ConvMix: Combining Intermediate Latent Features in Deep Convolutional Neural Networks

Mofassir ul Islam Arif(✉), Johannes Burchert, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Hildesheim, Germany
{mofassir,burchert,schmidt-thieme}@ismll.uni-hildesheim.de

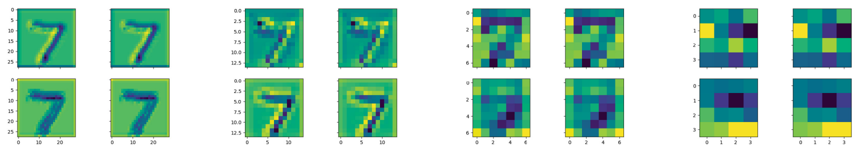
Abstract. In traditional deep learning models, latent features to the downstream task are received only from the terminal layer of the feature extractor. The intermediate layers of a feature extractor contain significant spatially salient information which, when pooled by the interleaved pooling operations, is lost. These intermediate latent embeddings can improve the overall performance for vision tasks when leveraged properly. Recently, more complex combination schemes leveraging the intermediate embeddings directly for the downstream task have been proposed, but often require additional hyperparameters, increasing their computational cost and have limited generalizability between datasets.

In this paper, we propose, ConvMix, a novel, learned combination scheme for intermediate latent features of a deep convolutional neural network which can be trained without incurring additional training cost and can be readily transferred between datasets. ConvMix leverages features at multiple stages of a CNN to distill spatial information in images, and create a richer embedding for the downstream task. Giving the network a 'wider view' by leveraging multi-level spatially pooled features of the image enables better regularization by preventing learning specific indentifying features but rather focusing on the wider image itself. We visually confirm this 'wider view' via GradCam and show that ConvMix ensure that spatially salient features are prioritized in the latent embeddings. In our experiments on CIFAR10-100, CINIC10, STL10, SVHN and TinyImageNet datasets, we show that our approach not only achieves better performance compared to state-of-the-art approaches but more importantly the percentage gain in performance scales with the increase in model/problem complexity due to the internal regularization effect of ConvMix.

1 Introduction

Deep Neural Networks (DNNs) leverage the underlying distributions of training data to learn latent embeddings which are then used for downstream tasks such as classification, detection, segmentation etc. for the computer vision domain. AlexNet [16] made use of Convolutional Neural Networks (CNN) for image classification tasks enabling state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [27]. CNNs distill spatial information in an image to an information rich representation using stacked-layer architectures to generate latent features ϕ at each

R. Gade et al. (Eds.): SCIA 2023, LNCS 13886, pp. 156–174, 2023.
https://doi.org/10.1007/978-3-031-31438-4_11



(a) Conv Block 1 (b) Conv Block 2 (c) Conv Block 3 (d) Conv Block 4

Fig. 1. Outputs of the 4 conv blocks of a Resnet18 with MNIST example. The decrease in spatial dimension can be seen here. The conv blocks produce [64, 128, 256, 512] activation maps respectively but only 4 are shown for each layer for ease of visualization.

layer which can be seen in Fig. 1. This allows the network to capture high-level features such as shapes, color, and edges in the initial layers, followed by capturing low-level features such as objects, patterns, etc. in later layers.

A CNN can be broken down into two parts $f(g(x))$ where $g(x)$ is the feature extractor for input x and $f(\cdot)$ is the downstream task. The feature extractor g maps inputs to latent embeddings $\varPhi$ such that $g : \mathbb{R}^{h\times w\times c} \rightarrow \mathbb{R}^M$ where (h, w, c) are the height, width, and channels of the image and $\varPhi \in \mathbb{R}^M$, M is the embedding size. The number of convolutional layers L corresponds to the expressivity of the network. Increasing L increases model expressivity but also leads to problems such as vanishing or exploding gradients [4]. Furthermore, deeper CNNs require more data to prevent overfitting on training data. The gradient issue has been addressed via different normalization schemes [19], while training data is augmented by transformations such as flip, scale and rotate to increase the amount of data for training. ResNets [12] were proposed to overcome the issues with gradients in deep neural networks by suggesting a skip connection from the input to the output of a stacked-layer architecture as seen in Eq. 1.

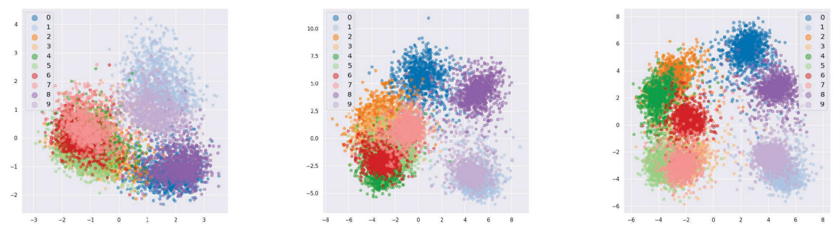
$$\phi_{l+1} = \mathcal{F}(\phi_l, \{|W_l|\}) + \phi_l \tag{1}$$

Here ϕ_l are the latent features at layer l and the function $\mathcal{F}(\phi_l, \{W_l\})$ represents the weights and biases of the layer l to be learned. Closely related to the work proposed by He et al. [2,12] presented Eq. 2 with a learnable parameter α_i for the i^{th} layer, which when initialized to zero, transforms the network into an identity function and encourages signal propagation.

$$\phi_{l+1} = \alpha_l \mathcal{F}(\phi_l, \{|W_l|\}) + \phi_l \tag{2}$$

These computationally inexpensive operations allow for deeper networks to be trained with millions of parameters and surpassed the performance of the then state-of-the-art methods such as the highway networks [30].

Recently, the focus has shifted to intelligent dataset augmentation techniques to train large models, [38,40,42] propose creating proxy instances that are a linear combination of two or more images. In the feature space, Manifold Mixup [35] put forth the creation of linear interpolation of the latent features at random using a beta distribution. Such methods are sensitive to the choice of hyper-parameters and add to the training time due to the secondary task of creating these proxy images, [26] spends 2x



(a) PCA Manifold Mixup (b) PCA ReZero (c) PCA ConvMix

Fig. 2. PCA of the latent embeddings for a ResNet18 trained of CIFAR10. Better separation indicates the ability of network to resolve between different classes. The numbers in the legend correspond to the classes.

time training and 1x time on validation. Overheads such as these are undesirable and thus encourage the need for a solution that can be used within the standard network train-test scheme.

