NOAH: A NEW HEAD STRUCTURE TO IMPROVE DEEP NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A modern deep neural network (DNN) for image classification typically consists of two parts: a backbone for feature extraction, and a head for feature encoding and class predication. We notice that the head structures of prevailing DNNs share a similar processing pipeline, exploiting global feature dependencies while disregarding local ones. Instead, this paper presents ***N**on-gl**O**bal **A**ttentive **H**ead* (NOAH), a simple and universal head structure, to improve the learning capacity of DNNs. NOAH relies on a novel form of attention dubbed *pairwise object category attention*, which models dense local-to-global feature dependencies via a concise association of feature split, interaction and aggregation operations. As a drop-in design, NOAH can replace existing heads of many DNNs, and meanwhile, maintains almost the same model size and similar model efficiency. We validate the efficacy of NOAH mainly on the large-scale ImageNet dataset with various DNN architectures that span convolutional neural networks, vision transformers and multi-layer perceptrons when training from scratch. Without bells and whistles, experiments show that: (a) NOAH can significantly boost the performance of lightweight DNNs, e.g., bringing $3.14\%|5.30\%|1.90\%$ top-1 accuracy improvement for MobileNetV2 $(0.5\times)$|Deit-Tiny $(0.5\times)$|gMLP-Tiny $(0.5\times)$; (b) NOAH can generalize well on relatively large DNNs, e.g., bringing $1.02\%|0.78\%|0.91\%$ top-1 accuracy improvement for ResNet50|Deit-Small|MLP-Mixer-Small [1]; (c) NOAH can still bring acceptable performance gains to large DNNs (having over 50 million parameters), e.g., $0.41\%|0.37\%|0.35\%$ top-1 accuracy improvement for ResNet152|Deit-Base|MLP-Mixer-Base. Besides, NOAH also retains its effectiveness in the aggressive training regime (e.g., a ResNet50 model with NOAH reaches 79.32% top-1 accuracy, yielding 0.88% gain) and other image classification tasks. Code is provided for results reproduction.

1 INTRODUCTION

Image classification, a central task in computer vision, has been actively studied for several decades. In the pre-deep learning era, the bag-of-words model based on hand-crafted features (Lowe, 1999; Dalal & Triggs, 2005; Perronnin et al., 2010; Carreira et al., 2012; Lazebnik et al., 2006) was commonly used. With the tremendous advances in deep learning, deep neural networks (DNNs) have become the predominant learning models to many vision problems, leading to a fundamental paradigm shift from hand-crafted feature designing to neural architecture designing. Modern DNN architectures for image classification are constructed with a de-facto engineering pipeline that decomposes the network body into two parts: a backbone for feature extraction, and a head for feature encoding and class predication. Both of them are essential to the performance of the resulting networks.

Along with substantial research efforts in the backbone engineering, current DNN architectures in general have evolved into three major categories, namely convolutional neural networks (CNNs), vision transformers (ViTs) and multi-layer perceptrons (MLPs), built on convolutional, self-attention and linear layers, respectively. Over the past decade, CNNs have been the go-to image classification models, which are known to have the inductive bias possessing the locality and translation equivariance across all convolutional layers. Early top performing CNN models, AlexNet (Krizhevsky et al.,

[1] Although these ViT and MLP models are named with a keyword "Small", they contain about 20 million of learning parameters (see Table 1, 2, 3), which is actually at a similar level to the model size of ResNet50.

2012) and VGG (Simonyan & Zisserman, 2015), use a parameter-intensive head consisting of a max pooling layer, three fully connected (FC) layers and a softmax classifier. GoogLeNet (Szegedy et al., 2015) replaces the max pooling layer by a global average pooling (GAP) layer based on NIN (Lin et al., 2014), and further shows that removing the first two FC layers does not incur accuracy drop yet enjoys significantly reduced model size. Consequently, subsequent CNNs (He et al., 2016; Huang et al., 2017; Xie et al., 2017; Zoph & Le, 2017; Radosavovic et al., 2020; Tan & Le, 2019) mostly follow the head design of GoogLeNet. In 2021, Dosovitskiy et al. (2021) directly applied a pure transformer, the dominant model in natural language processing (NLP), to image classification tasks, achieving promising performance compared to top CNNs. With much less inductive bias than CNNs, this ViT architecture adopts a patchify stem where the self-attention is directly computed within non-overlapping local image patches (i.e., visual tokens). Its head merely comprises an FC layer and a softmax classifier, and takes the representation of an extra class token (a learnable embedding vector) as the input to predict the classification output. The class token interacts with the visual tokens across all multihead self-attention (MSA) layers, resembling transformers in NLP applications (Vaswani et al., 2017; Devlin et al., 2019). Subsequent ViTs, such as DeiT (Touvron et al., 2021b) and PVT (Wang et al., 2021), use the same head structure. It is noteworthy that some contemporaneous works (Liu et al., 2021c; Zhai et al., 2022; Huang et al., 2021) select the head design of GoogLeNet and its follow-ups. Specifically, they apply a GAP layer over the feature maps from the last MSA layer to generate head input, achieving very similar accuracy but much better memory efficiency, compared to the use of the class token. In Table 2, we also validate this on ImageNet with Deit-Tiny architecture. The GAP based head shows 0.2% top-1 gain to the class-token, while the gain of our design is 2.13%. Recently, a type of conceptually more simple architectures, which is entirely built upon MLP layers repeatedly applied across either the channel-patch or patch-channel dimensions, has been presented (Melas-Kyriazi, 2021; Tolstikhin et al., 2021; Touvron et al., 2021a). MLPs retain the patchify stem of ViTs, but remove the self-attention component. Regarding the choice of head structure, they adopt the GAP-based design, as like (Liu et al., 2021c; Zhai et al., 2022; Huang et al., 2021). In summary, the above design instantiations indicate that the head structures of prevailing CNNs, ViTs and MLPs mostly share a similar processing pipeline, exploiting global feature dependencies while disregarding local ones. Such typed head structures are simple, and have proven to be easy to optimize. However, they maybe incapable of capturing rich class-specific cues as they coarsely process critical information about the spatial layout of local features. We conjecture this may limit the feature abstraction ability of image classification models and result in suboptimal performance, which will become even more serious for compact DNNs developed to adapt resource-constrained environments (Howard et al., 2017; Zhang et al., 2018b).

There exist many research works that are directly or indirectly related to designing a better classification head, when typically given a CNN backbone. They mainly focus on descriptive methods to pooling (Lin et al., 2015; Gao et al., 2016; 2019a; Ionescu et al., 2015; Li et al., 2018; Wang et al., 2020; Saeedan et al., 2018; Zhao & Snoek, 2021; Zhang, 2019; Rippel et al., 2015; Zhai et al., 2017; Gao et al., 2019b; Cui et al., 2017; Wang et al., 2018a; Zeiler & Fergus, 2013), multi-layer/multi-region feature aggregation (Kim et al., 2020; Islam et al., 2021; Gao & Zhou, 2021; Liu et al., 2015), parametric learnable embedding (Arandjelovic et al., 2016; Tang et al., 2016; Wang et al., 2017; 2019; Gou et al., 2018), and attentive decoder (Zhu et al., 2017; Liu et al., 2021b; Zhu & Wu, 2021). Generally, their performance either depends on careful tuning of basic hyperparameters (e.g., decay of learning rate) (Lin et al., 2015; Gao et al., 2016; Ionescu et al., 2015; Gou et al., 2018; Gao et al., 2019a; Wang et al., 2018a; 2020; Zhu & Wu, 2021; Liu et al., 2015) or customized regularization strategies such as second-order optimization (Ionescu et al., 2015; Li et al., 2018; Wang et al., 2017; 2019; Gou et al., 2018) and data/feature/knowledge augmentation (Islam et al., 2021; Zhu et al., 2017; Zhang, 2019; Zhao & Snoek, 2021; Kim et al., 2020). In addition, many of them suffer from heavy computational cost (Lin et al., 2015; Gao et al., 2016; Gou et al., 2018; Ionescu et al., 2015; Gao et al., 2019a; Wang et al., 2017; 2019; 2018a; Islam et al., 2021), and some are tailored to multi-label/fine-grained classification tasks with the region-based input (Gao & Zhou, 2021; Zhu et al., 2017; Liu et al., 2021b; Gou et al., 2018; Ionescu et al., 2015; Cui et al., 2017; Wang et al., 2017; 2019) or weakly supervised learning scenarios (Arandjelovic et al., 2016; Pu et al., 2022). Because of the above factors, these methods are hardly ever used in the DNN architecture engineering, to the best of our knowledge. Driven by this analysis, a critical and unexplored question arises:is it possible to develop a new, yet still simple and easy-to-optimize head alternative that can be generalized to various DNN architectures, for the improved image classification purpose, especially on large-scale datasets like ImageNet (Russakovsky et al., 2015)?

To explore this question, in this paper, we present ***N**on-gl**O**bal **A**ttentive **H**ead* (NOAH), which relies on a novel form of attention called *Pairwise Object Category Attention* (POCA) efficiently encoding dense spatial feature dependencies. When constructing NOAH, we learn POCAs at local to global scales by a neat association of feature split, interaction and aggregation operations, taking the feature maps from the last layer of a backbone as the input (for ViTs and MLPs, we remove the class token if exists, as like (Liu et al., 2021c; Zhai et al., 2022; Huang et al., 2021)). Specifically, we split the input into multiple non-overlapping feature groups containing the same number of channels, allowing NOAH to efficiently learn group-wise POCAs in parallel. Upon each feature group, we then take use of a POCA block, which starts from a pair of parallel linear embeddings followed by a mixing operator, to produce a tensor representation of the POCAs at a local scale. Finally, the tensors of local POCAs projected from all feature groups are aggregated into a global POCA vector via a simple summation along the spatial dimension, which is fed to a softmax classifier for estimating a class label per image.

