Learning Less-correlated Features in Network Aggregation

Anonymous submission

Abstract

This paper proposes a novel learning method to leverage multiple representations effectively. Aggregated features after individual training or after going through extra complicated heads are prone to cause redundancy in the feature space. Instead, we explicitly push the representations to be less correlated during training. Specifically, networks learn different representations of the target task with lowered redundancy by explicitly training with the proposed decorrelation loss. Furthermore, we propose a new network architecture consisting of lightweight sub-networks, which is turned out to be efficient yet has high capability compared with the prior arts using heavy head architectures. It collaborates with the proposed learning method to learn more less-correlated features. We additionally provide an analysis to reveal the relationship between the less-correlated features and performance. Finally, our proposed model outperforms recent state-of-the-art models with higher throughput evaluated on ImageNet. We believe the resultant model is a positive byproduct of the collaboration of less-correlated feature learning with the efficient architecture design. Our code will be publicly released.

Introduction

Over the past decade, numerous methods (Newell, Yang, and Deng 2016; Lin et al. 2017; Yu et al. 2018; Sun et al. 2018; Ryu, Yang, and Lim 2018; Du et al. 2020) have been proposed to achieve better performance on various tasks. They fall into a single line of study which aggregates multiple extracted features and aim to predict class labels (Yu et al. 2018; Sun et al. 2018) or bounding boxes/masks (Lin et al. 2017; Du et al. 2020; Liu et al. 2018; Tan, Pang, and Le 2020). Although the traditional methods have been proven to be simple yet effective, the features are prone to be trained with a high correlation, as disclosed in the literature (Srivastava et al. 2014; Cogswell et al. 2015; Ryu et al. 2019; Huang et al. 2018; Hua et al. 2021). Therefore, the architecture may not have maximal expressiveness.

From the architectural design perspective, features are usually extracted at different stages of a network to embed diverse architectural properties in the learned representation. Therefore, the architectures (Newell, Yang, and Deng 2016; Lin et al. 2017; Sun et al. 2018; Du et al. 2020) incorporate many trainable layers to refine the features to be diversified. For example, a well-designed architecture (Lin et al. 2017) successfully improves task performance due to learning more discriminative features by the new architectural design with many additional parameters. Therefore, it consumes substantial computational overheads of training due to the complicated extra layers that provide the feature extraction points to aggregate multiple features. However, whether it really learns by reducing the correlation of the representation has not been well investigated.

In this paper, we present a new learning method with a network architecture aiming to learn less-correlated features for improved performance. First, we introduce a learning technique that forces the shallow individual sub-networks to learn different representations of each other. Second, we propose an efficient architecture with multiple shallow sub-networks to learn more diversified multi-level features instead of heavily incorporating many trainable layers to refine features. Overall, our proposed method compels the lightweight sub-networks to learn different yet strong representations, resulting in the improved capability of the final feature. We empirically found that our learning technique and network architecture effectively lower the correlation between the output features compared with the method of aggregating features straightforwardly.

We further analyze the proposed method regarding the strength and correlation theory (Ryu et al. 2019; Breiman 2001) with the generalization error bound. According to the literature, an aggregated network is generalized when the correlation is lowered while the strength is increased. It is known that the correlation and strength are interconnected, but interestingly, training with our proposed method of lowering correlation will eventually increase strength, as similarly shown in Ryu et al. (2019). In addition, the generalization error bound gets decreased as we gradually append our design elements and train with the proposed learning technique.

Finally, we perform experimental evaluations with our proposed method trained on the CIFAR (Krizhevsky 2009) and ImageNet (Russakovsky et al. 2015) datasets to show the effectiveness. We systematically compare the models with similar computational budgets, including throughput; ours beat the competing network architectures in the accuracy and throughput trade-off. Furthermore, we validate our ImageNet-pretrained models by fine-tuning on the fine-grained visual classification datasets, including CUB-200 (Wah et al. 2011), Food-101 (Bossard, Guillaumin, and Van Gool 2014), Stanford Cars (Krause et al. 2013), FGVC Aircraft (Maji et al. 2013), and Oxford Flowers-102 (Nilsback and Zisserman 2008)[1]. Our contributions are summarized as follows:

(i) We propose a learning technique with a new decorrelation loss that regularizes multiple networks to yield less-correlated features to each other.

(ii) We propose a new design regime incorporating lightweight sub-networks instead of multiple backbones or complicated heads. Refined features by the sub-networks are architecturally less-correlated and further adjusted by the proposed less-correlated feature learning. Intriguingly, the learning method shows a much faster convergence speed than the traditional training one.

(iii) We provide an analysis to reveal the effectiveness of the proposed method based on the correlation and strength with the generalization bound.

Related work

Network architectures with feature aggregation. Inceptions (Szegedy et al. 2015; Ioffe and Szegedy 2015; Szegedy et al. 2016; Szegedy, Ioffe, and Vanhoucke 2016) showed aggregating multiple features could further improve the performance. Veit, Wilber, and Belongie (2016) interpreted ResNet (He et al. 2016) as an ensemble of numerous shallow neural networks, resulting in learning various features intrinsically. Some operations including group convolution (Xie et al. 2017; Gao et al. 2019), channel attentions (Hu, Shen, and Sun 2018; Woo et al. 2018; Tan and Le 2019; Han et al. 2021a), and channel shuffle (Zhang et al. 2018b; Ma et al. 2018) led the network to have improved performance, which can be interpreted as learning diverse and less-correlated features in terms of architectural operation. Inspired by Newell, Yang, and Deng (2016); Lin et al. (2017), many previous works (Yu et al. 2018; Sun et al. 2018; Ryu, Yang, and Lim 2018; Du et al. 2020; Liu et al. 2018; Tan, Pang, and Le 2020) proposed to design advanced architectures by aggregating multiple features. They heavily rely on multi-path connections with extra trainable layers as a head architecture. Albeit they showed outstanding task performance, the models are computationally heavy due to additional learnable parameters; the multiple paths may learn similar representations without explicit regularization. Our work shares a similar concept of aggregating features, but the difference is that we leverage a lightweight design regime for sub-networks instead of a complicated head architecture for a strong prediction through the aggregation. Furthermore, it turns out that our lighter model consisting of the operations above achieves better discriminative powers with less correlated features.