In this work, we propose **ConvMix**, a novel combination scheme for latent features from intermediate layers in a CNN to create richer latent embeddings that contain spatially salient information of the earlier stages for better generalization and training stability. ConvMix enables end-to-end training of CNNs without the need for hyperparameter tuning by introducing only four additional learnable scaling model parameters that dictate the combination of intermediate latent features during training. The intuition behind our method comes from the spatial saliency of image data which is not being leveraged by the current CNN architectures. We argue that by using features from multiple stages of a CNN we should be able to generate well-separated embeddings and thus perform better on the downstream tasks strictly based on these richer embeddings. In Fig. 2 we present the Principle Component Analysis (PCA) for the learned embeddings for a ResNet18 network trained on CIFAR10 dataset [15] using ReZero [2], Manifold Mixup [35] and ConvMix. Figure 2 verifies our claim of a better-separated embedding space when compared to recent similar methods. We encourage the reader to refer to the appendix B for a detailed breakdown of this effect.

To quantitatively show ConvMix and its effect on the embeddings we use the CIFAR10 dataset and skew the inputs as shown in Fig. 3 to train a classifier. The skewed images result in a training situation where the images contain significant areas with no information. The traditional latent features would be limited to a small region where there is useful information. Furthermore, methods that rely on just the final embedding space would be suboptimal for such scenarios and should lead to poor performance [20]. In Fig. 3, we demonstrate that this hypothesis holds as the performance of methods that rely on final embeddings Φ deteriorates while ConvMix reports better performance owing to the higher scaling of earlier layers. Investigating the mixing weights for the intermediate layer, we see that for the four blocks of ResNet18, ConvMix generates the following normalized scaling weight values $[\beta_1 : 0.93, \beta_2 : 0.93, \beta_3 : 0.99, \beta_4 : 0.1]$. We see that the network learns a higher weight for the initial blocks while it down-

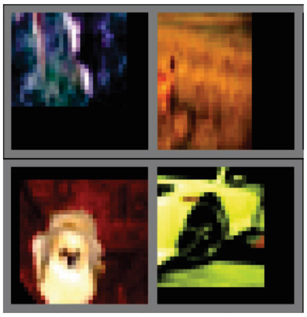
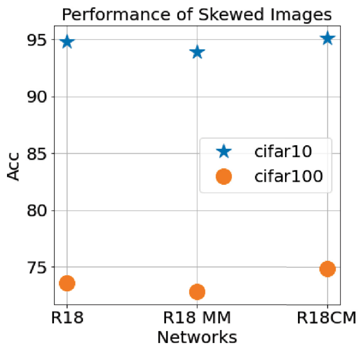



Fig. 3. Skewed CIFAR10 images and performance on skewed CIFAR10 using ResNet18 (R18), ResNet18 Manifold Mixup (R18 MM), and ResNet18 ConvMix (R18 CM) showcasing the importance of using latent features at mutliple scales.

weighs the final block to achieve higher performance compared to the base architectures. This phenomenon indicates that there is useful information contained in the outputs of the intermediate layers which needs to be directly leveraged to improve the performance of the downstream tasks.

By using learnable parameters in place of manually tuned ones, our method enables an end-to-end solution that outperforms the base architectures as well as the more involved data augmentation-based methods without adding computational complexity or demands for extensive data generation regimes. ConvMix is backbone agnostic and can be readily adapted by all CNN architectures to benefit from the gains. We demonstrate this ability of ConvMix by adapting a wide variety of networks ranging from the simple ConvNets to the more complex ResNet-based architectures such as WideResNets [39] and ResNext [37]. We also show that ConvMix scales well across network complexity ranging from ResNet18 to ResNet152. The main contributions of our work are threefold:

1. We analyze and systematize existing combination schemes for input data and latent features in CNNs while highlighting their need for hyperparameter optimization.
2. We propose a novel combination scheme for intermediate latent features that relies on trainable model parameters and aims to maximally utilize the spatially salient features at multiple stages of a CNN.
3. We demonstrate the consistency of our method across several backbones and empirically show ConvMix outperforming the more handcrafted methods.

Using CIFAR10/100 [15], CINIC10 [10], STL10 [6], SVHN [22] and Tiny ImageNet [17], we demonstrate ConvMix outperforming methods that rely on hyperparameters in both data and latent feature space. ConvMix achieves this without the need for expensive dataset generation steps of [7,8,38] or hyper-parameter overhead of [35,40].

2 Related Work

The availability of ever more powerful hardware is enabling deeper and wider networks to be investigated. The authors in [5] propose an autoregressive model with 175.0 Billion parameters to train. Similarly, in the architecture space we see models with tens of millions of parameters such as EfficientNet [31], ResNet [12] and WideResNet [39] being applied to a wide variety of machine learning tasks [14,18,25]. This increase in network depth leads to an exponential increase in model expressivity [24]. More expressivity leads to a better generalization but deeper networks come with a host of training issues as highlighted by [11]. Exploding and vanishing gradients are a big issue that has been mitigated with residual network [12] shown in Eq. 1. The identity mapping proposed in [12] enabled deeper models to be trained, bringing us ResNets with 60 million parameters. A myriad of initialization schemes [21,36,41] along with covariate shift mitigating normalization blocks [13] have enabled faster convergence of these networks. In addition to enhancing the networks, data augmentation [16] has also garnered a lot of attention, because these mammoth networks need a large amount of data to prevent overfitting. Reinforcement Learning based approaches [7] have been proposed to learn the best augmentation for datasets, however, such approaches come at the cost of computational complexity. To reduce the search space for ideal augmentation [8] proposed that it is sufficient to only search for a single distortion magnitude that jointly controls all operations, rather than searching for both magnitude and probability of each operation independently. The two main focus areas for improving the overall performance of deep networks are discussed going forward.

2.1 Instance-level Combination

The need for intelligent utilization of information contained in the dataset is paramount since one cannot simply keep adding to the model complexity to hunt for gains. To this end MixUp [40] proposed a linear interpolation on input data to create virtual training instances x' and labels y' governed by a mixing parameter $\lambda \sim Beta(\alpha, \alpha)$ in Eq. 3, here α is a hyper-parameter.

$$\begin{aligned} x' &= \lambda x_i + (1-\lambda)x_j \\ y' &= \lambda y_i + (1-\lambda)y_j \end{aligned} \tag{3}$$

In the same vein, CutMix [38] has been proposed which inserts a patch of a data instance atop another to create a proxy instance that is used to train the network. CutMix involves pixel-level training to arrive at the ideal patch to overwrite in the original image, CutMix also updates the ground-truth labels to carry forward the patched instance label. A similar approach termed SuperMix [9] has been proposed which creates the mixed training instances based on the salient regions in images. These methods rely on an expensive supervised training regimen to generate the training samples needed for a deep network, adding a significant computational and memory overhead.