We evaluate NOAH on ImageNet with various DNN architectures. Specifically, we apply NOAH to 10 CNN backbones, 8 ViT backbones and 8 MLP backbones, using up all of our computational resources to cover a relatively large range of model complexity. For these models, the number of learnable parameters (Params) ranges from 1.68 million to 86.86 million, and the number of multiply-adds (MAdds) ranges from 59.3 million to 17.58 billion. Despite its simplicity, NOAH attains promising performance in terms of both model accuracy and efficiency. In the standard from-scratch training regime, the top-1 gain by NOAH ranges from 0.35% to 5.30%, without bells and whistles. In the aggressive from-scratch training regime (Liu et al., 2022), NOAH reaches 79.32% top-1 accuracy with a ResNet50 model, yielding 0.88% gain. Besides, NOAH also retains its efficacy on other image classification benchmarks.

2 Related Work

Convolutional neural networks. The breakthrough in the ImageNet challenge 2012, made by AlexNet (Krizhevsky et al., 2012), ignited the resurgence of CNNs. A lot of subsequent works, e.g., VGG (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy et al., 2015), ResNet (He et al., 2016), DenseNet (Huang et al., 2017) and ResNeXt (Xie et al., 2017), focus on constructing more powerful CNN architectures by scaling up network depth or width. Besides, many lightweight CNNs are also presented to meet resource-constrained applications, for example, MobileNets (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019) and ShuffleNets (Zhang et al., 2018b; Ma et al., 2018). Unlike the aforementioned CNNs that are designed manually, automatic network design using neural architecture search has recently attracted great attention (Zoph & Le, 2017; Pham et al., 2018; Tan & Le, 2019; Real et al., 2019; Tan et al., 2019; Radosavovic et al., 2020).

Vision transformers. Along with the recent success of transformers (Devlin et al., 2019; Vaswani et al., 2017) in NLP, some works (Hu et al., 2018; Wang et al., 2018b; Chen et al., 2020; Carion et al., 2020) make explorations in combining convolution and self-attention for computer vision problems. Dosovitskiy et al. (2021) present the first clean ViT architecture for image classification, taking raw image patches as input. DeiT (Touvron et al., 2021b) uses a token based distillation strategy to boost the training of ViT models on ImageNet, with no external data. Many recent works extend popular architectural practices from CNNs to advance ViT designs, including PVT (Wang et al., 2021) and Swin (Liu et al., 2021c) using pyramid self-attention structures to construct a hierarchical representation, TNT (Han et al., 2021) and T2T (Yuan et al., 2021) using fine-grained patch partition strategies to strengthen the representation ability, and so forth (Touvron et al., 2021c; Zhai et al., 2022; Huang et al., 2021; Pan et al., 2021).

Multi-layer perceptrons. Although ViTs have become a strong alternative to CNNs, they typically suffer from the quadratic complexity of self-attention operations. Recently, several concurrent works, such as Mixer (Tolstikhin et al., 2021), ResMLP (Touvron et al., 2021a) and gMLP (Liu et al., 2021a), show that replacing self-attention operations by linear operations does not affect the training performance, resulting in a class of much simpler DNN architectures entirely built with MLP layers. Following MLP variants incorporate modifications like shift (Yu et al., 2022; Lian et al., 2021; Chen et al., 2021) or permutator (Hou et al., 2022) modules to strengthen feature communication.

Being a universal head design, our NOAH would be applicable to all these DNN architectures.

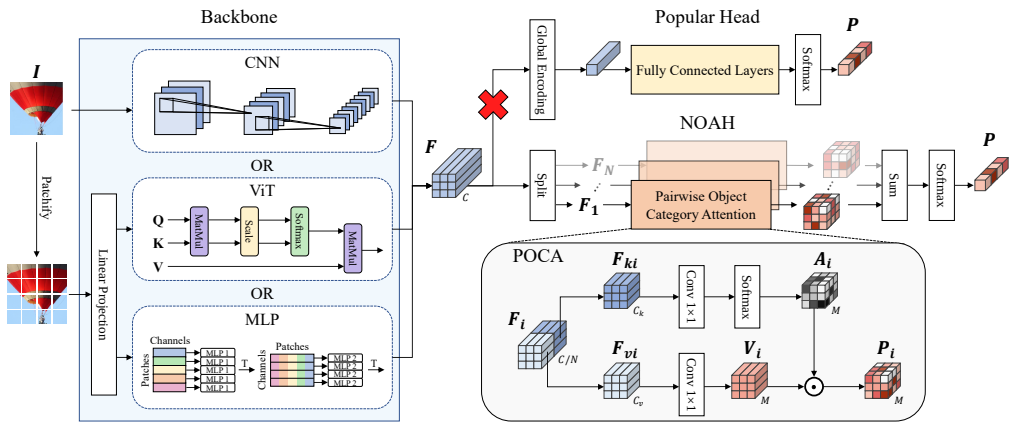

Figure 1: The macro-structure of DNNs with a ***N**on-gl**O**bal **A**ttentive **H**ead* (NOAH). Unlike popular heads using the global feature encoding, NOAH relies on *Pairwise Object Category Attentions* (POCAs) learnt at local to global scales via a neat association of feature split (two levels), interaction and aggregation operations, taking the feature maps from the last layer of a backbone as the input.

3 METHOD

Our basic goal is to design a simple, efficient and easy-to-optimize head alternative that can be used to improve various DNN models for image classification in a plug-and-play manner. An overview of our ***N**on-gl**O**bal **A**ttentive **H**ead*, abbreviated as NOAH, is shown in Figure 1.

Overall structure of non-global attentive head. NOAH is built upon a new form of attention called *Pairwise Object Category Attention* (POCA), which efficiently models dense spatial feature dependencies. In NOAH, we leverage a concise association of feature split (two levels), interaction and aggregation operations to learn POCAs at local to global scales in a group-wise manner. Given a backbone, let $\mathbf{F} \in \mathbb{R}^{H\times W\times C}$ be the feature maps from the last layer. For CNNs, H, W and C denote the channel height, width and number, respectively. For ViTs and MLPs, we remove the class token if exists, as like (Liu et al., 2021c; Zhai et al., 2022; Huang et al., 2021), and reshape the output feature sequence of the backbone into a square shape, then H and W denote the number of image patches in column and row, respectively. To ensure decent discrimination ability and efficiency of NOAH, we evenly split $\mathbf{F}$ into N non-overlapping feature groups $\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_N \in \mathbb{R}^{H\times W\times C/N}$ along the channel axis (this is called the first-level feature split), then parallelly apply N POCA blocks with the same structure to these N feature groups (one feature group fed to one POCA block). This results in N tensor representations $\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N \in \mathbb{R}^{H\times W\times M}$ of local POCAs, where M denotes the number of image classes. Next, they are directly aggregated into a global POCA vector $\mathbf{P} \in \mathbb{R}^{1\times 1\times M}$ via a simple summation of them along the spatial dimension ($NHW \rightarrow 1$), which is fed to a softmax classifier that will predict a class label for each image.

Formulation of pairwise object category attention. Clearly, the POCA block acts as a key component in NOAH. When formulating POCA, we mimic the notations of ViT (Dosovitskiy et al., 2021) for brevity (but it should be noted that POCA and self-attention have clear distinctions both in formulation and focus, which will be thoroughly discussed at the end of this section). Given the i^{th} feature group $\mathbf{F}_i \in \mathbb{R}^{H\times W\times C/N}, 1 \leq i \leq N$, we further split it into two disjoint subgroups $\mathbf{F}_{ki} \in \mathbb{R}^{H\times W\times C_k}$ and $\mathbf{F}_{vi} \in \mathbb{R}^{H\times W\times C_v}$ along the channel axis by a split ratio r, where $C_k = \lfloor rC/N \rfloor$ and $C_v = \lceil (1-r)C/N \rceil$. As a side benefit, this second-level feature split further improves the efficiency of NOAH, yet has slight effect on the model accuracy compared to the counterpart without it, as can seen from the results in Table 5. Then, we formulate POCA by taking use of a pair of parallel linear embeddings (termed "key" and "value" embeddings) and a mixing operator. The key embedding, composed of a 1×1 convolutional kernel $\mathbf{W}_{ki} \in \mathbb{R}^{1\times 1\times C_k\times M}$ along the channel dimension and a softmax activation function across the spatial dimension, which projects each pixel in $\mathbf{F}_{ki}$ to a desired image category dimension M, producing an "attention" tensor $\mathbf{A}_i \in \mathbb{R}^{H\times W\times M}$ that encodes dense position-specific object category attentions. The value embedding uses another 1×1 convolutional kernel $\mathbf{W}_{vi} \in \mathbb{R}^{1\times 1\times C_v\times M}$ to $\mathbf{F}_{vi}$, producing a "value" tensor $\mathbf{V}_i \in \mathbb{R}^{H\times W\times M}$ that maintains the same dimensions to the attention tensor $\mathbf{A}_i$. The final mixing

operator makes an interaction between the attention and value tensors via the Hadamard product, generating another tensor $\mathbf{P}_i \in \mathbb{R}^{H\times W\times M}$ capturing the POCAs at a local scale. Mathematically, the POCA block conditioned on the i^{th} feature group $\mathbf{F}_i$ can be written as:

$$\mathbf{A}_i = softmax(\mathbf{W}_{ki} * \mathbf{F}_{ki}), \quad \mathbf{V}_i = \mathbf{W}_{vi} * \mathbf{F}_{vi}, \quad \mathbf{P}_i = \mathbf{A}_i \odot \mathbf{V}_i, \tag{1}$$

where $*$ denotes the convolution operation, and $\odot$ denotes the Hadamard product.

