COMNET: CORTICAL MODULE IS ALL YOU NEED

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing CNN architectures may achieve efficiencies in either one or two dimensions: FLOPs, depth, accuracy, representation power, latency but not all. In this work, we present a pragmatically designed novel CNN architecture "CoMNet" which offers multi-dimensional efficiencies at once such as: simple yet accurate, lower latency and FLOPs, high representation power in limited parameters, low memory consumption, negligible branching, smaller depths, and negligible design hyperparameters. The keys to achieve the multi-dimensional efficiencies are our biological underpinnings into CoMNet which is primarily the organization of cortical modules in the visual cortex, and borrowing concepts from well understood CNN designs. Our solid experimental evaluations demonstrate superiority of CoMNet over many state-of-the-art industry and academia dominant architectures such as ResNet, RepVGG etc. For instance, CoMNet supersedes ResNet-50 on ImageNet while being 50% shallower, 22% lesser parameters, 25% lower FLOPs and latency, and in 16% lesser training epochs. Code will be opensourced post the review process.

1 INTRODUCTION

To date, a number of successful CNN designs (19; 4; 6; 5; 16; 22; 17; 3) exist. Among them, the earlier CNNs are accuracy oriented designs which exploit multi-branch topology (20; 21), variable filter sizes (21), large depth (4), and feature reuse (6) to achieve their objective. These design choices however increases architecture complexity, limiting their adoption in resource constrained scenarios. The mobile CNNs (5; 16; 26; 12) on the other hand, mainly focus on reducing computations by employing depthwise separable convolution. As a drawback, these designs have smaller representation power which results in increased training time ($3\times$) to achieve accuracies comprable to fully convolution CNNs, and reduced performance on other downstream tasks. For instance, (5) requires 200 epochs on ImageNet to perform similar to (19) (trained for 75 epochs), but performs poor on object detection.

To match the performance with earlier designs (4), neural architecture search using reinforcement learning (28; 22) has also been explored. For instance, (22) by using compound scaling (depth, width, image-resolution), achieves higher accuracies in just a fraction of FLOPs and parameters to that of bigger CNNs (4). However such architectures are even more complex due to extreme branching and very large depths, making them slower due to GPU under utilization, as demonstrated in (3). Also, high resolution of input image is crucial in this architecture to obtain high accuracies which is in contrast to the designs (4; 19; 17) that are always trained for a smaller resolution 224×224.

To balance between both, more recently the work (3) aims at making VGG (19) faster, but accurate than deeper models (4) while being offering trade-offs to (22; 17). (3) decouples training and testing time networks via structural reparameterization, offering speeded-up inference phase. However, training-phase network consists of many parameters, many branches which increases its training time.

In the above remarkable CNN designs, the primary design dimensions such as achieving higher accuracy (4; 21), smaller FLOPs (5; 16; 22), and inference time FLOPs (3) are explored only individually. However, due to increased demands of CNNs in real-time applications such as autonomous driving, autonomous robotics require to target many other dimensions along with the above simultaneously.

One of them is latency or per-sample runtime which is not targetted in CNN designs, instead only FLOPs or throughput is focussed. The real-applications tend to process as few as one frame but not large batches. Second, due to the success of neural networks in variety of tasks, many algorithms are now being replaced by their neural network counterparts. It poses a challenge to run multiple networks on a single device (11), bringing in resource efficiency constraints. Third, rising complexity of the tasks (11; 9) demands large networks to achieve satisfactory results. Mobile series networks are not sufficient because of their aforementioned limited representation power.

Hence, in this work, we explore the possibility to achieve multidimensional efficiencies, if not in all then in many, while offering stronger trade-offs in the remaining. To achieve that, we turn towards biological underpinnings on cortical modules (14) which are predominently found across a primate cortex. In particular, we refer to the structure of cortical modules in ventral stream which is responsible of object recognition in primates.