2.2 Feature-level Combination

Manifold Mixup (MM) [35] builds upon the mixup [40] idea by applying it to the latent features. Instead of generating a mixed-up training instance and label, MM seeks to mix up the latent embeddings of the input instances at layer l. Effectively, MM applies Mixup to the embedding space as shown in Eq. 4.

$$(\tilde{g}_l, \tilde{y}) := (Mix_\lambda(g_l(x), g_l(x')), Mix_\lambda(y, y')) \tag{4}$$

Here g_l is the latent embedding at layer l for instance x and x' and λ is the mixing parameter similar to Eq. 3. These approaches rely on creating a linear interpolation of the input space or the latent features which have shown substantial improvements over the state-of-the-art methods. Phantom Embeddings [33] proposes to create weighted mixed-up latent embeddings for instance x and x' to penalize the model from learning on the training data and improving generalization by maximizing the inter-class distances of the embeddings.

ReZero [2] builds on the residual network by introducing a learnable residual weight α for each block which is initialized to zero in order to turn the network into an identity mapping at the start of training as in Eq. 5.

$$\phi_{l+1} = \alpha_l \cdot \mathcal{F}(\phi_l, \{W_l\}) + \phi_l \tag{5}$$

Here $\alpha \in \mathbb{R}^L$ trained along with the weights of the network. All methods in Sect. 2 fail to account for the explicit spatially salient information in the training data. They follow the stacked-layer architecture and seek gains by augmenting the training data to improve generalization. ConvMix proposes a novel research direction whereby the dataset is maximally leveraged by tapping into the latent features at multiple stages of the stacked-layer architecture to generate an embedding space that is informed by the spatially salient features of an image.

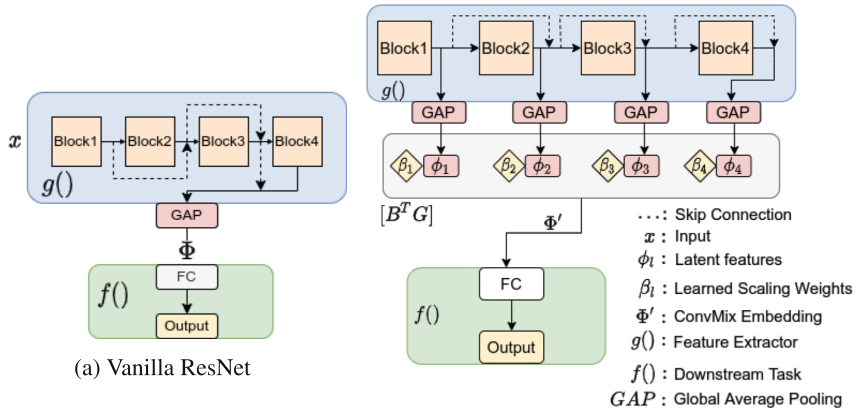
3 ConvMix: A Novel Combination Scheme for Latent Features

Spatial information in ordered data generally and in the image domain specifically carries a significant signal that needs to be leveraged maximally for the downstream tasks. Using just the final embedding space $\varPhi$ does not utilize the high-level feature information contained in the earlier stages of the network and leaves gains on the table.

Recalling that a CNN can be broken down into $f(g(x))$, we can break it up further by looking at the components inside the feature extractor $g(\cdot)$, represented as $\mathbf{G} = [\phi_1, \ldots, \phi_L]$. Here ϕ_l is the intermediate latent feature at layer l. The vanilla ResNet in Fig. 4a can, therefore, be represented as $f(g_L(\phi_{L-1}))$ or simply as $f(\varPhi)$, where $\varPhi$ is the final latent embedding used by the downstream task $f(\cdot)$. ConvMix aims to use the latent features at multiple levels of the feature extractor as seen in Eq. 6 to generate a spatially enriched embedding $\varPhi'$. The architecture of ConvMix and its comparison to the vanilla ResNet can be seen in Fig. 4.

$$\begin{aligned} ConvMix = \varPhi' &= [\beta_1 \cdot \phi_1, \ldots, \beta_L \cdot \phi_L] \\ &= \mathbf{B}^T\mathbf{G} \end{aligned} \tag{6}$$

Here $\mathbf{B} \in \mathbb{R}^L$ is the set of learnable scaling weights that we introduce to combine the latent features at all stages of the feature extractor. We allow the learned scaling weights



(a) Vanilla ResNet

(b) ResNet Architecture with ConvMix

Fig. 4. Standard ResNet Architecture is shown in 4a and 4b shows the architecture for our proposed method, ConvMix.

B to dictate the combination of latent features to achieve desirable behaviors during training namely richer embeddings (discussed in Sect. 3.2) and network compression (discussed in Sect. 3.3)

We take an input $x \in \mathbb{R}^{h\times w\times c}$ and pass it through the multiple stages of the feature extractor **G**, we tap into the latent features at each stage (ϕ_l) and apply Global Average Pooling (GAP) [20]. At this point we are left with a channel-wise pooled ϕ_l which we scale via $\beta_l \in$ **B**. These scaled latent embeddings are then aggregated to form the ConvMix embedding Φ' as seen in Eq. 6. The addition of **B** does not impact the objective function because **B** acts as a scaling factor for the intermediate latent features and is passed through to the cost function $J(y, \hat{y})$. We can demonstrate the scaling effect by creating a toy residual network with L layers and identity activations. Furthermore, the L layers share the same $\mathcal{F}\{\mathcal{W}\}$ and **B** weights. The output of the feature extractor at each level l, would be scaled by the factor β_l. We can represent such a network as in Eq. 7, where x is an input that is mapped to output $\hat{y}$ through L hidden layers. We substitute θ for $\mathcal{F}\{\mathcal{W}\}$ for ease of notation.

$$\hat{y}(x) = f(\beta^L(1+\theta)^L x) \tag{7}$$

We have stated that $\beta_l \in$ **B** so $\beta^L =$ **B**. Similarly, from Eq. 1 we know that $(1+\theta)^L x = \phi^L =$ **G** and $f(\cdot)$ is our downstream task. We can replace these terms in Eq. 7 to arrive at $f(\mathbf{B}^T\mathbf{G})$ which from Eq. 6 is the enriched embedding Φ'. This toy example shows how ConvMix serves to combine the intermediate latent features. Assuming a learning rate η, the gradient descent weight update would therefore be represented by Eq. 8 and we can see that ConvMix acts as a scaling factor.

$$\theta \leftarrow \theta - \eta L x \beta^L (1+\theta)^{L-1} \partial_\theta J(y, \hat{y}) \tag{8}$$

Since **B** is a learnable parameter, it will be updated similar to [2] represented in Eq. 9.

$$\beta \leftarrow -\eta L\theta x \partial_\theta J(y, \hat{y}) \tag{9}$$

In order to delineate our work from Vanilla ResNet and ReZero we utilize the same toy example, resulting in $\hat{y}_{\text{ResNet}}(x) = f((1+\theta)^L x)$ and $\hat{y}_{\text{ReZero}}(x) = f((1+\alpha \cdot \theta)^L x)$ respectively, here α is the residual weight proposed by [2]. Comparing ReZero to Eq. 7, we can notice an important difference between ConvMix and ReZero. ReZero performs a pixelwise scaling of the weights while ConvMix, by using multiple stages, is a channel-wise scaling of the weights enabling us to maintain the spatial information.