Towards diverse and less-correlated features. Despite the architectural advances, it has been reported that learned features are usually in high correlation (Srivastava et al. 2014; Cogswell et al. 2015; Huang et al. 2018; Zhang, Zhang, and Li 2018; Ryu et al. 2019; Hua et al. 2021). Algorithmic ways of training the features having a low correlation are also addressed in the literature (Cogswell et al. 2015; Xiong et al. 2016; Gu et al. 2018; Zhu, Zhou, and Li 2018). Our method has, in a similar line to Cogswell et al. (2015); Zhu, Zhou, and Li (2018) which proposed distinctive losses that explicitly promote decorrelation at activation or filter, respectively. On the other hand, ours learn less-correlated features for aggregation in an inter-feature (or inter-layer) manner, directly affecting the final classifier. Lan, Zhu, and Gong (2018) initially promoted to ensemble branches by knowledge distillation, but the learned features were found to be highly correlated with each other. Finally, it also turns out that our proposed architecture cooperates with the proposed learning technique towards improving less-correlation property.

[1]Experimental results of the corresponding downstream tasks are found in the supplementary material.

Method

We start the section by first stating the motivation. Then we present the proposed learning method to learn less-correlated features and an efficient network architecture equipped with multiple sub-networks for the proposed learning method.

Motivation

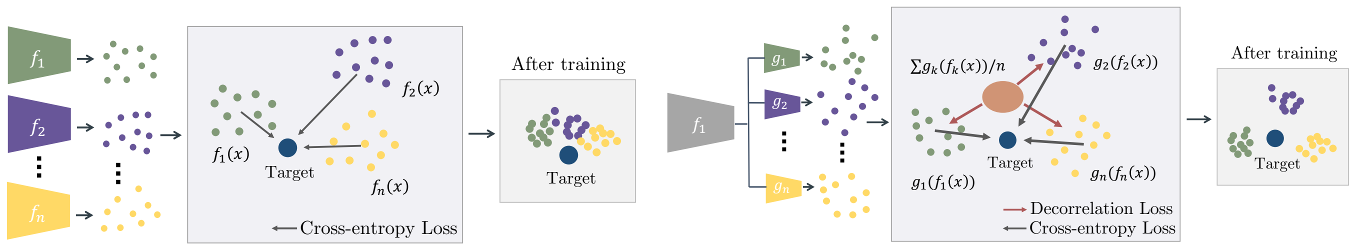
Our work is motivated by the literature (Breiman 2001; Ryu et al. 2019) that studied aggregating multiple features with a low correlation. A well-known theory (Breiman 2001) tells us if weak models are learned from different tasks to diversify the features, the resultant aggregated model is improved. The previous works (Lin et al. 2017; Tan, Pang, and Le 2020; Du et al. 2020; Lan, Zhu, and Gong 2018; Ryu, Yang, and Lim 2018) handle multiple features, but they underexplored the diversity and correlation among features, so we consider both strength and correlation (*i.e.*, the discriminative power and diversity) for features. We exclude learning with multiple or complicated networks but propose branched lightweight sub-networks following the theory above. We conjecture that 1) the correlation can be lowered by the effective learning method; 2) less-correlated representations can be further learned by the lightweight sub-networks.

Learning Less-correlated Features

Decorrelation loss. Modern network architectures (He et al. 2016; Tan and Le 2019) benefit from stable convergence in training, where the trained models have similar performance regardless of weight initialization (Ioffe and Szegedy 2015). However, in the case of training multiple identical network architectures or branches, this may not become a benefit. Since the models presumably converge to close local minima, the trained models are likely to learn correlated representations (see Fig.1a). In this light, we push the networks to learn distinct representations less-correlated. We begin with the model averaging loss (*i.e.*, equally weighting the outputs) with the l-th output prediction as

$$\mathcal{L} = \sum_{i}^{n} \mathtt{CE}_i = -\sum_{l}^{n} y^T \cdot \log f_i(x), \qquad (1)$$

where $\mathtt{CE}_l$ is the cross-entropy loss with the target ground-truth label y, and f_i is the output prediction of the input x



(a) **Traditional method.** Training multiple models individually or complicated heads in a supervised fashion and aggregating features.

(b) **Our method.** Training the network with sub-network heads with the less-correlated feature learning method.