Computational complexity of NOAH. For each POCA block, it has MC/N learnable parameters (Params) and requires $HWMC/N + HWM$ multiply-adds (MAdds) plus HWM Multiplications (for the Hadamard product), without considering the bias term. The summation and the softmax classifier require $HWMN$ Adds and M MAdds, respectively. In total, NOAH has MC Params and requires $HWMC + 2HWMN + M$ MAdds. Because of its simplicity, applying NOAH to replace existing heads of prevailing DNNs maintains almost the same model complexity in terms of both Params and MAdds in most cases, as can be seen from the results in Table 1, 2, 3.

Differences with the self-Attention. Although POCA in NOAH and self-attention in ViT share some similar notations, they are different both in formulation and focus: (1) POCA does not use a patchify stem and a "query" embedding, and does not need to compute the self-attention typically having a quadratic complexity; (2) In a POCA block, the attention and value embeddings process two disjoint feature subgroups separately, without a shared input used in the self-attention block; (3) Furthermore, for multiple POCA blocks run in parallel, there is also no feature sharing across them, in sharp contrast to the multihead self-attention (MSA) that shares the same input to all self-attention blocks in the same layer; (4) In order to encode dense position-specific object category attentions, the key and value embeddings of a POCA block project each pixel in their corresponding feature subgroups to a desired image category dimension M, and the resulting attention and value tensors are element-wisely mixed via the Hadamard product, which is another key distinction of POCA to the self-attention; (5) Besides the above differences in the attention formulation, our NOAH built upon POCA focuses on the classification head design but not the backbone design, and it can be used to improve the learning ability of different DNN architectures including CNNs,ViTs and MLPs.

4 EXPERIMENTS

In this section, we evaluate the performance of NOAH on image classification benchmarks, study the design of NOAH from different aspects, and explore its potentials in diverse training scenarios.

4.1 IMAGE CLASSIFICATION ON IMAGENET

Dataset, training setup and evaluation metric. Our main experiments are conducted on the popular ImageNet dataset (Russakovsky et al., 2015). It consists of over 1.2 million images for training and 50,000 images for validation, including 1,000 image classes. To have a comprehensive evaluation conditioned on the extreme capability of our current computational resources, we apply NOAH to a variety of DNN architectures including 10 CNN backbones, 8 ViT backbones and 8 MLP backones, covering a relatively large range of model complexity (see Table 1, 2, 3). For CNNs, we select backbones from ResNet (He et al., 2016), MobileNetV2 (Sandler et al., 2018), MobileNetV3 (Howard et al., 2019) and ShuffleNetV2 (Ma et al., 2018) families. For ViTs, we select backbones from DeiT (Touvron et al., 2021b) and PVT (Wang et al., 2021) families. For MLPs, we select backbones from Mixer (Tolstikhin et al., 2021) and gMLP (Liu et al., 2021a) families. In the experiments, we construct our networks by replacing the existing head of each selected DNN architecture by a NOAH. For NOAH, we set $N = \{4, 8\}$, $r = \{1/2, 1/4, 1/8\}$ for different typed DNNs following *an empirical principle*: the smaller the backbone size the larger the N, and the lager the C/N the smaller the r. We adopt the standard data augmentation to train and evaluate each network. For training, we first resize the input images to 256×256, then randomly sample 224×224 image crops or their horizontal flips. We standardize the cropped images with mean and variance per channel. For evaluation, we use the center crops of the resized images, and report top-1 and top-5 recognition rates on the ImageNet validation set. For fair comparisons, we use the public PyTorch codes of these networks with the exactly same settings to train all baseline models and our models from scratch. Our trained baseline models are either better than or at least on par with the reported ones. *Experimental details are put in the Appendix.*

Table 1: Results comparison on ImageNet with CNN backbones. For NOAH, we set: $N = 4$, $r = 1/2$ in ResNet18; $N = 4$, $r = 1/8$ in ResNet50, ResNet101 and ResNet152; $N = 8$, $r = 1/4$ in MobileNetV2 family, MobileNetV3-Small and ShuffleNetV2.

Network	Params	MAdds	Top-1(%)	Top-5(%)
ResNet18	11.69M	1.81G	70.25	89.38
+ NOAH	11.70M	1.84G	71.81 (↑**1.56**)	90.18 (↑**0.80**)
ResNet50	25.56M	3.86G	76.23	93.01
+ NOAH	25.56M	3.96G	77.25 (↑**1.02**)	93.65 (↑**0.64**)
ResNet101	44.55M	7.57G	77.41	93.67
+ NOAH	44.56M	7.67G	78.22 (↑**0.81**)	94.13 (↑**0.46**)
ResNet152	60.19M	11.28G	78.16	94.06
+ NOAH	60.20M	11.38G	78.57 (↑**0.41**)	94.36 (↑**0.30**)
MobileNetV2 (1.0×)	3.50M	300.8M	72.02	90.43
+ NOAH	3.52M	363.0M	73.35 (↑**1.33**)	91.13 (↑**0.70**)
MobileNetV2 (0.75×)	2.64M	209.1M	69.65	88.99
+ NOAH	2.65M	271.3M	71.44 (↑**1.79**)	89.87 (↑**0.88**)
MobileNetV2 (0.5×)	1.97M	97.1M	64.30	85.21
+ NOAH	1.98M	159.4M	67.44 (↑**3.14**)	87.11 (↑**1.90**)
MobileNetV2 (0.35×)	1.68M	59.3M	59.62	81.79
+ NOAH	1.69M	121.5M	63.40 (↑**3.78**)	83.91 (↑**2.12**)
MobileNetV3-Small	2.94M	61.1M	67.11	87.36
+ NOAH	2.95M	158.7M	68.92 (↑**1.81**)	88.08 (↑**0.72**)
ShuffleNetV2 (1.0×)	2.28M	144.3M	69.43	88.81
+ NOAH	2.29M	194.2M	70.72 (↑**1.29**)	89.38 (↑**0.57**)

Main results on CNNs. Table 1 shows the results comparison on CNNs. Generally, we can see that our NOAH always achieves superior results on all of these CNN backbones than the standard head structures, maintaining almost the same model size. In terms of top-1 accuracy, we can observe: (1) NOAH can significantly boost the performance of lightweight CNNs (having less than 5 million of learnable parameters), e.g., bringing an absolute top-1 gain of $1.33 \sim 3.78\%$, 1.81% and 1.29% for MobileNetV2 family, MobileNetV3-Small and ShuffleNetV2 ($\times 1.0$), respectively; (2) NOAH can generalize well on relatively large CNNs (having about $10 \sim 45$ million of learnable parameters), improving ResNet50 and ResNet101 by a top-1 gain of 1.02% and 0.81%, respectively; (3) NOAH can still bring acceptable performance gains to large CNNs (having over 60 million parameters), e.g., 0.41% top-1 gain to ResNet152. *Note that the performance improvement is obtained under the condition of simply replacing the existing heads of these CNN backbones by the corresponding NOAHs, without bells and whistles both in training and evaluation*. To NOAH, the performance improvement gradually decreases when the network becomes deeper, larger and more powerful. This is a common experimental trend in deep learning, as large CNN backbones have many more parameters, and tend to have much better learning capacities, compared to smaller CNN backbones. Another thing we want to emphasize is that, only to extremely lightweight CNN backbones (not including ViTs and MLPs), NOAH introduces obviously more extra MAdds, e.g., $1.60\times$ extra MAdds for MobileNetV3-Small (the worst case among all our experiments). This is because they typically use depthwise separable convolutions which significantly reduce the convolutional cost in terms of MAdds. However, the extra runtime cost of NOAH to MobileNetV3-Small is just $0.2684\times$ on a single GPU and $0.2729\times$ on a single CPU core (see Table 10). We find that the MAdds index usually cannot well reflect the runtime speed of prevailing CNNs (also including ViTs and MLPs), which has already been validated by some recent works (Wang et al., 2022; Radosavovic et al., 2020).

Main results on ViTs. From the results in Table 2, we can see that the ViT models trained with NOAH show consistently higher accuracy than the baseline models, and the top-1 gain is in the range of $0.37\% \sim 5.30\%$. The performance trend of NOAH on ViTs is similar to that on CNNs (shown in Table 1). Compared to the baseline models, our models maintain almost the same model complexity in terms of both Params and MAdds. It is noteworthy that current ViT models are usually much larger than CNN models. For instance, the number of learnable parameters in DeiT-Small and PVT-Tiny is 22.06 million and 13.23 million, respectively, although they are named with the keyword "Small" and "Tiny". Besides, in light of MAdds, they are also not lightweight, compared to CNN counterparts. In order to better explore the generalization ability of NOAH to more lightweight ViTs, we use a uniform width multiplier $(0.75, 0.5)$ to scale down the number of feature channels in every building block of DeiT-Small and PVT-Tiny, resembling MobileNetV2 (Sandler et al., 2018).

Main results on MLPs. The results comparison on MLPs is given in Table 3. Again, we can find that the MLP models trained with NOAH always achieve better performance than the baseline models, maintaining almost the same model size and very similar MAdds. Similar to the results

Table 2: Results comparison on ImageNet with ViT backbones. We use a uniform width multiplier $(0.75, 0.5)$ to scale down DeiT-Tiny and PVT-Tiny, resembling MobileNetV2. For NOAH, we set $N = 4$, $r = 1/2$ in all models.