We translate its design attributes in the context of CNN and develop a novel architecture "CoMNet", that offers multi-dimensional benefits such as lower complexity, smaller depth, hardware accelerators friendly, low memory consumption and memory access costs on parallel computing hardware, per-sample runtime, parameter efficiency, while outperforming sevral existing CNNs in many aspects. To the best of our knowledge, efficiency alltogether in many dimensions have not been explored as it is highly difficult task. Although CoMNet has some of its inspiration from biological studies, we do not claim on CoMNet's functional similarity with the visual cortex.

In a brief, the contributions of the paper can be summarized as follows: First, the notion of cortical modules in CNN. Second, the concept of one block - one stage. Third, Inter-stage Projections. We also present a rigorous experimental and ablation evaluation on CoMNet, and release several deployment ready CoMNet variants. write of quantitative achievements of CoMNet.

In the next section, we talk about the most relevant works and insights to the cortical modules. In Sec. 3, we describe the CoMNet architecture. In Sec. 4, we present a rigorous experimental analysis. Finally, in Sec. 5, we provide conclusions about the paper.

2 Related Work

2.1 Fully connected convolutions

The earlier CNN designs (8), (19), (20), (4) employ fully connected convolutions i.e. a kernel spans all the channels in the input. In general, a kernel with spatial size of 3×3 is quite popular. The deeper layers of these networks consist of a large number of channels e.g. 512 for (19), (4) to increase the representation power. However it also leads to too many connections in a kernel which becomes a predominant reason of overfitting. Dropout (8) is generally used to alleviate this issue but at the cost of increased training time. While keeping this number small decreases the representation power, thus limiting the accuracy. In order to avoid overly large parameters, (4) proposes 1×1 layers which reduces and expands the number of input channels before 3×3 convolutions. This strategy have also been employed in CNNs (16), (22), (17) and many more. It significantly reduces the number of parameters, however, still the 1×1 convolutions have a large number of connections per neuron.

2.2 Depthwise separable convolution

To reduce the computational density of the fully connected convolutional layer, the concept of depthwise separable convolutions (18) was proposed. In this type of convolution, a kernel is operated only on a single channel, resulting in networks with extremely smaller memory footprint, best suited for mobile applications. This convolution suffers from a major issue that single channel kernel operation can not take advantage of cross channel information which is necessary for achieving better performance. Therefore, a depthwise convolution is generally followed by a 1×1 convolution which combines the cross channel information. The architectures (5), (16), (22) are based on this concept.

Although depthwise convolution results in a huge reduction of computations and network footprint, it leads to drastically lower representation power, hampering the performance on large datasets. In order to achieve comparable performance, it becomes necessary to make them too deep and train for very large iterations e.g. (16) is trained for 300 epochs, (22) is trained for 365 epochs. In a real world setting, this much of longer duration are impractical for a resource constrained researcher.

2.3 Groupwise convolution

Groupwise convolution divides the input channels into groups. Therefore, such a convolution have access to more number of channels relative to the depthwise convolution and hence slightly more number of parameters can be obtained in a less number of layers. Groupwise convolution also suffers

with the issue of barred cross group channel communication. To remedy, (26), (12) propose channel shuffling which improves the performance.

2.4 INSIGHTS TO THE ABOVE DESIGNS

The advantage of (8), (19) lies in the fact that they run faster because of branch-less architecture and smaller number of layers. Whereas the architectures such as (4) have low theoretical FLOPs relative to (19) but runs slower. It is primarily due to high branching which leads to GPU under utilization. The mobile series network (22) are so deep and extremely complex that despite their theoretical FLOPs are very small relative to (4) and also accurate, but runs extremely slower. A recent work (3) addresses the issue but the improvements are only done for inference time. In addition, the network architecture is such that by scaling, the number of parameters in the final stage increases so large which requires stronger augmentation schemes (25), (1) and very large training duration.