Figure 1: **Schematic illustration of the proposed method.** (a) The individual features or features from complicated heads are trained using the task loss only; their output gets close to the ground-truth labels as training progresses; (b) since the decorrelation loss spreads out the prediction results of the sub-networks, the proposed learning technique lets the individual sub-networks learn features towards a less-correlated distribution. We argue the aggregated prediction is likely to be closer to the target, overcoming the intrinsic model bias. This is empirically proven in the experiment and visualized in Fig.3d and Fig.3f.

from each network. n is the total number of networks, and the final aggregated output is computed by the weighted summation of each network's output. Directly minimizing Eq. (1), the correlation among the predictions f_i is likely to be high, which is analogously stated in the literature (Srivastava et al. 2014; Cogswell et al. 2015; Ryu et al. 2019; Huang et al. 2018; Hua et al. 2021). We introduce a new regularizer to avoid it as follows:

$$\mathcal{L} = -\sum_{i}^{n} y^T \cdot \log g_i(\hat{x}) + \lambda \mathcal{L}_{dec}, \tag{2}$$

$$\text{s.t.} \quad \mathcal{L}_{dec} = \sum_{i}^{n}\sum_{j} \frac{g_j(\hat{x})^T}{n} \cdot \left(\log \sum_{k} \frac{g_k(\hat{x})}{n} - \log g_i(\hat{x}) \right), \tag{3}$$

where $\mathcal{L}_{dec}$ is coined by the decorrelation loss[2] (see Fig.1b), and λ is a tunable weighting hyper-parameter. Note that we replace the $f_i(x)$ in Eq.(1) with $g_i(\hat{x})$, where $\hat{x} = f_1(x)$ and use a negative value only for λ in Eq.(2) to realize the negatively-scaled function as the decorrelation loss.

Dissimilarity with knowledge distillation. One may argue that the proposed decorrelation loss relates to Kullback Leibler Divergence used in the knowledge distillations (Hinton, Vinyals, and Dean 2015; Romero et al. 2015). The canonical knowledge distillation methods use a positive value for λ. On the other hand, our method assigns a negative λ in Eq.(2), which lets the knowledge from the aggregated prediction be reversely transferred; therefore, each prediction g_i would deviate from the center of the aggregated prediction resulting in less-correlated. We argue that using knowledge distillation (*i.e.*, using the KD loss (Hinton, Vinyals, and Dean 2015)) to train individual student networks close to the aggregated prediction cannot achieve maximal performance improvement. This can be intuitively explained as the positive weighting in Eq.(2) makes the distance between the aggregated prediction and each prediction get closer, so the predictions get similar, as shown in Fig.1(a).We provide the experimental backup for this claim by comparing the cases of positive and negative λ in Fig.2c and Fig.3e.

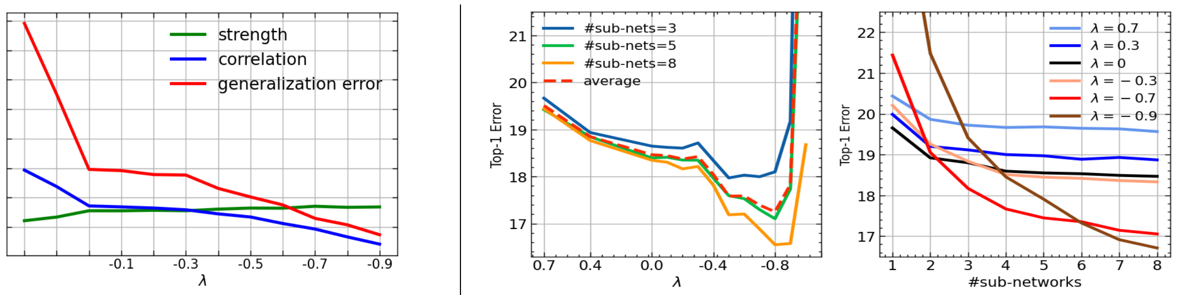
Towards Learning More Less-correlated Features

Network architecture. We propose a network architecture that promotes the learned features to have a low correlation. We aim to design the network with lightweight sub-networks refining the shared features to learn distinguishing representations. We aggregate the multiple features by similarly adopting the concept of feature aggregation networks (Lin et al. 2017; Yu et al. 2018) rather than solely using the final feature, which is to initially collect less-correlated features as the input feature (Zeiler and Fergus 2014).Therefore, we follow a simple way of extracting less-correlated representations from different stages (Yu et al. 2018; Sun et al. 2018; Ryu, Yang, and Lim 2018; Du et al. 2020; Liu et al. 2018; Tan, Pang, and Le 2020). However, we argue that naively aggregating the features by appending such heavy extra layers with many parameters is actually not required. Additionally, we entangle the features to calibrate the aggregated features through the proposed entanglement layer that further increases the feature diversity by learning to mix features. Its effectiveness in practice will be confirmed in Table 2.

Designing sub-network architecture. We alternatively propose an efficient replacement of the heavy heads (Sun et al. 2018; Lin et al. 2017; Liu et al. 2018; Tan, Pang, and Le 2020) by introducing lightweight but containing crucial elements for feature diversity in the sub-networks; we architecturally imbue a less-correlation property to each sub-network training. We use the 1×1 group convolution (Xie et al. 2017) as the first building layer. Additionally, the channel attention layer[3] (Woo et al. 2018) is adopted to calibrate feature maps with negligible extra costs. Note that we involve the channel shuffle (Zhang et al. 2018b) at the integrated features for further feature diversity. Finally, the feature processed with the bilinear pooling (Lin, RoyChowdhury, and Maji 2015;

[2]We use the term decorrelation here in the idiomatic sense of making the output predictions less relevant.

[3]Using other attention modules can be further investigated, but we do not focus here on searching for a better attention module.