Network	Params	MAdds	Top-1(%)	Top-5(%)
DeiT-Base	86.86M	17.58G	81.85	95.59
+ NOAH	86.86M	17.63G	82.22 (↑**0.37**)	95.75 (↑**0.16**)
DeiT-Small	22.06M	4.60G	79.78	94.99
+ NOAH	22.06M	4.66G	80.56 (↑**0.78**)	95.39 (↑**0.40**)
DeiT-Tiny (1.0×)	5.72M	1.26G	72.16	91.30
+ GAP	5.72M	1.25G	72.36 (↑0.20)	91.33 (↑0.03)
+ NOAH	5.72M	1.29G	74.29 (↑**2.13**)	92.27 (↑**0.97**)
DeiT-Tiny (0.75×)	3.29M	0.75G	62.55	85.32
+ NOAH	3.30M	0.77G	66.64 (↑**4.09**)	87.79 (↑**2.47**)
DeiT-Tiny (0.5×)	1.53M	0.37G	51.36	76.79
+ NOAH	1.54M	0.38G	56.66 (↑**5.30**)	80.41 (↑**3.62**)
PVT-Tiny (1.0×)	13.23M	1.96G	75.10	92.41
+ NOAH	13.24M	1.98G	76.51 (↑**1.41**)	93.25 (↑**0.84**)
PVT-Tiny (0.75×)	7.62M	1.12G	71.81	90.35
+ NOAH	7.62M	1.14G	74.22 (↑**2.41**)	91.82 (↑**1.47**)
PVT-Tiny (0.5×)	3.54M	0.51G	65.33	86.61
+ NOAH	3.55M	0.52G	68.50 (↑**3.17**)	88.42 (↑**1.81**)

Table 3: Results comparison on ImageNet with MLP backbones. We use a uniform width multiplier $(0.75, 0.5)$ to scale down Mixer-Small and gMLP-Tiny, resembling MobileNetV2. For NOAH, we set $N = 4$, $r = 1/2$ in all models.

Network	Params	MAdds	Top-1(%)	Top-5(%)
Mixer-Base	59.88M	12.62G	77.14	93.02
+ NOAH	59.88M	12.77G	77.49 (↑**0.35**)	93.27 (↑**0.25**)
Mixer-Small (1.0×)	18.53M	3.78G	74.18	91.56
+ NOAH	18.54M	3.88G	75.09 (↑**0.91**)	92.22 (↑**0.66**)
Mixer-Small (0.75×)	10.75M	2.14G	71.13	90.07
+ NOAH	10.76M	2.22G	72.32 (↑**1.19**)	90.57 (↑**0.50**)
Mixer-Small (0.5×)	5.07M	0.97G	65.22	86.34
+ NOAH	5.08M	1.02G	66.81 (↑**1.59**)	87.07 (↑**0.73**)
gMLP-Small	19.42M	4.41G	79.65	94.70
+ NOAH	19.42M	4.46G	79.95 (↑**0.30**)	94.86 (↑**0.16**)
gMLP-Tiny (1.0×)	5.87M	1.34G	72.05	91.23
+ NOAH	5.87M	1.36G	73.39 (↑**1.34**)	91.81 (↑**0.58**)
gMLP-Tiny (0.75×)	3.91M	0.84G	65.95	87.19
+ NOAH	3.91M	0.86G	67.71 (↑**1.76**)	88.32 (↑**1.13**)
gMLP-Tiny (0.5×)	2.41M	0.45G	54.99	80.02
+ NOAH	2.41M	0.47G	56.89 (↑**1.90**)	81.00 (↑**0.98**)

Table 4: Ablation on ImageNet with ResNet50 backbone: comparison of NOAH under different settings of the key ratio r and the number of the POCA blocks N. Best results are bolded.

Network	r	N	Params	MAdds	Top-1(%)	Top-5(%)
ResNet50	-	-	25.56M	3.86G	76.23	93.01
+ NOAH	1/2	4	25.56M	3.96G	76.84 (↑0.61)	93.32 (↑0.31)
	1/4	4	25.56M	3.96G	77.07 (↑0.84)	93.52 (↑0.51)
	1/8	4	25.56M	3.96G	**77.25** (↑**1.02**)	**93.65** (↑**0.64**)
	1/8	1	25.56M	3.96G	76.50 (↑0.27)	93.21 (↑0.20)
	1/8	2	25.56M	3.96G	76.77 (↑0.54)	93.36 (↑0.35)
	1/8	4	25.56M	3.96G	**77.25** (↑**1.02**)	**93.65** (↑**0.64**)
	1/8	8	25.57M	3.96G	76.94 (↑0.71)	93.34 (↑0.33)

Table 5: Ablation on ImageNet with ResNet backbones: comparison of NOAH without vs. with feature split (FS). Best results are bolded.

Network	1^{st} FS	2^{nd} FS	Params	MAdds	Top-1(%)	Top-5(%)
ResNet101	-	-	44.55M	7.57G	77.41	93.67
+ NOAH	-	-	58.89M	8.37G	78.10 (↑0.69)	94.07 (↑0.40)
	✓	-	46.60M	7.77G	**78.48** (↑**1.07**)	**94.22** (↑**0.55**)
	✓	✓	44.56M	7.67G	78.22 (↑0.81)	94.13 (↑0.46)
ResNet50	-	-	25.56M	3.86G	76.23	93.01
+ NOAH	-	-	39.90M	4.66G	77.24 (↑1.01)	93.62 (↑0.61)
	✓	-	27.61M	4.06G	**77.48** (↑**1.25**)	**93.77** (↑**0.76**)
	✓	✓	25.56M	3.96G	77.25 (↑1.02)	93.65 (↑0.64)
ResNet18	-	-	11.69M	1.81G	70.25	89.38
+ NOAH	-	-	15.28M	2.01G	**73.08** (↑**2.83**)	**91.01** (↑**1.63**)
	✓	-	12.21M	1.86G	72.22 (↑1.97)	90.42 (↑1.04)
	✓	✓	11.70M	1.84G	71.81 (↑1.56)	90.18 (↑0.80)

on CNNs and ViTs, the top-1 accuracy improvement by NOAH is pronounced when the model size becomes smaller. Just like ViTs, current small MLPs, such as Mixer-Small and gMLP-Small, actually are not small or efficient, compared to CNN counterparts. We also apply the width scaling strategy to Mixer-Small and gMLP-Small besides DeiT-Small and PVT-Tiny, and use the resulting more lightweight variants to test the performance of NOAH to thin MLPs.

The aforementioned experiments well validate the effectiveness and the generalization ability of NOAH. The performance improvement by NOAH is attributed to the learnt pairwise object category attentions, which effectively model local-to-global feature dependencies in a group-wise manner.

4.2 Ablation studies

To have a deep analysis of our NOAH, we further provide a lot of ablative experiments, which are mostly performed on ImageNet, unless otherwise stated.

The selection of N and r. Recall that NOAH uses two levels of simple feature split to learn POCAs at local to global scales. The first-level feature split uses a hyper-parameter N to control the number of POCA blocks in NOAH, and the second-level feature split uses another hyper-parameter r to control the key ratio in each POCA block. Accordingly, our first set of ablative experiments is conducted for the selection of N and r. In the experiments, we use ResNet50 as the backbone, and heuristically set $N = 4$ to compare different r options first, and then choose the best r to compare different N options. From the results of Table 4, we can find that all settings improve the model accuracy while maintaining almost the same computational complexity. Comparatively, the setting of $N = 4$ and $r = 1/8$ is the best, so we choose it for ResNet50. For the other 25 backbones, we simply adjust this setting via an empirical principle: the smaller the backbone size the larger the N, and the lager the C/N the smaller the r, without tuning N and r network by network.

The role of the feature split. In Table 5, a comparison of NOAH based models trained without vs. with feature split operations is given. We can see that: (1) When removing two levels of feature split from the standard NOAH, it shows obviously better results on small ResNet18 backbone but slightly

Table 6: Ablation on ImageNet with ResNet18 backbone: comparison of the POCA block with a softmax function along different dimensions to obtain attention tensor A_i.

Network	Activation	Params	MAdds	Top-1(%)	Top-5(%)
ResNet18	-	11.69M	1.81G	70.25	89.38
+ NOAH	Along spatial (our design)	11.70M	1.84G	**71.81** (↑**1.56**)	**90.18** (↑**0.80**)
	Along channel	11.70M	1.84G	69.09 (↓1.16)	88.88 (↓0.50)

Table 7: Ablation on ImageNet with ResNet18 backbone: comparison of NOAH using different operators to aggregate all local POCA tensors into a global POCA vector.

Network	Method	Params	MAdds	Top-1(%)	Top-5(%)
ResNet18	-	11.69M	1.81G	70.25	89.38
+ NOAH	Sum (our design)	11.70M	1.84G	**71.81** (↑**1.56**)	**90.18** (↑**0.80**)
	Max pooling	11.70M	1.84G	67.73 (↓2.52)	87.72 (↓1.66)
	Average pooling	11.70M	1.84G	71.12 (↑0.87)	89.81 (↑0.43)

Table 8: Ablation on ImageNet with ResNet50 backbone: comparison of NOAH in the standard training regime vs. in the aggressive from-scratch training regime used in Liu et al. (2022).

Network	Params	MAdds	Top-1(%)	Top-5(%)
ResNet50	25.56M	3.86G	78.44	94.24
+ NOAH	25.56M	3.96G	79.17 (↑0.73)	94.51 (↑0.27)
+ NOAH (w/o 2^{nd} FS)	27.61M	4.06G	**79.32** (↑**0.88**)	**94.60** (↑**0.36**)
+ NOAH (w/ no FS)	39.90M	4.66G	79.00 (↑0.56)	94.33 (↑0.09)

Table 9: Ablation on person ReID dataset Market-1501 with ResNet50 backbone: comparison of NOAH in the from-scratch training regime and in the fine-tuning regime.

Network	Training from scratch		Fine-tuning	
	Top-1(%)	mAP(%)	Top-1(%)	mAP(%)
ResNet50-FC	91.5	78.2	92.2	79.9
+ NOAH	**93.5**(↑**2.0**)	**81.8**(↑**3.6**)	**93.7**(↑**1.5**)	**82.3**(↑**2.4**)

worse results on larger ResNet50/ResNet101 backbone. However, the model size is significantly increased compared to the baseline model, for example, the number of Params for ResNet50 with NOAH increases from 25.56 million to 39.9 million; (2) The standard NOAH with two levels of feature split can well balance model accuracy and efficiency; (3) Removing the second-level feature split from the standard NOAH always gets more accurate models with few extra Params and MAdds.