The networks such as (16), (26), (12) all also suffers from this issue. They might have low theoretical FLOPs but might not run faster. In addition, smaller representation power becomes a bottleneck in such designs. They might obtain comparable accuracy on ImageNet (2) dataset but perform worse on other datasets. For example, (5) has roughly performs as accurate as one variant of (19) on ImageNet but performs poor on object detection (5).

2.5 BIOLOGICAL VISUAL CORTEX

The work (13) presents an experimental study on how the connections are being formed in the visual cortex of a cat's brain. The work claims that the visual cortex consists of columnar structures called 'cortical columns' which is a group of neurons forming micro-neural circuits. The column then extends deep down and forms a stack of layers of neuron along the column direction. A cortical column also consists of mini columns which contains roughly $70 - 100$ neurons. All neurons in a mini-column have a common input and a common output. The above model is also termed as "Cortical-Column" model of cerebral cortex.

Based on (13), the work (7) presents a cortex model known as Hubble-and-Weisel model which quickly became popular due to its remarkable findings. As per the model, the visual cortex is essentially a columnar and layered architecture which consists of ocular dominance columns. In such a column, the neuron responds to a particular stimuli in the visual scene and these ocular dominance columns are replicated across the visual field. In the cortex, multiple ocular dominance columns can receive input from a common part of retina while different ocular dominance columns might have same stimuli response. This organization helps in achieving massive parallalization.

As we go deep down the cortex, much more complex connections are seen to be formed (14) but exact topology still remains a mystery. Nonetheless, it has been established that the neurons in deeper stages respond to a complex stimuli such as a face of monkey, human, hand etc., while also contains the information of the stimuli from several viewpoints. The ensemble of such neurons is called a cortical module. The cortical modules rarely have horizontal connections except pyramidal neurons (23). The region which houses the cortical modules is particularly known as inferotemporal cortex which is responsible for object detection, recognition tasks. A primary observation from the model is that multiple cortical modules receives input from a common region of cortex.

In the earlier CNN designs, the concept of ocular dominance has been realized upto an extent via weight sharing. A kernel can be considered as a mini cortical column which is replicated across the whole spatial region of the input image. In biological cortex, multiple mini columns may have identical function in order to support parallalization as mentioned previously. However, in a CNN, only a single kernel is used to span the whole input spatial region. The concept of cortical modules however has never been seen in CNN. In this work, we focus on bringing cortical modules to CNNs and investigate its effect, advantages and drawbacks.

3 HIPERNET

In this section, we develop the architecture of HiPerNet. Since there are multiple concepts involved with HiPerNet, therefore we introduce them as incremental changes for better grasping of the topic.

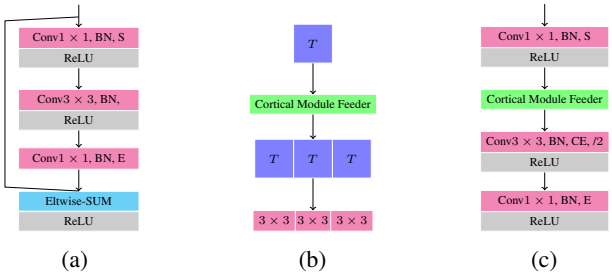

Figure 1: (a) Residual block, (b) cortical module entity, and (c) a neuronal cortical module block.

We start with the notion of cortical modules in the context of CNNs. Then, we introduce the concept of one block one stage and finally, we discuss the inter-stage projections for dense gradient flow.

3.1 Cortical modules in CNN

The work (23) best describes the cortical modules in a biological visual cortex. A cortical module receives input from preceding stages in the cortex and encodes the representations. Thousands of such modules are present in the inferotemporal cortex. Interestingly, many cortical modules may receive input from a common part of the cortex. This can be understood by the fact that at a given location in the visual field, there can be a monkey, a car or an apple. Therefore, a module which has higher similarities with a stimuli sends the signal to the other parts of the brain. How does it happen is yet unclear.