(a) Generalization error bound (b) Error vs. λ (c) Error vs. #sub-networks

Figure 2: **Empirical studies with our proposed method.** (a) We compute the correlation (ρ) and strength (s) using Eq.(4) and Eq.(5), respectively, and the upper bound of the generalization error by $\rho(1-s^2)/s^2$. We visualize the measures versus the architectural elements and different λs in the deccorelation loss; (b) top-1 error versus λ in the deccorelation loss; (c) top-1 error versus the number of the sub-networks. The left side in (a) shows that our architecture contributes to lowering the generalization error bound, and λ in our less-correlated feature learning drops the bound on the right side. In (b) and (c), we observe training with the sub-networks with $\lambda < 0$ drops top-1 error.

$\mathcal{C}$	dim	Att	BP	$\mathcal{S}$	Dec	FLOPs	#Params	Top-1 Err. (%)
1	32	-	-	-	-	0.26G	2.55M	22.09
1	32	✓	-	-	-	0.26G	2.55M	21.82
1	32	✓	✓	-	-	0.26G	2.56M	21.16
1	64	✓	✓	-	-	0.35G	3.98M	19.82
2	64	✓	✓	-	-	0.26G	2.60M	20.14
2	128	✓	✓	-	-	0.35G	4.11M	18.39
4	128	✓	✓	-	-	0.26G	2.73M	18.74
8	128	✓	✓	-	-	0.22G	2.04M	18.92
8	128	✓	✓	✓	-	0.22G	2.04M	18.65
8	128	✓	✓	✓	✓	**0.22G**	**2.04M**	**16.88**

Table 1: **Factor analysis of sub-network's elements and decorrelation loss.** $\mathcal{C}$ and 'dim' denote the cardinality and channel dimension of the group convolution. We verify the impact of attention ('Att'), bilinear pooling ('BP'), and the channel shuffle ($\mathcal{S}$). We further evaluate the impact of training with the decorrelation loss ('Dec'). The experiments are performed with ResNet110 on CIFAR100.

Lin and Maji 2017)[4] for the following final classifier. One may consider that a simple single-convolutional head after the global average pooling at the features is sufficient, but we argue the sort of too-simple architecture (even with a heavy head) may not promote a less-correlation property to the features. We provide the empirical backups with the factor analyses, including the decorrelation loss on CIFAR in Table 1. Fig.1b shows the schematic illustration; we dub the proposed architecture LCFNet.

Analysis with Generalization Bound

We revisit the foundation theory of the generalization bound in Breiman (2001) which primitively investigates model averaging principles in terms of correlation and strength; random forest gets generalized when decision trees are strengthened individually and less correlated.

Strength and correlation. Inspired by the theoretical analysis using the degree of strength and correlation with the generalization bound (Ryu et al. 2019), we adopt the upper bound of generalization error to analyze our model. The strength is defined by the expectation $\mathbb{E}$ of the margin function between model prediction and ground truth label as

$$s = \mathbb{E}_{\mathbf{Y}_\phi, \hat{\mathbf{Y}}} P(\mathbf{Y}_\phi = \hat{\mathbf{Y}}) - P(\mathbf{Y}_\phi = \hat{\mathbf{J}}), \tag{4}$$

where $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ denote the output of a sub-network and the ground-truth labels of the data points, respectively. ϕ indicates a sub-network of our network, and $\hat{\mathbf{J}}$ denotes a set of the labels with the largest probability among non-correct answers. To compute the correlation ρ, we first define the raw margin function ψ as

$$\psi(\mathbf{Y}_\phi, \hat{\mathbf{Y}}) = \mathrm{I}(\mathbf{Y}_\phi = \hat{\mathbf{Y}}) - \mathrm{I}(\mathbf{Y}_\phi = \hat{\mathbf{J}}), \tag{5}$$

where $\mathrm{I}(\cdot)$ is the indicator function. The correlation ρ is then computed by averaging the Pearson correlation coefficient of ψ between all combinations of (ϕ_i, ϕ_j).

Generalization error bound. The generalization error bound γ is compute with the strength $\mathcal{S}$ and correlation ρ as

$$\gamma \leq \rho(1-s^2)/s^2. \tag{6}$$

As shown in Eq. (6), the correlation and strength are opposite concepts to achieve a low generalization error, but Ryu et al. (2019) showed that training to decrease the correlation results in increased strength. This can be considered evidence that introducing a less-correlation aspect in training improves performance. Our conjecture will be experimentally proven by verifying that the upper bound of generalization error is significantly reduced (see Fig. 2a). Furthermore, the correlation gets consistently lowered as appending the architectural elements and the degree of the decorrelation (adjusted by λ). We believe this result is because our network architecture with sub-networks trained with the decorrelation loss pushes the model to learn less-correlated and diversified features to improve the performance.

Experiment

In this section, we first provide the image classification results with an additional factor analysis of network design. The

[4]The bilinear pooling computes the channel-wise dot products as $\mathcal{B}(X,Y) = \sum_{l=1}^{w\times h} \mathbf{x}_l \mathbf{y}_l^{\mathbf{T}}$ that performs a dense feature association.