The importance of the softmax function and the summation. Within each POCA block, we use a softmax activation function along the spatial dimension to generate the attention tensor A_i, which is an important design of NOAH to learn POCAs at a local scale. Another natural choice is to apply the softmax along the channel dimension. In Table 6, we study these two choices on ImageNet using ResNet18 as the backbone. Surprisingly, the softmax along the channel dimension leads to 1.16% top-1 accuracy drop against the baseline, but our design brings 1.56% improvement. This suggests that explicitly modeling dense spatial feature dependencies by producing a spatial attention map for each image category is essential for the POCA block. Moreover, we use a summation operator along the spatial dimension to aggregate all local POCA tensors into a global POCA vector. Despite of its simplicity, this design is also important. We compare it with max pooling and average pooling in Table 7. Obviously, the performance gap between the summation and the max pooling even reaches to 4.08%. Our design also outperforms the average pooling by 0.69% top-1 gain.

Model training in the aggressive regime. Recent work (Liu et al., 2022) shows that, compared to the standard training regime, much more accurate models can be attained when properly using strong training augmentations like increasing training epochs by multiple times, a linear warmup with tens of epochs, a cosine weight decay, a combination of data augmentation methods, multiple regularization strategies to alleviate overfitting, etc. Table 8 studies the generalization ability of NOAH in this aggressive from-scratch model training regime. Experiments are performed on ImageNet with ResNet50. We can see that NOAH performs well in this aggressive training regime, no matter using feature split operations or not. With two levels of feature split, NOAH brings 0.73% top-1 gain to the baseline model. Without the second-level feature split, the ResNet50 model with NOAH reaches 79.32% top-1 accuracy, showing 0.88% gain. *Detailed training settings are put in the Appendix*.

Performance on other image classification benchmarks. To evaluate the generalization ability of NOAH, we next perform ablative experiments on the popular person re-identification dataset Market-1501 (Zheng et al., 2015), which contains 750 and 751 identities for training and testing, respectively. We adopt ResNet50-FC as the baseline, following the common settings on Market-1501. Specifically, an extra FC layer is appended after the global average pooling layer of ResNet50 first, then the output 512-D feature vector is used for person matching. We consider two training regimes: the standard from-scratch training and the fine-tuning. Table 9 summarizes the results, showing that NOAH consistently achieves large margins against the baseline models trained in these two different training regimes, in terms of both top-1 accuracy and mAP. Comparatively, NOAH shows better results with the fine-tuning than the from-scratch training. However, NOAH gets higher accuracy margins for the from-scratch training than the fine-tuning, and the margin gaps are 0.5% for top-1 accuracy (2.0% vs. 1.5%) and 1.2% for mAP (3.6% vs. 2.4%).

Table 10: Ablation on ImageNet with 26 different CNN, ViT and MLP backbones: runtime inference speed (frames per second) comparison of our models with NOAHs vs. the baseline models. All models are tested on an NVIDIA TITAN X GPU (with batch size 200) and a single core of Intel E5-2683 v3 CPU (with batch size 1), separately. The input image size is 224×224.

Network	Vanilla			NOAH			+ GPU(%)	+ CPU(%)
	Params	Speed on GPU	Speed on CPU	Params	Speed on GPU	Speed on CPU		
CNNs								
ResNet152	60.19M	275.08	2.15	60.20M	269.70	2.09	1.96	2.79
ResNet101	44.55M	398.98	3.20	44.56M	388.38	3.08	2.66	3.75
ResNet50	25.56M	674.44	5.46	25.56M	641.36	5.25	4.90	3.85
ResNet18	11.69M	2394.00	12.98	11.70M	2117.88	12.20	11.53	6.01
MobileNetV2 (1.0×)	3.50M	1522.62	17.38	3.52M	1361.38	16.06	10.59	7.59
MobileNetV2 (0.75×)	2.64M	1762.84	20.31	2.65M	1565.28	17.32	11.21	14.72
MobileNetV2 (0.5×)	1.97M	2829.84	28.43	1.98M	2361.18	22.88	16.56	19.52
MobileNetV2 (0.35×)	1.68M	3576.58	36.09	1.69M	2850.02	28.60	20.31	20.75
MobileNetV3-Small	2.94M	4178.04	48.84	2.95M	3056.74	35.51	26.84	27.29
ShuffleNetV2 (1.0×)	2.28M	4241.94	34.36	2.29M	3312.82	29.21	21.90	14.99
ViTs								
DeiT-Base	86.86M	172.59	1.64	86.86M	164.16	1.59	4.89	3.05
DeiT-Small	22.06M	525.86	5.71	22.06M	456.79	5.48	13.14	4.03
DeiT-Tiny (1.0×)	5.72M	1246.51	15.76	5.72M	1039.72	14.55	16.59	7.68
DeiT-Tiny (0.75×)	3.29M	1606.39	23.52	3.30M	1271.78	20.74	20.83	11.82
DeiT-Tiny (0.5×)	1.53M	2169.99	34.51	1.54M	1644.79	29.30	24.20	15.10
PVT-Tiny (1.0×)	13.23M	619.85	9.13	13.24M	603.67	8.82	2.61	3.40
PVT-Tiny (0.75×)	7.62M	804.17	14.35	7.62M	766.34	13.20	4.70	8.01
PVT-Tiny (0.5×)	3.54M	1088.50	23.75	3.55M	1035.93	20.47	4.83	13.81
MLPs								
Mixer-Base	59.88M	236.49	2.13	59.88M	229.55	2.05	2.94	3.76
Mixer-Small (1.0×)	18.53M	733.96	7.18	18.54M	674.54	6.81	8.10	5.15
Mixer-Small (0.75×)	10.75M	1050.20	11.25	10.76M	929.10	10.57	11.53	6.04
Mixer-Small (0.5×)	5.07M	2037.45	23.88	5.08M	1615.76	20.62	20.70	13.65
gMLP-Small	19.42M	381.12	5.57	19.42M	368.22	5.28	3.38	5.20
gMLP-Tiny (1.0×)	5.87M	776.58	13.65	5.87M	712.21	12.31	8.29	9.82
gMLP-Tiny (0.75×)	3.91M	987.63	19.33	3.91M	882.56	17.29	10.64	10.55
gMLP-Tiny (0.5×)	2.41M	1343.41	26.95	2.41M	1140.75	23.10	15.09	14.29

Inference speed. In the deep learning community, the Params and the MAdds are two popular indices to measure the model size and the model efficiency, respectively. However when deploying a well-trained model on real computational devices, the wall-clock time at inference is significantly more important than the MAdds index. We also perform comprehensive experiments on ImageNet to study the runtime inference speed of our method. Specifically, we use an NVIDIA TITAN X GPU (with batch size 200) and a single core of Intel E5-2683 v3 CPU (with batch size 1) to test and compare 26 pairs of the NOAH based model and the baseline model, including 10 CNN pairs, 8 ViT pairs and 8 MLP pairs. Detailed results are shown in Table 10. We can observe that: (1) both on GPU and CPU, the models trained with NOAH in general show a relatively slower runtime speed than the baselines, but the extra latency by NOAH is acceptable ($1.96\% \sim 26.84\%$ on GPU, and $2.79\% \sim 27.29\%$ on CPU) considering the accuracy improvement; (2) the extra latency by NOAH gradually increases when the network size becomes smaller; (3) similar speed trends are demonstrated on different CNN, ViT and MLP architectures. It is worth mentioning that all these results are obtained under the condition that the NOAH based models maintain almost the same model size to the respective baseline models (the number of Params ranges from 1.68 million to 86.86 million). In addition, we can also find that the MAdds index usually cannot well reflect the practical latency of prevailing DNN models, which has already been validated on CNNs by some recent works (Wang et al., 2022; Radosavovic et al., 2020).

More analysis. *Please note that in the Appendix* we provide more ablative experiments to study NOAH from other perspectives, including: (1) several variant attention designs for the POCA block; (2) some visualization results to illustrate the learnt attention feature A_i, value feature V_i and POCA feature P_i; (3) the stability of the model training process with vs. without NOAH; (4) comparisons of NOAH with some previous methods on ImageNet. The limitations of NOAH are also discussed.

5 Conclusion

In this paper, we explore the head structure designing to improve the representation learning capabilities of DNNs for image classification. NOAH, a simple and easy-to-optimize head alternative built upon dense pairwise object category attentions learnt at local to global scales, is presented. We show that NOAH can attain promising performance (in terms of both model accuracy and efficiency) on the ImageNet classification benchmark when plugging it into various DNN models including CNNs, ViTs and MLPs, in both standard and aggressive from-scratch training regimes. We hope NOAH would inspire the community to pay more attention to the head structure in future DNN architecture engineering and representation learning research.

REFERENCES

Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

Shoufa Chen, Enze Xie, Chongjian Ge, Runjian Chen, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. In *ICLR*, 2021.

Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Le Quoc V. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020.

Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *CVPR*, 2017.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Bin-Bin Gao and Hong-Yu Zhou. Learning to discover multi-class attentional regions for multi-label image recognition. *IEEE TIP*, 2021.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *CVPR*, 2016.

Zilin Gao, Jiangtao Xie, Qilong Wang, and Peihua Li. Global second-order pooling convolutional networks. In *CVPR*, 2019a.

Ziteng Gao, Limin Wang, and Gangshan Wu. Lip: Local importance-based pooling. In *ICCV*, 2019b.

Mengran Gou, Fei Xiong, Octavia Camps, and Mario Sznaier. Monet: Moments embedding network. In *CVPR*, 2018.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *NeurIPS*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *TPAMI*, 2022.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *ICCV*, 2019.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

Gao Huang, Yu Sun, Zhuang Liu, Sedra Daniel, and Weinberger Kilian Q. Deep networks with stochastic depth. In *ECCV*, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021.

Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *ICCV*, 2015.