We mark this structural design quite important and observe that CNNs still do not exploit this attribute. In order to realize the above concept, we start with the residual bottleneck design of ResNet as shown in Fig. 1a. We choose this bottleneck because of its simplicity and popularity. We believe that it can provide better understanding of what we intend to develop. The residual bottleneck consists of three layers i.e. a 1×1 squeeze layer (S), a 3×3 context extraction layer (CE), and a 1×1 expand layer (E). The squeeze layer reduces the number of channels of an input tensor $\in \mathbb{R}^{C \times H \times W}$ to $\in \mathbb{R}^{C/4 \times H \times W}$. The factor of 4 is used in the ResNet. The expand layer reverses this operation while the context extraction layer keeps the number of channels same.

In order to bring the concept of cortical modules to CNNs, we define an operator which converts a tensor $\in \mathbb{R}^{C \times H \times W}$ to a tensor $\in \mathbb{R}^{(M \times C) \times H \times W}$, where M denotes the number of cortical modules. This operator takes the input tensor and replicates it along the channel dimension. In other words, the output tensor can be seen as M sliced tensors $\in \mathbb{R}^{C \times H \times W}$ which are identical (Figure 1b). This operator is implemented in form of a layer which is inserted after the squeeze layer (Figure 1c). We name the operator as cortical module feeder or CM feeder which is followed by a 3×3 context extraction layer. The CE layer has M groups which simultaneously operates on all the M slices of the input tensor. It extracts the context and sends the resulting tensor to the subsequent layers. It is important to note that the concept of groups can be found in literature such as (26), (12) etc. but they divide the input into a few groups. Contrarily, we follow the biological organization where input from previous stages in replicated and is sent to several cortical modules at once. This is the key difference between HiPerNet and the existing CNNs employing groupwise convolutions.

3.1.1 Spotlight

The proposed structure greatly mimics the organization of the cortical modules in the inferotemporal cortex or ventral stream (14). In our structure, the input tensor is first transformed by the CM feeder which is essentially a splitting operation in the visual cortex. In addition, a cortical module consists of a small number of neurons which are specific to a particular kind of stimuli (14). In the same way, we hypothesise that each group of the CE layer behaves as a cortical module which comprises of a very small number of neurons and can learn to respond a particular stimuli. The above multicolumn structure offers several benefits in the context of CNNs as discussed below.

Multiple smaller neurons vs one large neuron. First, it results in a limited number of synaptic connections per neuron. Therefore a single neuron has less choices to learn from the data which

improves generalization rather cramming. Here we build a hypothesis that multiple neurons with lesser synaptic connections are better than a single neuron with large number of connections. To understand, consider a case of visual stimuli which would require a definite amount of representation power or neurons. In the previous designs, a kernel operates on a large number of channels. Therefore during the learning phase it gets penalized for all the visual stimuli even it is uncorrelated with them. Moreover, it has also been demonstrated that for a given dataset and a large model, many weights turn out be very small which carry no effect on the accuracy rather a wastage of synaptic connections.

On the other hand in our hypothesis, we claim that separating the neurons help the network learn in a better way. Our proposed cortical module structure alleviates the issue of large neurons by organizing the neurons of a layer into modules while limiting the number of synaptic connection as found in a visual cortex. In other words, we can synthesize more number of neurons with sufficient synaptic connections in contrast to fewer neurons with large synaptic connection in a same number of parameters. Through our improvements, we can achieve higher speeds, lower parameters and better accuracy even in smaller training duration comparable to significantly larger models. We experimentally verify our claims in the experimental evaluation (Sec. 4, Table 4).

Explicit control over parameters. Second, it allows one to control the number of parameters in a kernel explicitly. It is quite customary to keep large number of channels in the deeper layers. However, as we increase the number of channels, the number of parameters also grow exponentially which becomes the primary cause of overfitting. For example, ResNet-50 (4), ResNet-101 (4) have only 3 blocks in the final stage (stage-5), and have 6 and 23 blocks respectively in stage-4. Interestingly, in ResNet-101, the three blocks of the final stage contribute towards $\sim 31\%$ of the network's parameters in comparison to the 23 blocks of the stage-4 which constitute the $\sim 68\%$. Based on our experiments, even if we reduce the number of blocks in the stage-4, the training error roughly remains the same but validation error increases. It shows that the larger number of connections in stage-5 are not helping much rather appearing as a potential reason towards overfitting. On contrary, by using the cortical module structure, the number of synaptic connections can be controlled in a well defined manner which in turn governs the number of parameters.