Elements	BP	Att	$\mathcal{S}$	Dec	Ent	SE/D	Epochs	#Sub-Net.	#Layers	#Channels	Top-1 Acc. (%)	Top-5 Acc. (%)
Baseline	-	-	-	-	-	-	50	5	(3,4,6,3)	1024	74.76	92.24
+ BP&Att	✓	✓	-	-	-	-	50	5	(3,4,6,3)	1024	76.64	93.10
+ Shuffle	✓	✓	✓	-	-	-	50	5	(3,4,6,3)	1024	76.95	93.16
+ Dec. loss	✓	✓	✓	✓	-	-	50	5	(3,4,6,3)	1024	77.75	93.43
+ Ent. layer	✓	✓	✓	✓	✓	-	50	5	(3,4,6,3)	1024	79.04	94.10
+ SE/ResNetD	✓	✓	✓	✓	✓	✓	50	5	(3,4,6,3)	1024	80.31	94.98
LCFNet-S	✓	✓	✓	✓	✓	✓	300	**3**	(3,4,6,3)	1024	**81.68**	**95.63**

Table 2: **Factor analysis on ImageNet.** BP, Att, $\mathcal{S}$, Dec, Ent, and SE/D denote the bilinear pooling, attention module, channel shuffle, training with the decorrelation loss, entanglement layer, and SE (Hu, Shen, and Sun 2018) with design tweaks (He et al. 2019), respectively. #Sub-Net., #Layers, and #Channels indicate the number of sub-networks, stage configuration, and the channel dimension of the last stage, respectively.

Network	#Sub-Net.	Input size	#Layers	#Channels	Throughput (img/sec)	FLOPs (G)	#Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
LCFNet-S	3	224	(3,4,6,3)	1024	1279	5.60	28.94	**81.68**	**95.63**
LCFNet-M	5	256	(3,4,6,3)	1024	740	7.35	33.47	**82.63**	**96.27**
LCFNet-L	5	288	(4,5,12,3)	1216	471	15.32	51.58	**83.55**	**96.58**

Table 3: **Specification of our models.** Based on LCFNet-S in Table 2, we scale it up to LCFNet-L by adjusting the design elements. Note that the configurations shown here are not searched exhaustively but are instantly tuned towards larger models, as provided in the model configurations that are not changed much from the baseline LCFNet-S.

scaled-up architectures are compared with the state-of-the-art network architectures on ImageNet. Second, we investigate whether our method actually learns less-correlated features and whether it relates to performance in further empirical studies.

ImageNet Classification

Architectural details. We build our model upon ResNet-50 with some tweaks; we reduce the channel dimension of the last three residual blocks to 1024 and exploit SE (Hu, Shen, and Sun 2018) and design tweaks in He et al. (2019). As shown in Table 1, bilinear pooling, attention module, channel shuffling, and decorrelation loss are applied to our baseline model as shown in Table 2. The resultant model is dubbed LCFNet-S. To augment our model, we increase 1) the channels of the ResNet backbone and entanglement layer to 1216; 2) the number of layers in the third group of the backbone to 12; 3) the input size as 228 $\times$ 228. We use five residual blocks in the entanglement layer with the identical channel dimension. We dub this augmented large network LCFNet-L and the other middle size network as LCFNet-M depending on the model size (see Table 3). Note that training runs very quickly (requires only 50 epochs); we believe this is another advantage of our design.

Factor analysis on sub-network design. We showed how each element works on CIFAR in Table 1, and we now extend the study to ImageNet. Table 2 fundamentally follows a similar accuracy trend to Table 1 as elements are added, and the effectiveness of the new elements of the entanglement layer and the SE/ResNetD is also observed. All the experiments are performed with identical network configurations, such as the stage configuration and channel dimension.

Comparison with state-of-the-arts on ImageNet. We compare our proposed models with the recent state-of-the-art network architectures in terms of accuracy, the number of parameters, and computational complexity. We compare the proposed LCFNet with recent network architectures, including the CNN architectures of ResNeSt (Zhang et al. 2020), TResNet (Ridnik et al. 2021), RegNet (Radosavovic et al. 2020); the ViT architectures of DeiT (Touvron et al. 2021), TNT (Han et al. 2021b), ConViT (d'Ascoli et al. 2021), PoolFormer (Yu et al. 2021), and XCiT (Ali et al. 2021). We systematically compare our entire models with the competing models grouped according to the computational budgets, mainly focusing on throughput. Table 4 shows that our models have clear advantages in throughput over their counterparts and outperform the competing networks, including TResNet and ViT-based models.

Applicability to other network architectures. We confirm how the proposed method works well with different backbones such as ResNeXt (Xie et al. 2017) and RegNetY (Radosavovic et al. 2020) using the same training setup. As shown in Table 5, the performance is greatly improved when applying our method. This result verifies the generalization ability of the proposed method, which we expect it further enhance the performance of other strong backbones.

Training Setups

ImageNet. Recent state-of-the-art networks (Zhang et al. 2020; Ridnik et al. 2021; Wightman, Touvron, and Jégou 2021b; Ali et al. 2021; Han et al. 2021b; Touvron et al. 2021) exploit similar training regimes with strong data augmentations, mostly based on the timm library[5] (Wightman, Tou-