The need for learnable weights $\mathbf{B}$ for combining latent features comes from the undesirable alternatives. A naive way to create the Φ' would be to average $\Phi = \frac{1}{L}\sum_{l=1}^{L}\phi_l$ where ϕ_l is the l^{th} latent embedding from $g_l(\phi_{l-1})$. However, doing so puts equal weight on latent features from inconsistent sizes since activation map count increases polynomially with an increase in layers. Alternatively, we could assign weights $\mathbf{B} \in \mathbb{R}^L$ to $\mathbf{G}$ for a weighted average such that $\Phi = \frac{1}{L}\sum_{l=1}^{L}\beta_l \cdot \phi_l$ but that would add more heuristics to train requiring additional hyper-parameter tuning while not being transferable across different datasets.

3.1 Benefits of ConvMix

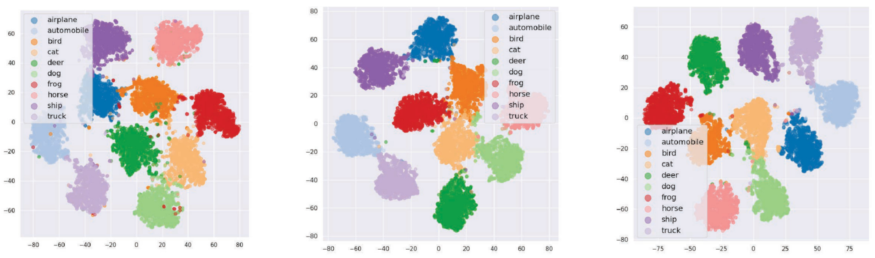
Having laid out the working of ConvMix, we would like to point out a few of the positive effects of our approach on the network.

3.2 Richer Embeddings

The feature extractor output Φ directly impacts the performance of the network since it is the input to $f(\cdot)$. An ideal Φ can be broken down into N specific regions corresponding to each of the N labels. Capturing the latent features at multiple levels enables us to create a richer embedding space which can be empirically demonstrated by comparing the separability of embeddings for ConvMix against ReZero and Manifold mixup. Using Eq. 10, we can calculate the separation between the embeddings of class i and j.

$$D_{ij} = \mathop{\mathbb{E}}_{z\in i, z'\in j}\left[\frac{1}{M}\|z - z'\|^2\right] \tag{10}$$

Here, z and z' are the test set latent embeddings generated from training ResNet18 on CIFAR10 data for classes i and j. We normalize the distance metrics by the length of the embedding M to discount for differing sizes of the embedding spaces. Using upper triangular D_{ij} we can calculate the expected separation as $Separation = \frac{1}{N(N-1)}\sum_{i,j} D_{ij}$. The separation for ConvMix (**0.394**) is better than that for Manifold MixUp (0.1933) and ReZero (0.285). ConvMix generates embeddings that have a higher expected separation thus indicating that each class is better separated in the embedding space which leads to a higher performance on the downstream tasks which can be seen in the experiment section. We present TSNE [34] plots in Fig. 7 for a more qualitative look at the separability. When dealing with class clusters, we would like to see two characteristics i-e. the centers should be well separated and the spread of the intra-class samples should be tight. These two characteristics indicate that the method



(a) TSNE Manifold Mixup (b) TSNE ReZero (c) TSNE ConvMix

Fig. 5. TSNE of the latent Embeddings for a ResNet18 trained of CIFAR10. Better separation indicates the ability of the network to resolve between different classes.

has learned unique embeddings for the different classes and therefore should be a better classifier. The latter claim is verified in the experiment section with results on the CIFAR10-100, CINIC10, SVHN, STL10 and Tiny ImageNet datasets using a wide range of networks. In Fig. 7, we can see that the class centers are better separated in our methods when compared to the baselines. Due to space constraints, we direct the reader to the appendix for a more detailed discussion on these plots.

Table 1. KL-Divergence of the Embedding Space for the test split of CIFAR10. Trained using ResNet18

	CIFAR10	CIFAR100
Manifold Mixup	1.779358	1.030353
ReZero	1.765742	1.032229
ConvMix	**1.697480**	**1.028727**

We also report the KL-Divergence to highlight that the embedding space of ConvMix results in a lower KL-Divergence when compared to the baseline methods.

3.3 Network Compression

In information theory, compression of embedding space leads to better generalization [29,32] and also has a regularization effect as shown by [1,3]. We provide empirical proof of this compression in our method by training a CNN on the MNIST dataset and analyzing the spectral decay using Singular Value Decomposition (SVD) for no regularization, weight decay, dropout, and ConvMix at the first convolution layer. We follow [35] and compare the maximum singular value per class label. The results show baseline (131), weight decay (83.1), dropout (58.4), and ConvMix (**48.9**). The decrease in maximal singular value using regularization techniques can be empirically seen here. Also

Table 2. Comparing the performance of ConvMix (CM) against Manifold Mixup (MM) and ReZero(RZ). We present the mean and standard deviation of 5 runs. The networks we test are ResNet18, 34 and 50 (**R18**, **R34** and **R50**). Best performance is highlighted in bold