Md Amirul Islam, Matthew Kowal, Sen Jia, Konstantinos G Derpanis, and Neil DB Bruce. Global pooling, more than meets the eye: Position information is encoded channel-wise in cnns. In *ICCV*, 2021.

Ildoo Kim, Woonhyuk Baek, and Sungwoong Kim. Spatially attentive output layer for image classification. In *CVPR*, 2020.

Takumi Kobayashi. Gaussian-based pooling for convolutional neural networks. In *NeurIPS*, 2019a.

Takumi Kobayashi. Global feature guided local pooling. In *ICCV*, 2019b.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.

Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *CVPR*, 2018.

Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. As-mlp: An axial shifted mlp architecture for vision. In *ICLR*, 2021.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *ICCV*, 2015.

Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. In *NeurIPS*, 2021a.

Lingqiao Liu, Chunhua Shen, and Anton Van Den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *CVPR*, 2015.

Shilong Liu, Lei Zhang, Xiao Yang, Hang Su, and Jun Zhu. Query2label: A simple transformer way to multi-label classification. *arXiv preprint arXiv:2107.10834*, 2021b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021c.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

David G Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.

Luke Melas-Kyriazi. Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet. *arXiv preprint arXiv:2105.02723*, 2021.

Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *ICCV*, 2021.

Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 1992.

Tao Pu, Tianshui Chen, Hefeng Wu, and Liang Lin. Semantic-aware representation blending for multi-label image recognition with partial labels. In *AAAI*, 2022.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *CVPR*, 2020.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. In *NIPS*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *CVPR*, 2018.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Shlens Jonathon, and Wojna Zbigniew. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

Peng Tang, Xinggang Wang, Baoguang Shi, Xiang Bai, Wenyu Liu, and Zhuowen Tu. Deep fishernet for object classification. *arXiv preprint arXiv: 1608.00182*, 2016.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021.

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021a.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021b.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Herve Jegou. Going deeper with image transformers. In *ICCV*, 2021c.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Linnan Wang, Chenhan Yu, Satish Salian, Slawomir Kierat, Szymon Migacz, and Alex Fit Florea. Gpunet: Searching the deployable convolution neural networks for gpus. In *CVPR*, 2022.

Qilong Wang, Peihua Li, and Lei Zhang. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *CVPR*, 2017.

Qilong Wang, Zilin Gao, Jiangtao Xie, Wangmeng Zuo, and Peihua Li. Global gated mixture of second-order pooling for improving deep convolutional neural networks. In *NeurIPS*, 2018a.

Qilong Wang, Peihua Li, Qinghua Hu, Pengfei Zhu, and Wangmeng Zuo. Deep global generalized gaussian networks. In *CVPR*, 2019.

Qilong Wang, Li Zhang, Banggu Wu, Dongwei Ren, Peihua Li, Wangmeng Zuo, and Qinghua Hu. What deep cnns benefit from global covariance pooling: an optimization perspective. In *CVPR*, 2020.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018b.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. S2-mlp: Spatial-shift mlp architecture for vision. In *WACV*, 2022.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

Sangdoo Yun, , Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

Matthew D Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogerio Feris. S3pool: Pooling with stochastic spatial sampling. In *CVPR*, 2017.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022.

Hongyi Zhang, Cisse Moustapha, N Dauphin Yann, and Lopez-Paz David. mixup: Beyond empirical risk minimization. In *ICLR*, 2018a.

Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018b.

Jiaojiao Zhao and Cees G. M. Snoek. Liftpool: Bidirectional convnet pooling. In *ICLR*, 2021.

Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *ICCV*, 2015.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020.

Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. In *CVPR*, 2017.

Ke Zhu and Jianxin Wu. Residual attention: A simple but effective method for multi-label recognition. In *ICCV*, 2021.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

A APPENDIX

A.1 DATASETS AND IMPLEMENTATION DETAILS

A.1.1 IMAGE CLASSIFICATION ON IMAGENET

Recall that our main experiments are conducted on the popular ImageNet dataset (Russakovsky et al., 2015). It consists of over 1.2 million images for training and 50,000 images for validation, including 1,000 image classes. To have a comprehensive evaluation conditioned on the extreme capability of our current computational resources, we apply NOAH to a variety of DNN architectures including 10 CNN backbones, 8 ViT backbones and 8 MLP backones, covering a relatively large range of model complexity (see Table 1,2,3 in the main manuscript). For CNNs, we select backbones from ResNet (He et al., 2016), MobileNetV2 (Sandler et al., 2018), MobileNetV3 (Howard et al., 2019) and ShuffleNetV2 (Ma et al., 2018) families. For ViTs, we select backbones from DeiT (Touvron et al., 2021b) and PVT (Wang et al., 2021) families. For MLPs, we select backbones from Mixer (Tolstikhin et al., 2021) and gMLP (Liu et al., 2021a) families. In the experiments, we construct our networks by replacing the existing head of each selected DNN architecture by a NOAH (in the main manuscript, we provide some ablative experiments to study the settings of NOAH). Typically, we adopt the standard data augmentation to train and evaluate each network, unless otherwise stated. For training, we first resize the input images to 256×256, then randomly sample 224×224 image crops or their horizontal flips. We standardize the cropped images with mean and variance per channel. For evaluation, we use the center crops of the resized images, and report top-1 and top-5 recognition rates on the ImageNet validation set. For fair comparisons, we use the public PyTorch codes of these networks [2] [3] [4] [5] with the exactly same settings to train all baseline models and our models from scratch. *Note that our trained baseline models are better than or at least on par with the reported ones*.

Specifically, the models of ResNet18, ResNet50, ResNet101, ResNet152, MobileNetV2 ($1.0\times$), MobileNetV2 ($0.75\times$), MobileNetV2 ($0.5\times$), MobileNetV2 ($0.35\times$), MobileNetV3-Small, ShuffleNetV2 ($1.0\times$), DeiT-Tiny ($1.0\times$), DeiT-Tiny ($0.75\times$), DeiT-Tiny ($0.5\times$), PVT-Tiny ($1.0\times$), PVT-Tiny ($0.75\times$), PVT-Tiny ($0.5\times$), Mixer-Small ($1.0\times$), Mixer-Small ($0.75\times$), Mixer-Small ($0.5\times$), gMLP-Tiny ($1.0\times$), gMLP-Tiny ($0.75\times$) and gMLP-Tiny ($0.5\times$) are trained on servers with 8 NVIDIA Titan X GPUs. The other models of DeiT-Base, DeiT-Small, Mixer-Base and gMLP-Small, which require much larger memory cost, are trained on servers with 8 NVIDIA Tesla V100 GPUs. Detailed training setups for different DNN backbones are as follows.

Training setup for ResNet models. The initial learning rate is set to 0.1 and decayed by a factor of 10 every 30 epochs. All models are trained by the stochastic gradient descent (SGD) optimizer for 100 epochs, with a batch size of 256, a weight decay of 0.0001 and a momentum of 0.9.

Training setup for MobileNet models. The initial learning rate is set to 0.05 and scheduled to arrive at zero with a cosine decaying strategy. All models are trained by the SGD optimizer for 150 epochs, with a batch size of 256, a weight decay of 0.00004 and a momentum of 0.9. Unlike the other backbones including the MobileNetV2 family, the original head of MobileNetV3-Small architecture has two fully connected (FC) layers after the global average pooling (GAP) layer. When constructing NOAH, we additionally insert one 1×1 convolutional layer ahead of our basic NOAH to match the parameter size of this head structure. This is the reason why NOAH introduces obviously more extra MAdds to MobileNetV3-Small backbone compared to the other DNN backbones (see Table 1 in the main manuscript).

Training setup for ShuffleNetV2 models. The initial learning rate is set to 0.5 and scheduled to arrive at zero linearly. All models are trained by the SGD optimizer for 240 epochs, with a batch size of 1024, a weight decay of 0.00004 and a momentum of 0.9.

Training setup for DeiT and PVT models. The initial learning rate is set to 0.0005 and scheduled to arrive at zero with a cosine decaying strategy. All models are trained by the AdamW optimizer (Loshchilov & Hutter, 2019) for 300 epochs, with a batch size of 1024, a weight decay of 0.05

[2] https://github.com/pytorch/vision/tree/main/torchvision/models

[3] https://github.com/facebookresearch/deit

[4] https://github.com/whai362/PVT

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

and a momentum of 0.9. Following Touvron et al. (2021b) and Wang et al. (2021), we use Label Smoothing (Szegedy et al., 2016), RandAugment (Cubuk et al., 2020), Random Erasing (Zhong et al., 2020), Mixup (Zhang et al., 2018a) and CutMix (Yun et al., 2019) during training.

Training setup for Mixer and gMLP models. The initial learning rate is set to 0.0007 and scheduled to arrive at zero with a cosine decaying strategy. All models are trained by the AdamW optimizer for 300 epochs, with a batch size of 1536, a weight decay of 0.067 and a momentum of 0.9. Following Tolstikhin et al. (2021) and Liu et al. (2021a), we use Label Smoothing, RandAugment, Random Erasing, Mixup and CutMix during training.

Training setup in the aggressive regime. Besides the aforementioned training settings, in our ablative experiments we also study the generalization ability of NOAH to a much more aggressive from-scratch training regime used by ConvNeXt (Liu et al., 2022). Based on the public code [6] with the default settings, we compare the training of ResNet50 on ImageNet with vs. without using NOAH. Specifically, we use the AdamW optimizer to train each model for 300 epochs with a learning rate of 0.004, a batch size of 4096 and a weight decay of 0.05. There is a 20-epoch linear warmup and a cosine learning rate decaying schedule afterward. For data augmentations, popular schemes including Mixup, CutMix, RandAugment and Random Erasing are used. Besides, Stochastic Depth (Huang et al., 2016), Label Smoothing, LayerScale (Touvron et al., 2021c) and Exponential Moving Average (Polyak & Juditsky, 1992) are also adopted to regularize the training process. The results of Table 8 in the main manuscript show that, in the best case, the ResNet50 model with NOAH reaches 79.32% top-1 accuracy, bringing 0.88% top-1 gain to the baseline.