Computational density. Third, the cortical module structure offers huge computational density and high GPU utilization. It helps achieving lower FLOPs and latency simultaneously even at high accuracy which is not the case with previous models where if FLOPs are decreased, latency increases (4) or if latency is decreased accuracy drops (19). With the help of columnar organization, we need a smaller number of neurons to achieve similar level of accuracy. It must be noticed that when we talk about the accuracy, we mean to achieve that in short training epochs ($\sim 90 - 100$ as in (4)) in contrast to larger training schedules ($\sim 200 - 400$ as in (5), (16), (22), (26), (12)).

3.2 One block one stage

The earlier residual CNNs are organized as several stages, each consisting of repeated blocks (4), (16), (22). The network (4) consists of regular residual blocks whereas (16), (22) consist of inverted residual block. The residual blocks of a stage are connected via shortcut connections also known as residual connections so that gradients can flow without undergoing the vanishing phenomenon (4).

The above designs consist of only one context extraction or CE layer per block. The CE layer receives input from the squeeze layer and feeds the expand layer which increases the number of channels. The receptive field is primarily governed by the CE layer in contrast to the squeeze and expand layers because of their pointwise nature. Upon taking a closer look at the residual block structure (4), (16), it is observable that there are more number of point-wise convolutions then the CE layers. Moreover, the point-wise convolutions contribute to very large number of parameters relative the CE layer, and also out-number them. For example, the stage-5 in ResNet-50 consists of six pointwise layers while has only 3 CE layers. The CE layers have a total of $\sim$ 7M whereas the pointwise layers have $\sim$ 8M parameters.

We deeply account for these observations and look for better design possibilities. We aim at keeping the architecture complexity and computational demands very low while achieving higher receptive fields in smaller depths necessary for several task such as (27), (10). Towards our investigation, we come up with a novel idea of employing several the CE layers in one block instead of only one. This leads to serially connected CE layers. Now, in contrast to the multiple blocks per stage in (4), (16),

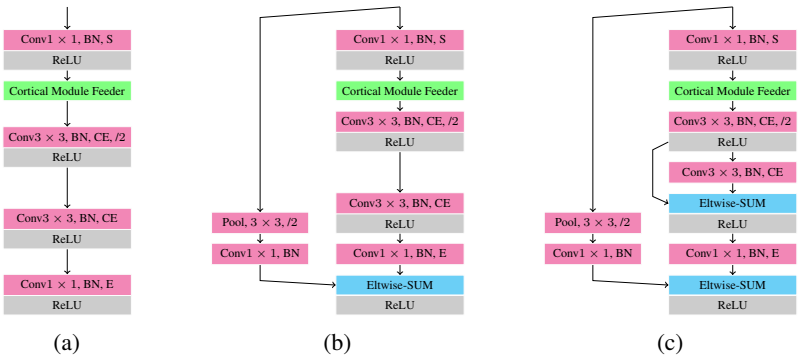

Figure 2: A HiPerNet stage/block (a) without inter-stage projections, (b) with projections, and (c) with residuals in CE layers. We name the stages as S0, S1, and S2 repectively for future reference.

(22), (26), (12) etc, we have only one block which can also be termed as a stage. The Figure 2a shows a HiPerNet stage with repeated CE layers or layered cortical module architecture.

In other words, if we have three residual blocks in ResNet, we have one block with three CE layers and two point convolutions which serve as squeeze and expand layers. The above strategy reduces the overall depth of the network while having lower latency since the number of serially connected layers have decreased. In order to construct a deeper network, we deploy multiples of blocks/stages which operates at different resolution. Note that we set only one block/stage for a given scale as the scale is reduced while proceeding down the network.