	Cifar-10	Cifar-100	CINIC-10	STL10	SVHN
	Prec@1 ↑				
R18 MM	93.79 ± 0.16	75.15 ± 0.95	87.59 ± 0.08	86.55 ± **0.25**	96.58 ± 0.06
R18 RZ	94.59 ± 0.35	75.52 ± 0.96	87.31 ± 0.14	82.01 ± 0.38	96.40 ± **0.02**
R18 CM	**95.25** ± **0.10**	**77.56** ± **0.28**	**87.77** ± **0.02**	**89.63** ± 0.26	**96.65** ± 0.11
R34 MM	94.1 ± 0.62	77.27 ± 1.10	86.36 ± 0.13	86.84 ± **0.18**	95.81 ± 0.30
R34 RZ	94.65 ± **0.13**	75.94 ± 0.76	87.82 ± **0.02**	83.18 ± 0.30	96.43 ± **0.01**
R34 CM	**95.34** ± 0.15	**77.91** ± **0.55**	**87.87** ± 0.08	**89.67** ±0.44	**96.82** ± 0.08
R50 MM	92.38 ± 0.33	77.89 ± **0.35**	85.02 ± 0.25	84.50 ± **0.46**	95.10 ± 0.60
R50 RZ	95.46 ± 0.12	76.51 ± 0.60	84.67 ± 0.15	85.44 ± 0.54	96.45 ± 0.09
R50 CM	**95.72** ± **0.06**	**79.08** ± 0.53	**88.85** ± **0.10**	**89.45** ± 0.52	**96.77** ± **0.07**

highlighted here is that ConvMix leads to a higher network compression thus demonstrating it's regularization effect which enables us to perform better than the baseline methods is difficult setting i-e. expressive networks and insufficient data.

4 Experiments

We verify the efficacy of our method on image classification tasks using six popular image datasets namely, CIFAR10, CIFAR100, CINIC10, STL10, SVHN and Tiny Imagenet (100K). CIFAR10 dataset contains a total of 60K (50K train and 10K test instances) 32×32 color images in 10 classes, with 5K images per class. CIFAR100 dataset is similar to the CIFAR10 dataset but with 100 classes and 500 images per class. CINIC10 is a variant of CIFAR10 sampled from the ImageNet dataset and is a more challenging usecase since it is 4.5 times bigger than CIFAR10 with 90,000 training image per class. STL10 presents the opposite situation of CINIC10 by having only 5000 labeled training images and 9000 test images. Tiny ImageNet is a subset of the ImageNet dataset in the ILSVRC [27] and contains 100K 64×64 images from 200 classes, each class containing 500 images. SVHN10 contains 600,000 images with numbers from 0 to 9 as the target. These datasets give us a range of complexity from easy(CIFAR10) to complex(Tiny Imagenet) allowing us to empirically show ConvMix's performance in different training scenarios. In addition to difficulty in terms of the datasets, we also use the popular block-based ResNets namely, ResNet18-34-50-101-152, Wide ResNet28-50, and ResNext50 to highlight how our method can cope with varying model complexity. We discount methods that generate mixed-up instances for training because our method does not rely on dataset regeneration, we take Manifold Mixup and ReZero as our primary baselines because they are the closest to our approach in that they operate on the feature level.

Table 3. Pretrained warm-up training on Vanilla ResNet vs ConvMix. ConvMix outperforms the baselines. The bigger gains can be seen for TinyImagetNet and the more complex networks ResNet101-152.

	Cifar-10		Cifar-100		TinyImageNet	
	Prec@1↑	Gain	Prec@1↑	Gain	Prec@1↑	Gain
Resnet18	95.59 ± 0.430		78.08 ± 1.45		70.80 ± 0.146	
Resnet18 ConvMix	**95.93** ± 0.246	0.34%	**79.82** ± 0.419	1.74%	**72.42** ± 0.101	1.62%
Resnet34	96.52 ± 0.247		81.79 ± 0.811		75.66 ± 0.206	
Resnet34 ConvMix	**96.61** ± 0.171	0.09%	**82.34** ± 0.741	0.55%	**76.97** ± 0.05	1.31%
Resnet50	96.67 ±0.141		83.26 ± 1.01		77.19 ± 0.115	
Resnet50 ConvMix	**97.11** ± 0.155	0.44%	**83.6** ± 0.640	0.34%	**79.61** ± 0.165	2.42%
Resnet101	97.06 ± 0.100		83.53 ± 0.633		79.18 ± 0.361	
Resnet101 ConvMix	**97.17** ± 0.012	0.19%	**84.89** ± 0.104	1.36%	**82.8** ± 0.269	3.6%
Resnet152	97.36 ± 0.156		84.25 ± 0.205		79.49 ± 0.105	
Resnet152 ConvMix	**97.44** ± 0.143	0.08%	**84.83** ± 0.01	0.58%	**83.34** ± 0.163	3.85%

4.1 Implementation Details

For CIFAR10 and CIFAR100 we use the data augmentation scheme as in [16]. All experiments were run for 600 epochs with an initial learning rate of 0.1 (unless otherwise stated) and was decreased by a factor of 10 at 50% and 75% of the total epochs. Stochastic Gradient Descent (SGD) optimizer was used with a weight decay of 0.0001 and momentum of 0.9. All backbones networks are taken from the PyTorch Torchvision models library [23] and then adapted using for our method. For a fair comparison, we tune the hyper-parameters for the baselines and compare them against our method. For Tiny Imagenet we use an initial learning rate of 0.001 as it provided the best baseline performance, the rest of the training regime is maintained similar to the other datasets under consideration.

4.2 ConvMix in Classification Tasks

ConvMix relies on the latent features of multiple stages of a CNN and to verify the claim that information contained in the earlier stages is useful for the task of image classification we compare the performance of Manifold Mixup and ReZero on the above mentioned datasets and present our findings in Table 2. We reimplemented the baselines and compared them against our method and note that ConvMix outperforms both Manifold Mixup and ReZero for Prec@1 for ResNet(18-34-50) for all datasets. Considering CIFAR100, CINIC10 and STL10, the more challenging datasets, we see that our method has substantial gains over the baselines in terms of Prec@1 for ResNet50. The increasing performance gap between the baselines and ConvMix when the problem complexity increases shows that our approach has an internal regularization effect that enables better generalization and test performance, as established in Sect. 3.3.

Table 4. Ablation study: Showcasing the impact of learning **B** end-to-end vs using the learned **B** values as initializations. We also show how ConvMix (CM) can be used in addition to other methods namely, ReZero(RZ)+CM and the resultantly increase in performance over ReZero.

	R18 ConvMix	R18 Fixed B	R34 ConvMix	R34 Fixed B
Cifar10	**95.25**	92.95	**95.34**	94.6
Cifar100	**77.56**	74.2	**77.91**	76.5
	R18 RZ	**R18 RZ+CM**	**R34 RZ**	**R34 RZ+CM**
Cifar10	94.59	**94.99**	94.65	**95.23**
Cifar100	75.52	**76.56**	75.94	**76.48**

In Table 3 we present the results of our model averaged over 5 runs against the stock variants of the backbones. We use pre-trained weights learned on the ImageNet dataset, starting with a learning rate of 0.01, and train all networks for 300 epochs. It can be seen that our method is better in the final accuracy across the board. Furthermore, from the standard deviation, we see that the model also performs consistently better than the stock variant. For CIFAR10, the gains are small but consistent as it is a smaller dataset and easy to learn with deep networks such as ResNets. CIFAR100 and Tiny ImageNet are the more challenging datasets and ConvMix can outperform the baselines under similar training settings. We also see an increase in %gain for ConvMix as the complexity of the problem and the model increases.