A.1.2 Person re-identification on Market-1501

To evaluate the generalization ability of NOAH to other image classification tasks, we also perform ablative experiments (see Table 9 in the main manuscript) on the popular person re-identification dataset Market-1501 (Zheng et al., 2015), which contains 750 and 751 identities for training and testing, respectively. We adopt ResNet50-FC as the baseline, following the common settings on Market-1501. Specifically, an extra FC layer is appended after the GAP layer of ResNet50 first, then the output 512-D feature vector is used for person matching. We consider two training regimes: the standard from-scratch training and the fine-tuning.

From-scratch training: the initial learning rate is set to 0.065 and decayed by a factor of 10 at epoch 150, 225 and 300. All models are trained by the SGD optimizer for 350 epochs, with a batch size of 64, a weight decay of 0.0001 and a momentum of 0.9.

Fine-tuning: the initial learning rate is set to 0.0003 and decayed by a factor of 10 every 60 epochs. All models pre-trained on ImageNet are fine-tuned by the Amsgrad optimizer (in PyTorch) for 150 epochs, with a batch size of 64, a weight decay of 0.0001 and a momentum of 0.9.

All models are trained with a single NVIDIA Titan X GPU.

A.2 Variant designs for the POCA block

Note that NOAH relies on the *Pairwise Object Category Attentions* (POCAs) learnt at local to global scales. In the main manuscript, we provide several sets of ablative experiments to study our basic components for learning either local POCAs (see Table 4, 5, 6) or global POCAs (see Table 7). For a better understanding of our proposed attention mechanism for the POCA block, here we provide more ablative experiments on the ImageNet dataset with ResNet18 as the backbone. Specifically, in the experiments, we set $N = 4$ and $r = 1/2$, and compare our proposed attention design for the POCA block with the following 4 variant designs: (1) for the activation function, replacing the softmax by the sigmoid; (2) for the key embedding W_{ki} and the value embedding W_{vi}, replacing 1×1 convolutional kernels by 3×3 convolutional kernels; (3) for the key embedding W_{ki}, removing the second-level feature split, and generating a single (instead of 1000, that is, one unique spatial attention matrix per image category) spatial attention $\mathbf{A}_i \in \mathbb{R}^{H \times W \times 1}$ which is shared to the value tensor $\mathbf{V}_i \in \mathbb{R}^{H \times W \times M}$ along the image category dimension M; (4) adding an extra linear (1×1 convolutional) layer before the key embedding W_{ki}, and also adding an extra linear (1×1 convolutional) layer before the value embedding W_{ki}, whose output features have the same dimensions

[6] https://github.com/facebookresearch/ConvNeXt

Table 11: Results comparison of NOAH with different attention designs for the POCA block. Experiments are performed on ImageNet with ResNet18 as the backbone. In the experiments, we set $N = 4$ and $r = 1/2$, and compare our proposed POCA design with 4 variant designs including: (1) for the activation function, replacing the softmax by the sigmoid; (2) for the key embedding W_{ki} and the value embedding W_{vi}, replacing 1×1 convolutional kernels by 3×3 convolutional kernels; (3) for the key embedding W_{ki}, removing the second-level feature split, and generating a single (instead of 1000, that is, one unique spatial attention matrix per image category) spatial attention $\mathbf{A}_i \in \mathbb{R}^{H \times W \times 1}$ which is shared to the value tensor $\mathbf{V}_i \in \mathbb{R}^{H \times W \times M}$ along the image category dimension M; (4) adding an extra linear (1×1 convolutional) layer before the key embedding W_{ki}, and also adding an extra linear (1×1 convolutional) layer before the value embedding W_{ki}, whose output features have the same dimensions to the input features, respectively. Best results are bolded.

Network	Method	Params	MAdds	Top-1(%)	Top-5(%)
ResNet18	-	11.69M	1.81G	70.25	89.38
+ NOAH	The POCA block (our design)	11.70M	1.84G	**71.81** (↑**1.56**)	**90.18** (↑**0.80**)
	Replacing the softmax by the sigmoid	11.70M	1.84G	70.68 (↑0.43)	89.64 (↑0.26)
	Replacing 1×1 conv. kernels by 3×3 ones	15.79M	2.04G	71.57 (↑1.32)	90.05 (↑0.67)
	Sharing a single spatial attention to M image categories	11.69M	1.84G	70.26 (↑0.01)	89.41 (↑0.03)
	Adding an extra linear layer before W_{ki} and W_{vi}	11.73M	1.84G	71.28 (↑1.03)	89.97 (↑0.59)

to the input features, respectively. Table 11 shows the results, from which we can see that all these 5 attention designs for the POCA block can improve model accuracy. Comparatively, our proposed design is the best, bringing 1.56% top-1 accuracy improvement to the baseline while maintaining almost the same model complexity. Surprisingly, replacing 1×1 convolutional kernels by 3×3 convolutional kernels does not lead to improved model accuracy although it introduces more pixels (that encode a larger spatial field) to compute each local POCA.

A.3 VISUALIZATION RESULTS

Recall that our NOAH leverages a concise association of feature split (two levels), interaction and aggregation operations to learn local-to-global POCAs in a group-wise manner. To have a better understanding of this parallel POCA learning mechanism, it is necessary to study the learnt values of POCAs. To this end, we use the well-trained ResNet18/DeiT-Tiny model with NOAH (see Table 1/2 in the main manuscript) to analyze the learnt attention tensor $\mathbf{A}_i \in \mathbb{R}^{H \times W \times M}$, the learnt value tensor $\mathbf{V}_i \in \mathbb{R}^{H \times W \times M}$, and the learnt local POCA tensor $\mathbf{P}_i \in \mathbb{R}^{H \times W \times M}$ for each of 4 POCA blocks. Given any image sample in the ImageNet validation set, for visualization, we select the spatial attention/value channel (that corresponds to the ground truth image category, and is normalized into $[0, 1]/[-1, 1]$ for visualization) of A_i/V_i for each of 4 POCA blocks, and the summation output (that corresponds to the ground truth image category, and is normalized into $[-1, 1]$ for visualization) of the local POCA channels of P_i tensors generated by 4 POCA blocks, respectively. Illustrative results are shown in Fig. 2. Here, for the NOAH based ResNet18/DeiT-Tiny model, $N = 4$, $r = 1/2$, and $(H = 7, W = 7)/(H = 14, W = 14)$, respectively. We can observe that parallel POCA blocks tend to learn varying spatial object category attention distributions, which are complementary to each other by visualization examples, showing their capability to capture rich spatial context cues to some degree.

A.4 TRAINING STABILITY

Fig. 3 shows the training and validation accuracy curves of the ResNet18/ResNet50/MobileNetV2 $(1.0\times)$/MobileNetV2 $(0.5\times)$ models trained on the ImageNet dataset with the standard head vs. NOAH. We can see that the ResNet18/ResNet50/MobileNetV2 $(1.0\times)$/MobileNetV2 $(0.5\times)$ model with NOAH shows relatively high top-1 gains throughout the training process compared to the baseline, respectively.

A.5 COMPARISON OF NOAH WITH SOME PREVIOUS METHODS ON IMAGENET

As we discussed in the Introduction section of the main manuscript, there exist many research works that are directly or indirectly related to designing a better classification head, when typically given a CNN backbone. Due to different focuses (e.g., multi-label/fine-grained classification tasks with the