[5]https://github.com/rwightman/pytorch-image-models/

Network	Image Size	Throughput (img/sec)	FLOPs (G)	#Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNeSt50-4s2x40d (Zhang et al. 2020)	224	861	4.42	30.42	81.11	95.56
TResNet-M (Ridnik et al. 2021)	224	1248	5.74	31.39	80.80	N/A
ResNeSt-50 (Zhang et al. 2020)	224	1040	5.42	27.48	81.13	N/A
ConViT-S (d'Ascoli et al. 2021)	224	1116	5.35	27.78	81.30	95.7
PoolFormer-S36 (Yu et al. 2021)	224	1072	4.99	30.86	81.40	N/A
LCFNet-S	224	**1279**	5.60	28.94	**81.68**	**95.63**
DeiT-B (Touvron et al. 2021)	224	874	16.85	86.57	81.80	N/A
PoolFormer-M36 (Yu et al. 2021)	224	743	8.79	56.17	82.10	N/A
TNT-S (Han et al. 2021b)	224	463	4.83	23.76	81.50	95.70
TResNet-L (Ridnik et al. 2021)	224	653	10.88	55.99	81.50	N/A
ConViT-B (d'Ascoli et al. 2021)	224	586	16.8	86.54	82.40	95.90
LCFNet-M	256	**740**	7.35	33.47	**82.63**	**96.27**
PoolFormer-M48 (Yu et al. 2021)	224	561	11.57	73.47	82.50	N/A
ResNeSt-101 (Zhang et al. 2020)	256	505	13.41	48.28	83.00	N/A
XCiT-L24 (Ali et al. 2021)	224	364	35.46	189.10	82.90	N/A
TNT-B (Han et al. 2021b)	224	360	13.40	65.41	82.90	96.30
DeiT-B (Touvron et al. 2021)	384	236	49.35	86.86	83.10	N/A
TResNet-M (Ridnik et al. 2021)	448	320	22.95	31.39	83.20	N/A
LCFNet-L	288	**471**	15.32	51.58	**83.55**	**96.58**
RegNetY-12GF (Radosavovic et al. 2020)	224	658	12.11	51.82	80.30	N/A
LCFNet-S†	224	**1279**	5.60	28.94	**81.19**	**95.55**
RegNetY-16GF (Radosavovic et al. 2020)	224	601	15.93	83.59	80.40	N/A
LCFNet-S/256†	256	**1015**	7.32	28.94	**81.82**	**95.74**
RegNetY-32GF (Radosavovic et al. 2020)	224	361	32.31	145.05	81.00	N/A
LCFNet-M†	256	**740**	7.35	33.47	**82.10**	**95.97**

Table 4: **ImageNet performance comparison.** Our models are compared with the competing SOTA networks on ImageNet. We group the networks according to the computational budgets they consume. Additionally, we match our training epochs for ours† to compare with RegNetYs trained for 100 epochs (Radosavovic et al. 2020). We report the accuracies of their original papers, except for ResNeSt50-4s2x40d, taken from (Wightman, Touvron, and Jégou 2021a). We measure the throughput by ourselves, running on an RTX 3090 GPU.

Network	FLOPs (G)	#Params (M)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNeXt	4.27	25.0	78.06	93.92
LCF-ResNeXt	5.84	30.1	**80.89**	**96.13**
RegNetY-800MF	0.80	6.3	74.17	91.81
LCF-RegNetY-800MF	0.97	9.8	**75.78**	**92.81**

Table 5: **More ImageNet results with our methods.** We apply our proposed sub-network design and learning technique to two renowned networks ResNeXt (Xie et al. 2017) and RegNet (Radosavovic et al. 2020). This is to show the applicability of our method, and the improvements are consistently observed.

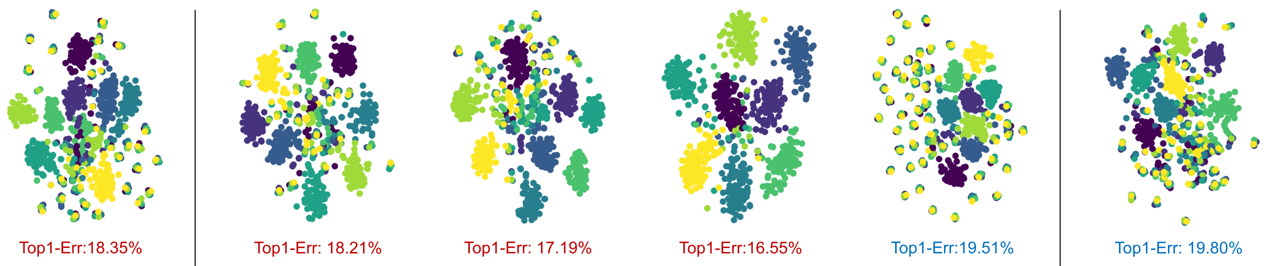
vron, and Jégou 2021a). We adopt a similar training regime, which employs Mixup (Zhang et al. 2018a), CutMix (Yun et al. 2019), and RandAugment (Cubuk et al. 2019) for data augmentation and use the cosine learning rate scheduling (Loshchilov and Hutter 2017) with 300 epochs[6]. We use 512 batch sizes in four GPUs for training.

[6]We primarily train our models for 50 epochs for the experiments in Table 2 and Table 5 except for RegNets. RegNets are trained for 100 epochs to compare by following the training setup in the original paper (Radosavovic et al. 2020).

CIFAR. We follow the standard 300-epochs training protocol with SGD (Han, Kim, and Kim 2017; Yun et al. 2019) with the initial learning rate of $1\text{e}{-3}$ decaying by 0.1 at 150 and 225 epochs. We use 64 batch sizes in two GPUs for training. We use ResNet110, and the aggregated feature inputs are reduced to 128 dimensions for each sub-network, and the cardinality for the group convolution is 8. The number of sub-networks is set to 8, and λL_{dec} in Eq.(3) is configured to -0.7 for the proposed method.