A key item to note here; For CIFAR10 data using ResNet50 ConvMix yields an accuracy of 97.11% while vanilla ResNet101 being a significantly bigger model is at 97.06%. The same trend can be seen for CIFAR100 used on ResNet34 and ResNet50 where our approach can close the gap between itself and the one level higher network. This gain is attributed to the increase in embedding quality generated using our approach.

4.3 Tiny ImageNet

To check the efficacy of ConvMix on challenging problem settings, we used TinyImagenet with 200 classes and 500 instances per class. We make a small change to the preprocessing where we scale the images from 64×64 to 224×224 pixels to use a wider stride for the convolutions as in [12]. We also use pre-trained weights of the complete ImageNet dataset for every backbone and use a lower learning rate (0.001). These steps were taken to overcome the time constraints of training on bigger datasets. To keep a fair comparison, we provide the same preprocessing pipeline to the baseline variants as we do to the ConvMix, enabling us to see the effect of ConvMix while every other experimental parameter is kept the same. As seen in Table 3, ConvMix categorically outperforms the baselines for Tiny ImageNet. It should be noted, that our biggest gains are seen when dealing with the most complex task i-e. TinyImageNet and for the most expressive networks i-e. Resnet101-152. These networks are prone to overfitting,

Table 5. Integration of ConvMix to more complex ResNets. CM represents ConvMix.

	Cifar10	Cifar100
	Prec@1 $\uparrow$	**Prec@1** $\uparrow$
ResNext50	97.09 $\pm$ 0.379	84.18 $\pm$ 0.506
ResNext50 CM	**97.29** $\pm$0.176	**84.70** $\pm$ 0.753
WideResNet28-2	94.615 $\pm$ 0.050	72.53 $\pm$ 0.064
WideResNet28-2 CM	**94.66** $\pm$ 0.191	**73.21** $\pm$ 0.042
WideResNet28-10	95.71 $\pm$ 0.120	78.82 $\pm$ 0.220
WideResNet28-10 CM	**95.86** $\pm$ 0.106	**78.88** $\pm$ 0.120
WideResNet50	96.68 $\pm$ 1.337	84.15 $\pm$ 0.400
WideResNet50 CM	**97.35** $\pm$0.250	**84.72** $\pm$ 0.230

however, with ConvMix, we can see consistent and substantial gains in this challenging setting. Thus, indicating that ConvMix also has an automatic regularization effect during training.

4.4 Ablation Study

To ascertain that the gains seen in Table 2-3, stem from the learned nature of weights **B**, we create a second experiment where we initialized **B** with the weights learned from a previous run and do not treat **B** as a learned parameter. The aim here is to prove that by learning the weights during training we can improve the overall performance of the network. Furthermore, with this experiment, we can discount our contribution from being an initialization effect. The results are presented in Table 4, for CIFAR10 we were able to achieve 92.9% and 94.6% Prec@1 for ResNet18 and ResNet34 respectively, which is a 2.69% and 2.01% decline in the overall performance when compared to treating **B** as a learned parameter. For CIFAR100, we see a similar decline of 3.36% and 1.41% respectively. We also show the improvements ConvMix can offer to other methods by implementing ConvMix with ReZero and showing the improvements over vanilla ReZero. For CIFAR10, we report a 0.4% and $\sim$ 0.6% increase over ReZero ResNet18 and ResNet34 respectively. For CIFAR100, we report a 1.04% and 0.54% increase over ReZero ResNet18 and ResNet34 respectively.

Lastly, we also implement ConvMix to the more advanced variants of ResNet models such as ResNext [37], and WideResNets to show that our model outperforms or matches the performance of these methods. WideResNets natively come with a host of hyper-parameters and we did not do an extensive search for the optimal ones for ConvMix because that is not the intent of our work. We see that ConvMix leads to an overall improvement over the baseline variants of these advanced networks. For challenging datasets (CIFAR100) and deeper networks such as WideResNet50 and ResNext50, ConvMix outperforms the baseline architectures significantly. This indicates that while the baseline methods struggle to generalize for deeper networks due to limited data and disproportionate model complexity, ConvMix, with its internal regularization using

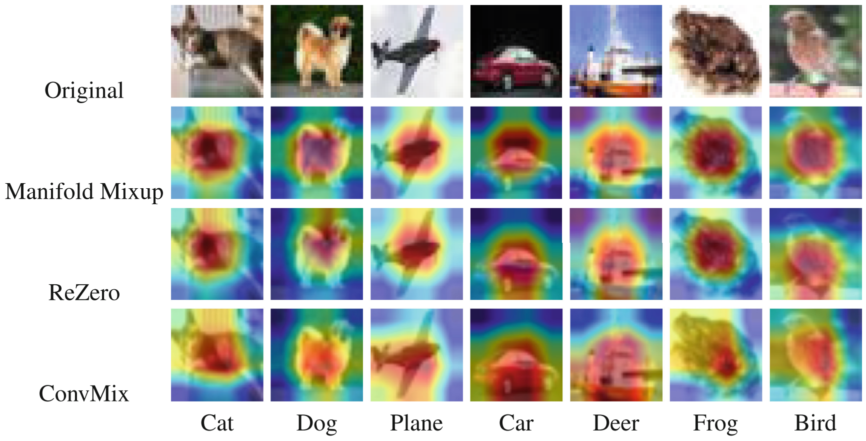


Fig. 6. GradCam visualizations for CIFAR10 dataset trained using ResNet18, comparing Manifold Mixup, ReZero, and ConvMix. The Dark red region indicates the salient features in the images learned by the network. GradCams for ConvMix embeddings (at layer4) indicate that our method learns embeddings that take into account a wider area of the input. (Color figure online)

latent features of multiple stages, can improve the overall performance. We present GradCam [28] in Fig. 6 images to highlight the generalization effect of ConvMix (more details can be seen in appendix C). It can be seen that baselines tend to focus on a very small region to make while ConvMix allows the model to learn from a wider area of the image. This is a visual representation of the regularization offered by ConvMix enabling it to outperform the baselines.