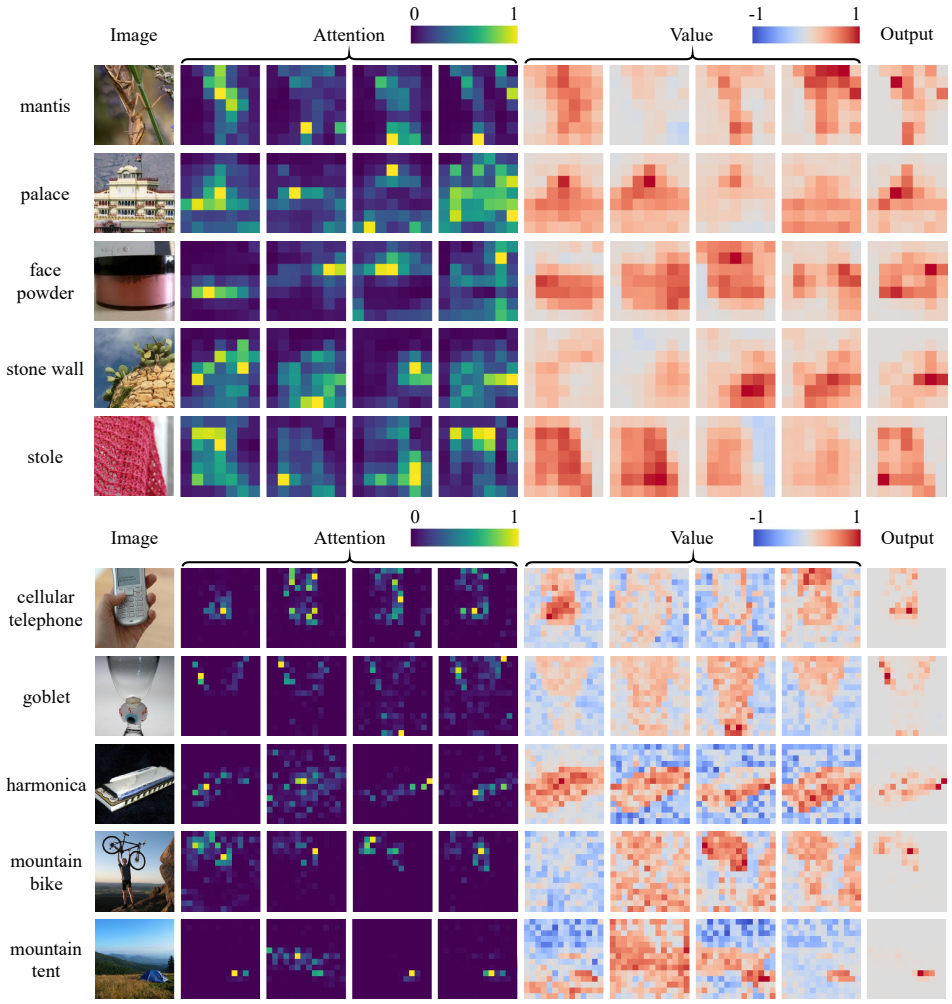

Figure 2: Illustrative visualization results. We use the well-trained ResNet18/DeiT-Tiny model with NOAH (see Table 1/2 in the main manuscript) to analyze the learnt attention tensor $\mathbf{A}_i \in \mathbb{R}^{H\times W\times M}$, the learnt value tensor $\mathbf{V}_i \in \mathbb{R}^{H\times W\times M}$, and the learnt local POCA tensor $\mathbf{P}_i \in \mathbb{R}^{H\times W\times M}$ for each of 4 POCA blocks. Given any image sample in the ImageNet validation set, for visualization, we select the spatial attention/value channel (that corresponds to the ground truth image category, and is normalized into $[0, 1]/[-1, 1]$ for visualization) of A_i/V_i for each of 4 POCA block, the summation output (that corresponds to the ground truth image category, and is normalized into $[-1, 1]$ for visualization) of the local POCA channels of P_i tensors generated by 4 POCA blocks, respectively. For the NOAH based ResNet18/DeiT-Tiny model, $N = 4$, $r = 1/2$, and $(H = 7, W = 7)/(H = 14, W = 14)$, respectively. These visualization results indicate that parallel POCA blocks tend to learn varying spatial attention distributions, which are complementary to each other by visualization examples, showing their capability to capture rich spatial context cues to some degree.

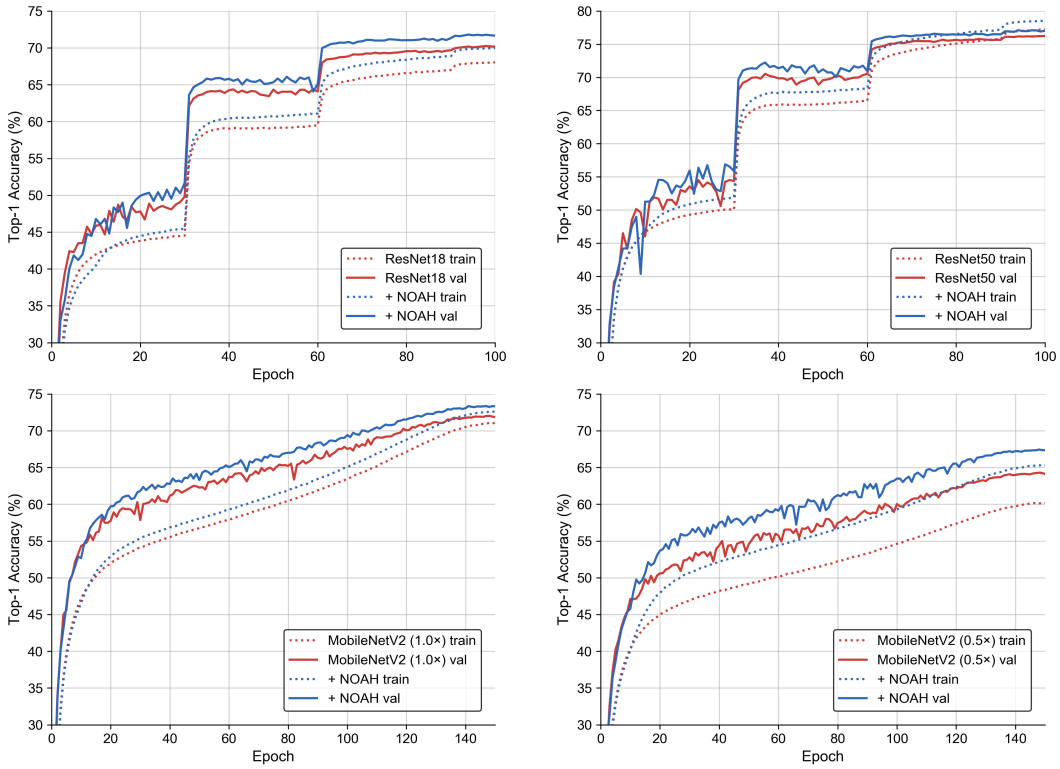

Figure 3: Curves of top-1 training accuracy (dashed line) and validation accuracy (solid line) of the ResNet18/ResNet50/MobileNetV2 $(1.0\times)$/MobileNetV2 $(0.5\times)$ models trained on the ImageNet dataset with the standard head vs. NOAH. Comparatively, the ResNet18/ResNet50/MobileNetV2 $(1.0\times)$/MobileNetV2 $(0.5\times)$ model with NOAH converges with the best validation accuracy, showing $1.56\%/1.02\%/1.33\%/3.14\%$ top-1 gain to the baseline while maintaining almost the same model size, respectively.

Table 12: Horizontal comparison of NOAH with some previous methods which report the results for ResNet50 on the ImageNet dataset. For NOAH, we set $N = 4$ and $r = 1/8$. The results for the reference methods are collected from the original papers. Typically, these methods differ in focus, training configuration and regularization strategy, and thus the apple-to-apple performance comparison is not applicable. Best results are bolded.

Network	Params	MAdds	Top-1(%)	Top-5(%)
ResNet50	25.56M	3.86G	76.23	93.01
GatedPool (AISTATS2016) (Lee et al., 2016)	NA	NA	77.73	93.67
MixedPool (AISTATS2016) (Lee et al., 2016)	NA	NA	77.19	93.47
DPP (CVPR2018) (Saeedan et al., 2018)	25.60M	6.59G	77.22	93.64
BlurPool (ICML2019) (Zhang, 2019)	NA	NA	77.04	NA
LIP (with Bottleneck-128) (ICCV2019) (Gao et al., 2019b)	24.70M	5.33G	78.19	93.96
LIP (with Bottleneck-256) (ICCV2019) (Gao et al., 2019b)	25.80M	7.61G	78.15	94.02
GFGP (ICCV2019) (Kobayashi, 2019b)	NA	NA	78.21	94.05
Half-GaussPool (NeurIPS2019) (Kobayashi, 2019a)	NA	NA	78.34	94.12
iSP-GaussPool (NeurIPS2019) (Kobayashi, 2019a)	NA	NA	78.63	94.32
GCP (CVPR2020) (Wang et al., 2020)	NA	NA	78.03	93.95
LiftDownPool (ICLR2021) (Zhao & Snoek, 2021)	NA	NA	77.64	93.89
3G-Net (CVPR2019) (Wang et al., 2019)	NA	NA	78.69	94.39
CSRA (ICCV2021) (Zhu & Wu, 2021)	NA	NA	75.70	NA
SAOL+KD (CVPR2020) (Kim et al., 2020)	NA	NA	77.11	93.59
SAOL+KD+Cutmix (CVPR2020) (Kim et al., 2020)	NA	NA	78.85	94.24
NOAH (standard from-scratch training)	25.56M	3.96G	77.25	93.65
NOAH (aggressive from-scratch training)	25.56M	3.96G	**79.17**	**94.51**

region-based input, or weakly supervised learning scenarios), training configurations (e.g., network architectures, decay of learning rate, or warm-up time) and regularization strategies (e.g., second-order optimization, data/feature augmentation, or knowledge distillation), the apple-to-apple performance comparison is not applicable. Here, we provide a horizontal comparison of our method with some previous methods which report the results for ResNet50 on the ImageNet dataset. Specifically, the reference methods mainly focus on pooling (GatedPool (Lee et al., 2016), MixedPool (Lee et al., 2016), DPP (Saeedan et al., 2018), BlurPool (Zhang, 2019), LIP (Gao et al., 2019b), GFGP (Kobayashi, 2019b), GaussPool (Kobayashi, 2019a), GCP (Wang et al., 2020) and LiftDownPool (Zhao & Snoek, 2021)), parametric learnable embedding (3G-Net (Wang et al., 2019)), and attentive decoder (SAOL (Kim et al., 2020) and CSRA (Zhu & Wu, 2021)). A brief horizontal performance comparison is shown in Table 12. We can see that NOAH achieves competitive performance although it differs with these methods in motivation (we attempt to develop a new yet still simple and easy-to-optimize head alternative that can be generalized to various DNN architectures, for the improved image classification purpose, especially on large-scale datasets like ImageNet).

A.6 LIMITATIONS OF NOAH

Clearly, the above experiments well validated that NOAH has favorable abilities to improve the representation learning of a variety of CNN, ViT and MLP architectures for image classification tasks. Despite of its simplicity and effectiveness, NOAH has two limitations. The major limitation lies in the generalization to dense downstream tasks, mainly due to different training paradigms. For example, when training a deep model for object detection, it typically starts with removing the head of a pre-trained classification network, then appends two new head structures (one for region-based classification, and the other for object localization) to the pre-trained backbone, and finally fine-tunes them on the given dataset while keeping either all layers or some particular layers of the pre-trained backbone fixed. Such a fundamental training paradigm difference makes NOAH cannot be directly used to dense downstream tasks. Our preliminary test of merely using the backbone pre-trained with NOAH to object detection dataset MS-COCO only brings negligible mAP gains. Besides, restricted by our current computational resources, the potential of applying NOAH to super-large DNN architectures (particularly ViTs and MLPs) is not explored. We hope we can explore it in the future, especially for training them with NOAH on a significant larger volume of training samples besides the ImageNet dataset.