The proposed one-block-one-stage produces surprizing results which we didn't envisage but could only know after experimentation. Such network converges faster, requires smaller number of parameters, and layers to achieve a similar or better accuracy relative to deeper models. Also, as the number of layers decreases, latency also gets lowered. Primary reason behind these improvements is our belief that the 3×3 layers are more important than 1×1 because one can achieve bigger receptive fields in short depth. However using 3×3 suffers with overly large parameters causing overfitting for which squeeze and expand bottleneck was developed (4). In our case, we alleviate this issue by employing the cortical module structure along with one block one stage. As an example, if convert ResNet-50 to HiPerNet, we get reduction in 50% of layers. Definitely, number of parameters also decreases, yet we achieve better accuracy while having lower latency (Sec. 4, Table 6)

Furthermore, this strategy also aligns with the cortical modules developed previously. It is not yet known whether the cortical modules have layers in the cortex. However, we believe that there can be sequentially connected neurons which behave as a layered architecture. Our one block one stage helps achieving the above desired goal where each cortical module can have layers, helping in learning complex representation of a stimuli.

3.3 INTER-STAGE PROJECTIONS

The projections are widely used in ResNet (4) at the first residual block of each stage. It is essentially a 1×1 convolution which transforms the channel dimension of a feature map to facilitate a residual summation. Without a projection layer, the gradient will have to cross three layers of any block which might causes vanishing. On contrary, with a projection layer, gradients from deeper stages are passed to the shallower one by using only a single pointwise convolution with relatively lower diminishing.

We investigate the effect of using projections layers with an analogy that the gradient from deeper stage can be passed on to the shallower stages without diminishing. Hence, to incorporate projections in HiPerNet, we slightly modify the projection structure from (4) where the projection convolution has a stride of 2. In our design, we insert a pooling operation with stride 2 before the convolution. Figure 2a shows a HiPerNet stage without inter-stage projections while 2b, 2c incorporates projections.

It must be noted that, in (4), only the first block of any stage consists of a projection layer. However, since in our case, a block essentially is a stage, therefore, each stage consists of the projections. Although, there are networks such as (22) which do not use projection layers. However, due to

fundamental architectural modifications, it is necessary to study the impact of projections. Our findings suggest that inter-stage projections are extremely important (Table 3).

3.4 HiPerNet instantiation

While developing a CNN model, there can be a variety of choices such as number of channels in a layer, number of layers in a stage. Therefore, in order to not complicate the process, we keep our instantiation principles simple. To achieve that we follow the structural design of ResNet-50. We set total number of stages equal to five among which first is a plain 3×3 convolution with stride 2 while remaining are made up of cortical module structure (Figure 2). The channel dimensions of squeeze and expand layer are set same as the ResNet model i.e. stage-1 has squeeze output of 64 which gets multiplied by a factor of two at each stage. The reduction ratio is set to 4 so that the expand layer outputs 256 channels in the first stage.

In order to asses HiPerNet variants, we start with single cortical module in each stage. The number of CE layers in stage are set to total number of blocks in corresponding stage of ResNet-50. With the resulting model, we perform several ablations such as effect of projections, width of CE layers, residual connections inside the stage, structural reparameterization (3).

Then, we proceed to multiple cortical module setting in HiPerNet. The basic structure remains same as that of single module HiPerNet, however the number of cortical modules are increased with suitable selection of width of CE layers. We start with four cortical modules in the base model. Then we analyse the effect of increasing number of cortical modules as well as increasing the depth.

While instantiation, we keep in mind that the designs can cater real-time demands while not being falling short on representation power and achieving comparable accuracy to the larger model in lower parameter and computational budget, latency and shorter training times.

4 Experiments

In this section we present a rigorous experimental study of HiPerNet. We instantiate several models and present ablation study. For public release, we mainly focus on small models equivalent to ResNet-50. We present our study on ImageNet (2) dataset which is a standard benchmarking dataset for novel architectures. As a performance metric, we report Top-1 error rate.