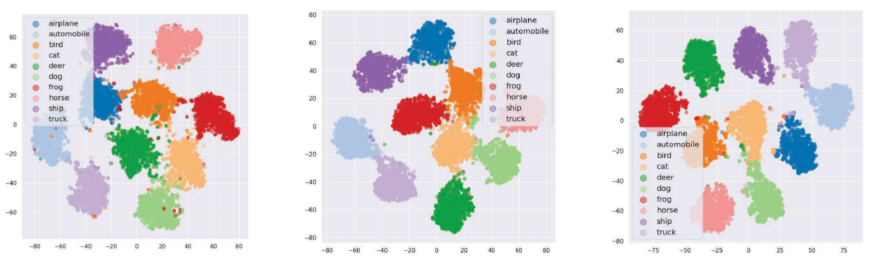
5 Conclusion

In this work, we proposed, ConvMix, a novel end-to-end combination scheme for latent features of a CNN built upon the assumption that the current network architectures fail to account for the salient information in images. ConvMix learns mixed latent features aided by a learned weight **B** by tapping into the latent features at multiple stages of the network. Being a learned method, ConvMix does not add any hyper-parameters to be tuned thus, avoiding additional training time overhead. ConvMix is also backbone-agnostic and can be readily implemented in any stacked-layer architecture. We empirically show the benefits of ConvMix in its ability to improve training behavior, network compression, and richer embeddings allowing it to outperform the methods that fail to account for the spatial information in images. We also provide extensive experimental evidence of our method's efficacy and how well it scales across network complexity. We hope our work can drive interest in exploring the information contained within a dataset in an end-to-end manner and bring forward more methods that reduce the hyper-parameter overheads of the current approaches.

A Appendix

B Richer Embeddings

Continuing our discussion about the ConvMix generating better-separated embeddings, in Fig. 7 we present the TSNE plots on the test set embeddings of a ResNet18 trained on CIFAR10 dataset. We have quantified the inter-class already in the main text (Table 1), here we present qualitative proof for our claim that leveraging the latent features at multiple stages generates better embeddings. In Fig. 7 we compare methods that work on the latent space, namely Manifold MixUp and ReZero, against ConvMix.



(a) TSNE Manifold Mixup (b) TSNE ReZero (c) TSNE ConvMix

Fig. 7. TSNE of the latent Embeddings for a ResNet18 trained of CIFAR10. Better separation indicates the ability of the network to resolve between different classes.

Analysing Fig. 7, we can see that the class centers and better separated in our methods when compared to the others but a deeper inspection is needed to see how the different classes are represented in by our method. CIFAR10 dataset has some classes that are frequently confused together, namely Cats-Dogs, Airplane-Ship, and the 4-legged animals Cats-Dogs-Deers. This effect can be seen in the TSNE plots, we looking at Manifold MixUp we can see this clearly with cats, dogs, and deers all being clustered in relatively similar areas. This is an intuitive finding since these classes are intrinsically close together however, this also leads to misclassification. In our method, we see that while cats and dogs occupy a close place in the embedding space, the deer class is well separated. This effect is caused by the earlier features of the CNN being used since in the earlier stages of a CNN still maintains spatial saliency in the image features.

Moving on to the other troublesome classes, Airplane-Ship. We can rationalize why they would be placed together in the embeddings space by Manifold MixUp and ReZero. Both these classes contain a lot of blue in them due to the sea and sky. In our method, we see a substantial separation between the two classes indicating that using the latent features at multiple stages have enabled to model to resolve between the shapes of the subject in the pictures and therefore, place them in well separated embedding spaces.

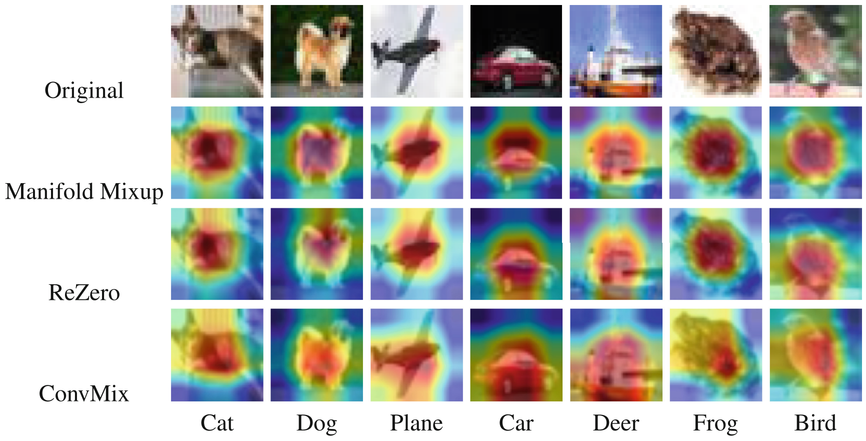


Fig. 8. GradCam visualizations for CIFAR10 dataset trained using ResNet18, comparing Manifold Mixup, ReZero, and ConvMix. The Dark red region indicates the salient features in the images learned by the network. GradCams for ConvMix embeddings (at layer4) indicate that our method learns embeddings that take into account a wider area of the input. (Color figure online)

