and its generalizations. *Journal of the American Statistical Association*, 70(350):311–319, 1975. 4

[11] Brian S Everitt and Anders Skrondal. The cambridge dictionary of statistics, 2010. 1

[12] Dominique Fourdrinier, William E Strawderman, and Martin T Wells. *Shrinkage estimation*. Springer, 2018. 2

[13] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 8

[14] Marvin HJ Gruber. *Improving efficiency by shrinkage: the James-Stein and ridge regression estimators*. Routledge, 2017. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 5, 6, 8

[16] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. 2, 7

[17] Parisa Hosseini, Seyedalireza Khoshsirat, Mohammad Jalayer, Subasish Das, and Huaguo Zhou. Application of text mining techniques to identify actual wrong-way driving (wwd) crashes in police reports. *International Journal of Transportation Science and Technology*, 2022. 2

[18] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 7

[19] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *arXiv preprint arXiv:2009.12836*, 2020. 2

[20] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 791–800, 2018. 3

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 1, 2, 3

[22] W James and C Stein. Estimation with quadratic loss. volume 1 of proc. fourth berkeley symp. on math. statist. and prob, 1961. 4

[23] William James and Charles Stein. Estimation with quadratic loss. In *Breakthroughs in statistics*, pages 443–460. Springer, 1992. 1, 4

[24] Seyedalireza Khoshsirat and Chandra Kambhamettu. Semantic segmentation using neural ordinary differential equations. In *Advances in Visual Computing: 17th International Symposium, ISVC 2022, San Diego, CA, USA, October 3–5, 2022, Proceedings, Part I*, pages 284–295. Springer, 2022. 1

[25] Seyedalireza Khoshsirat and Chandra Kambhamettu. Embedding attention blocks for the vizwiz answer grounding challenge. *VizWiz Grand Challenge Workshop*, 2023. 2

[26] Seyedalireza Khoshsirat and Chandra Kambhamettu. Empowering visually impaired individuals: A novel use of apple live photos and android motion photos. In *25th Irish Machine Vision and Image Processing Conference*, 2023. 2

[27] Seyedalireza Khoshsirat and Chandra Kambhamettu. Sentence attention blocks for answer grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6080–6090, 2023. 2

[28] Seyedalireza Khoshsirat and Chandra Kambhamettu. A transformer-based neural ode for dense prediction. *Machine Vision and Applications*, 34(6):1–11, 2023. 2

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1, 2

[30] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004. 2

[31] Ming Lin, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Neural architecture design for gpu-efficient networks. *arXiv preprint arXiv:2006.14090*, 2020. 4, 5, 6, 7, 8

[32] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12009–12019, 2022. 4, 5, 6

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 5

[34] Elham Maserat, Reza Safdari, Hamid Asadzadeh Aghdaei, Alireza Khoshsirat, and Mohammad Reza Zali. 43: Designing evidence based risk assessment system for cancer screening as an applicable approach for the estimating of treatment roadmap. *BMJ Open*, 7(Suppl 1):bmjopen–2016, 2017. 1

[35] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 2

[36] Rohit Mohan and Abhinav Valada. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision*, 129(5):1551–1579, 2021. 4, 6

[37] F Mosteller. Data analysis, including statistics, the collected works of john w. tukey: Philosophy and principles of data analysis 1965–1986., 1987. 1

[38] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017. 4

[39] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 5, 6

[40] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 5, 6

[41] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *arXiv preprint arXiv:2206.04670*, 2022. 5, 6

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5

[43] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016. 2, 3

[44] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018. 2

[45] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 7

[47] C Stein. Inadmissibility of the usual estimator for the mean of a multivariate distribution, vol. 1, 1956. 2, 4

[48] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 4, 5, 6

[49] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pages 10096–10106. PMLR, 2021. 7

[50] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 2, 7

[51] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2

[52] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019. 6

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 5

[54] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *arXiv preprint arXiv:1908.07919*, 2019. 4, 6

[55] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 5, 6

[56] Wikipedia. Shrinkage (statistics). *Wikipedia*, Sep 2021. 1

[57] Wikipedia. James–stein estimator. *Wikipedia*, 2023. 2

[58] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2, 3

[59] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 6

[60] Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[61] Haotian Yan, Chuang Zhang, and Ming Wu. Lawin transformer: Improving semantic segmentation transformer with multi-scale representations via large window attention. *arXiv preprint arXiv:2201.01615*, 2022. 4, 6

[62] Junjie Yan, Ruosi Wan, Xiangyu Zhang, Wei Zhang, Yichen Wei, and Jian Sun. Towards stabilizing batch statistics in backward propagation of batch normalization. *arXiv preprint arXiv:2001.06838*, 2020. 3

[63] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022. 5, 6

[64] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *European conference on computer vision*, pages 173–190. Springer, 2020. 4, 6

[65] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2

[66] Xiao-Yun Zhou, Jiacheng Sun, Nanyang Ye, Xu Lan, Qijun Luo, Bo-Lin Lai, Pedro Esperanca, Guang-Zhong Yang, and Zhenguo Li. Batch group normalization. *arXiv preprint arXiv:2012.02782*, 2020. 3

[67] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005. 2