Further Empirical Studies

On learning less-correlated features. First, we investigate the impact of the decorrelation loss by visualizing the output features with t-SNE (Van der Maaten and Hinton 2008). Fig.3 shows the clear trend when using $\lambda < 0$; larger (in the negative direction) λ let the model learn less-correlated features; the performance follows the trend. Next, we validate the performance concerning λ in Eq.(2). As shown in Fig.2b, we achieve the best performance when the λ is near -0.7, and when $\lambda > 0$ (identical to the traditional KD loss), the performance is poorer than ($\lambda = 0$). Additionally, as shown in Fig.2c, when the decorrelation loss weight λ is -0.7 or -0.9, the performance improves significantly as the number of sub-networks increases. Interestingly, the performance is saturated trained only with three sub-networks without the



(a) $\lambda = 0$ (b) $\lambda = -0.3$ (c) $\lambda = -0.5$ (d) $\lambda = -0.8$ (e) $\lambda = 0.7$ (f) Model averaging

Figure 3: **t-SNE visualization of features**. We visualize how much the proposed learning method scatters the output features of each sub-network and the features of individually trained networks. We extract the features from the images in the validation set and distinguish them from different sub-networks by color. We use features of a ResNet110 for (a) to (e) and eight ResNet65s (to match the computational costs) for (f). Specifically, (a) eight sub-networks without the decorrelation loss ($\lambda = 0$); (b), (c), (d), and (e) different weighting parameters λ, respectively; (f) model averaging (*i.e.*, late-fusion) of eight individual features. We observe that 1) accuracy is aligned with the feature correlation; 2) our proposed learning method (*i.e.*, $\lambda > 0$) works to increase the feature diversity with lowered correlation; 3) individually trained networks yield the most correlated features as shown in (f); 4) a proper architectural design can improve this, compare (e) and (f); 5) our network with the less-correlation learning achieve lowered correlation, see (b), (c), and (d). When $\lambda = 0.7$, the features get more correlated.

decorrelation loss ($\lambda = 0$).

On sub-network architecture. Fig.3 manifests that features can be learned with a lowered correlation induced architecturally (similarly shown in Fig.2a). Comparing Fig.3a and Fig.3f shows that naively fusing the multiple outputs fails to avoid learning collapsed features. On the other hand, our model realizes less-correlation features without using the decorrelation loss, as shown in Fig.3a.

Comparison with multiple feature aggregation method. Here, we conduct additional experiments comparing a multiple feature aggregation method to show whether our method with the lightweight sub-network architecture works well over the method with heavy and complicated heads. Since such architectures (Lin et al. 2017; Du et al. 2020) aimed at different tasks, so for comparison, a milestone work (Lan, Zhu, and Gong 2018) that also uses multiple high-level branches is chosen. The branches look similar to our sub-networks; however, ONE-E uses a copy of fractions in a backbone network, making the overall model computationally heavy. ONE-E training is performed based on knowledge distillation to learn similar features at the branches of each other. We argue that the reported accuracy gain may stem from the heavy branches; it could learn highly expressively diversified features yet are highly correlated.

We use the identical architecture in the paper (Lan, Zhu, and Gong 2018) for training, which is in the publicly released codebase[7] with three branches from the middle layer of ResNets; ResNet32 (R32) and ResNet110 (R110) are used, which are the standard network architectures of the CIFAR training. We train the models for ONE-E and ours with the identical training setup for a fair comparison. Table 6 shows that our proposed method with the same number of sub-networks achieves better performance with less computational demands.

[7]https://github.com/lan1991xu/one_neurips2018

Method	FLOPs (M)	Params (M)	Top-1 Err. (%)	Top-5 Err. (%)
ONE-E (R32)	0.12	1.19	23.98	5.64
LCFNet (R32)	**0.08**	**0.75**	**21.85**	**5.14**
ONE-E (R110)	0.29	2.96	19.86	4.34
LCFNet (R110)	**0.22**	**2.04**	**19.03**	**3.94**

Table 6: **Comparison with ONE-E.** We perform an experimental comparison of our method with Lan, Zhu, and Gong (2018). Two baselines ResNet32 (R32) and ResNet110 (R110) are used, and ours consistently outperform the counterparts.

Conclusion

This paper introduces a new learning method with a network architecture that leverages lightweight sub-networks. Our learning method with the newly proposed decorrelation loss makes a network learn less-correlated features, which boosts performance due to the improved quality of the aggregated feature. Our proposed network architecture has been designed to diversify the features more when applying the proposed learning method. Additionally, unlike popular feature aggregation networks, our architecture aggregates intermediate features with the lightweight sub-networks, where the computational budget is significantly low. We have analyzed our proposed method's effectiveness based on the correlation and strength theory. We found that the generalization bound has been consistently reduced for each proposed element; the strength and correlation are increased and decreased, respectively. Finally, our network architecture has significantly outperformed the recent state-of-the-art CNNs and ViTs on ImageNet evaluation. Our proposed method can be applied to any network architecture to improve performance. We believe the proposed learning method and design paradigm facilitate future research.

References

Ali, A.; Touvron, H.; Caron, M.; Bojanowski, P.; Douze, M.; Joulin, A.; Laptev, I.; Neverova, N.; Synnaeve, G.; Verbeek, J.; et al. 2021. Xcit: Cross-covariance image transformers. In *NeurIPS*.

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*.

Breiman, L. 2001. Random forests. *Machine learning*, 45(1): 5–32.

Cogswell, M.; Ahmed, F.; Girshick, R.; Zitnick, L.; and Batra, D. 2015. Reducing overfitting in deep networks by decorrelating representations. *arXiv*.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. V. 2019. RandAugment: Practical data augmentation with no separate search. *arXiv*.

Du, X.; Lin, T.-Y.; Jin, P.; Ghiasi, G.; Tan, M.; Cui, Y.; Le, Q. V.; and Song, X. 2020. SpineNet: Learning scale-permuted backbone for recognition and localization. In *CVPR*.

d'Ascoli, S.; Touvron, H.; Leavitt, M. L.; Morcos, A. S.; Biroli, G.; and Sagun, L. 2021. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*.