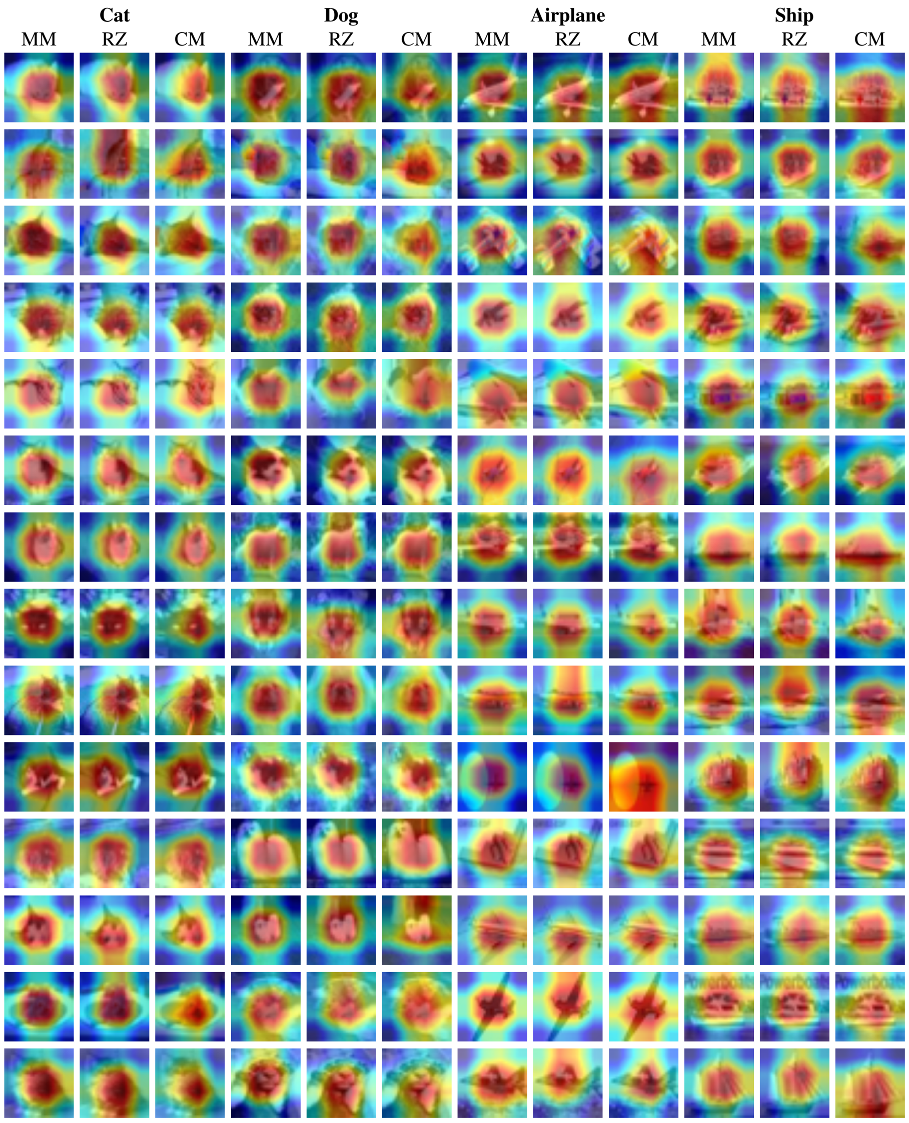
C GradCAM for Explainability

We argue in the main text of the paper that ConvMix has an internal regularization effect, enabling the model to generalize better than the baseline methods (Manifold MixUp and ReZero). We qualitatively demonstrate this effect in the final model by presenting the gradCAMs on the CIFAR10 validation split for a fair demonstration of the viability of our claim. In Fig. 6 we show that both Manifold Mixup and ReZero tend to have a very narrow focus and use key areas in the image to classify the images. However, ConvMix shows that it makes the classification by using a more holistic view of the input. This can be seen in the last row of Fig. 6, ConvMix enables the model to maintain a "wider view" of the input by spreading the focus on the subject as a whole rather than just key aspects of the input.

The key benefit of this characteristic can be realized when we take the learning from Fig. 6 and put it in the context of Fig. 7. We have a peculiar situation seen in the TSNE plots where Cats, Birds, and Frogs are located close together in both Manifold Mixup and ReZero due to this focus on key areas. Using ConvMix, we can see that Cats and Birds still occupy a similar but well-separated space, however, frogs have been moved away further away in the embedding space. This shows a strength of ConvMix to be able to use the features of frogs are multiple stages and rightly place them away from birds and Cats.

Another example we would like to point out here is the Automobile-Ship pair, which can be seen to be closer together in the Manifold Mixup and ReZero method. We can understand what the model is trying to do here by comparing the GradCAMs for Automobiles and Ships. With the narrow view of Manifold Mixup and ReZero, the model sees similar features such as windows, doors, and frames. However, by looking at the wider view offered by ConvMix, we can see the model making use of the entire image for classification. Resultantly, we see that Automobiles and Ships are well separated in the embedding space.

Table 6. A random sampling of GradCAMs to showcase that the generalization effect discussed above holds for the majority of the dataset. A wider spread is more desirable since it enables the model to learn a more general representation of the class.



References

1. Achille, A., Soatto, S.: Information dropout: Learning optimal representations through noisy computation. IEEE Trans. Pattern Anal. Mach. Intell. **40**(12), 2897–2905 (2018)
2. Bachlechner, T., Majumder, B.P., Mao, H., Cottrell, G., McAuley, J.: Rezero is all you need: Fast convergence at large depth. In: Uncertainty in Artificial Intelligence, pp. 1352–1361. PMLR (2021)
3. Belghazi, M.I., et al.: Mine: mutual information neural estimation. arXiv preprint arXiv:1801.04062 (2018)
4. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
5. Brown, T., et al.: Language models are few-shot learners. Adv. Neural Inform. Process. Syst. **33**, 1877–1901 (2020)
6. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 215–223. JMLR Workshop and Conference Proceedings (2011)
7. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
8. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 702–703 (2020)
9. Dabouei, A., Soleymani, S., Taherkhani, F., Nasrabadi, N.M.: Supermix: Supervising the mixing data augmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13794–13803 (2021)
10. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: Cinic-10 is not imagenet or cifar-10. arXiv preprint arXiv:1810.03505 (2018)
11. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034 (2015)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)
14. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing systems, vol. 25 (2012)
17. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N, **7**(7), 3 (2015)
18. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
19. LeCun, Y.A.,. Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012). https://doi.org/10.1007/978-3-642-35289-8_3
20. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
21. Mishkin, D., Matas, J.: All you need is a good init. arXiv preprint arXiv:1511.06422 (2015)
22. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)

23. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. In: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems vol. 32, pp. 8024–8035. Curran Associates Inc (2019)
24. Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., Ganguli, S.: Exponential expressivity in deep neural networks through transient chaos. Advances in neural information processing systems vol. 29 (2016)
25. Radford, A., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
26. Ramé, A., Sun, R., Cord, M.: Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 823–833 (2021)
27. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
28. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)
29. Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. arXiv preprint arXiv:1703.00810 (2017)
30. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
31. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning, pp. 6105–6114. PMLR (2019)
32. Tishby, N., Zaslavsky, N.: Deep learning and the information bottleneck principle. In: 2015 ieee information theory workshop (itw), pp. 1–5. IEEE (2015)
33. Arif, M.U.I., Jameel, M., Grabocka, J., Schmidt-Thieme, L.: Phantom embeddings: Using embeddings space for model regularization in deep neural networks. In: LWDA, pp. 47–58 (2020)
34. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**(11) (2008)
35. Verma, V., et al.: Manifold mixup: learning better representations by interpolating hidden states (2018)
36. Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S., Pennington, J.: Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In: International Conference on Machine Learning, pp. 5393–5402. PMLR, (2018)
37. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1492–1500 (2017)
38. Yun, S., Han, D., Oh, J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6023–6032 (2019)
39. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
40. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
41. Zhang, H., Dauphin, Y.N., Ma, T.: Fixup initialization: Residual learning without normalization. arXiv preprint arXiv:1901.09321 (2019)
42. Zhu, J., Shi, L., Yan, J., Zha, H.: Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In: European Conference on Computer Vision, pp. 633–649. Springer (2020)