Gao, S.; Cheng, M.-M.; Zhao, K.; Zhang, X.-Y.; Yang, M.-H.; and Torr, P. H. 2019. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gu, S.; Hou, Y.; Zhang, L.; and Zhang, Y. 2018. Regularizing Deep Neural Networks with an Ensemble-based Decorrelation Method. In *IJCAI*.

Han, D.; Kim, J.; and Kim, J. 2017. Deep Pyramidal Residual Networks. In *CVPR*.

Han, D.; Yun, S.; Heo, B.; and Yoo, Y. 2021a. Rethinking Channel Dimensions for Efficient Model Design. In *CVPR*.

Han, K.; Xiao, A.; Wu, E.; Guo, J.; Xu, C.; and Wang, Y. 2021b. Transformer in transformer. In *NeurIPS*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*.

He, T.; Zhang, Z.; Zhang, H.; Zhang, Z.; Xie, J.; and Li, M. 2019. Bag of tricks for image classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. In *CVPR*.

Hua, T.; Wang, W.; Xue, Z.; Ren, S.; Wang, Y.; and Zhao, H. 2021. On feature decorrelation in self-supervised learning. In *CVPR*.

Huang, L.; Yang, D.; Lang, B.; and Deng, J. 2018. Decorrelated batch normalization. In *CVPR*.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*.

Krause, J.; Deng, J.; Stark, M.; and Fei-Fei, L. 2013. Collecting a large-scale dataset of fine-grained cars. *In Second Workshop on Fine-Grained Visual Categorization*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. In *Tech Report*.

Lan, X.; Zhu, X.; and Gong, S. 2018. Knowledge distillation by on-the-fly native ensemble. In *NeurIPS*.

Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017. Feature pyramid networks for object detection. In *ICCV*.

Lin, T.-Y.; and Maji, S. 2017. Improved bilinear pooling with cnns. *arXiv*.

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 1449–1457.

Liu, S.; Qi, L.; Qin, H.; Shi, J.; and Jia, J. 2018. Path aggregation network for instance segmentation. In *CVPR*.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*.

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv*.

Newell, A.; Yang, K.; and Deng, J. 2016. Stacked hourglass networks for human pose estimation. In *ECCV*.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 722–729. IEEE.

Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; and Dollár, P. 2020. Designing network design spaces. In *CVPR*.

Ridnik, T.; Lawen, H.; Noy, A.; Ben Baruch, E.; Sharir, G.; and Friedman, I. 2021. Tresnet: High performance gpu-dedicated architecture. In *WACV*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. In *ICLR*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252.

Ryu, J.; Kwon, G.; Yang, M.-H.; and Lim, J. 2019. Generalized convolutional forest networks for domain generalization and visual recognition. In *ICLR*.

Ryu, J.; Yang, M.-H.; and Lim, J. 2018. DFT-based Transformation Invariant Pooling Layer for Visual Classification. In *ECCV*, 84–99.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15: 1929–1958.

Sun, S.; Pang, J.; Shi, J.; Yi, S.; and Ouyang, W. 2018. FishNet: A Versatile Backbone for Image, Region, and Pixel Level Prediction. In *NeurIPS*.

Szegedy, C.; Ioffe, S.; and Vanhoucke, V. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. In *ICLR Workshop*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

Tan, M.; and Le, Q. V. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv*.

Tan, M.; Pang, R.; and Le, Q. V. 2020. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10781–10790.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11).

Veit, A.; Wilber, M. J.; and Belongie, S. 2016. Residual networks behave like ensembles of relatively shallow networks. In *NeurIPS*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wightman, R.; Touvron, H.; and Jégou, H. 2021a. Resnet strikes back: An improved training procedure in timm. *Arxiv*.

Wightman, R.; Touvron, H.; and Jégou, H. 2021b. Resnet strikes back: An improved training procedure in timm. *Arxiv*.

Woo, S.; Park, J.; Lee, J.-Y.; and Kweon, I. S. 2018. Cbam: Convolutional block attention module. In *ECCV*.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated Residual Transformations for Deep Neural Networks. In *CVPR*.

Xiong, W.; Du, B.; Zhang, L.; Hu, R.; and Tao, D. 2016. Regularizing deep convolutional neural networks with a structured decorrelation constraint. In *ICDM*.

Yu, F.; Wang, D.; Shelhamer, E.; and Darrell, T. 2018. Deep layer aggregation. In *CVPR*.

Yu, W.; Luo, M.; Zhou, P.; Si, C.; Zhou, Y.; Wang, X.; Feng, J.; and Yan, S. 2021. Metaformer is actually what you need for vision. *Arxiv*.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *ECCV*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018a. mixup: Beyond empirical risk minimization. In *ICLR*.

Zhang, H.; Wu, C.; Zhang, Z.; Zhu, Y.; Lin, H.; Zhang, Z.; Sun, Y.; He, T.; Mueller, J.; Manmatha, R.; et al. 2020. Resnest: Split-attention networks. *Arxiv*.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018b. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*.

Zhang, Z.; Zhang, Y.; and Li, Z. 2018. Removing the feature correlation effect of multiplicative noise. In *NeurIPS*, volume 31.

Zhu, X.; Zhou, W.; and Li, H. 2018. Improving Deep Neural Network Sparsity through Decorrelation Regularization. In *IJCAI*.