One of our intention through this paper has been to develop architecture which can achieve high accuracy even in short training epochs ($90 - 100$) similar to ResNet which is trained for 120 epochs. We do not look for overly large epochs ($200 - 465$) as used in smaller models (5), (16), (26), (12), (22). In addition, we do not follow complex augmentation procedures such as (25), (24), (1) because they require longer training schedules.

We keep our training policy simple for reproducibility on commodity GPUs in contrast to models which are trained on thousands of GPUs (22). We train each HiPerNet variant to 100 epochs using `SGD` optimizer with `Nesterov` momentum. The `base_lr` is set to 0.1 which is decreased by 10 times every 30 epochs following (4). We use RandomResized crop (15), random flip augmentation only.

In the experimental discussion, we first present the ablation study of the various concepts in the HiPerNet design, and then present comparison with existing networks in terms of accuracy, parameter and FLOPs budget, Latency. We also present an analysis on the recent concept of structural reparameterization (3) during inference phase and show that HiPerNet is fully compatible with the concept and can take advantage of structural reparameterization during inference.

In the experiments, CE width refers to the number of channels in one cortical module, #CE refers to number of CE layers in a stage, #Blocks refers to number of blocks per stage, and #M refers to the number of cortical modules throughout the network, and #Depth denotes the network depth.

Table 1: Effect of residual connections in CE layers and varying the number of CE layers (#CE).

CE width	#CE	#Blocks	#M	#Depth	Projection	Pooled	Residual	Stage Type	#Params	#FLOPs	Top-1 (%)
$\{32, 64, 128, 256\}$	3 4 6 3	1 1 1 1	1	26	✓	✓	✗	$S1$	9.8M	1.44B	25.91
	3 4 6 3	1 1 1 1	1	26	✓	✓	✓	$S2$	9.8M	1.44B	25.89
	4 5 20 5	1 1 1 1	1	44	✓	✓	✗	$S1$	13.1M	1.96B	25.70
	4 5 20 5	1 1 1 1	1	44	✓	✓	✓	$S2$	13.1M	1.96B	25.00

Table 2: Effect of varying the width of CE layers (#CE).

CE width	#CE	#Blocks	#M	#Depth	Projection	Pooled	Residual	Stage Type	#Params	#FLOPs	Top-1 (%)
$\{32, 64, 128, 256\}$	3 4 6 3	1 1 1 1	1	26	✓	✓	✓	$S2$	9.8M	1.44B	25.89
$\{64, 128, 256, 512\}$	3 4 6 3	1 1 1 1	1	26	✓	✓	✓	$S2$	18.2M	2.81B	23.95

4.1 Ablation Study

4.1.1 Effect of residual connections in CE layers

Table 1 shows the corresponding analysis. It consists of two variants: a shallower and a deeper. The shallower model has as many #CE layers per stage as the number of blocks per stage in ResNet-50. For shallower model, residual connection shows only minor improvement however for deeper model the effect of residual connections in CE layers is significant. We have noticed that this gap could be bigger if we turn off the projection connections as shown in Table 3.

4.1.2 Varying the Number of CE layers (#CE)

Next we vary the number of CE layers per stage and analyse the effect. The metrics can be seen from Table 1. It can be observed that by increasing #CE per stage, the accuracy improves roughly by $\sim 1\%$ while increment in the FLOPs is negligible.

4.1.3 Varying the width of CE layers

Table 2 shows the corresponding analysis. We simply double the CE width and observe that accuracy increases by $\sim 2\%$. It is interesting to note that the doubled variant reaches the accuracy of ResNet-50 (23.8%) in lesser resources (discussed in state-of-the art comparison).

4.1.4 Effect of inter-stage projections

In order to analyze the effect of inter-stage projections, we first train the models without projections. Then we conduct two experiments with projections inserted. In one, we use pooling with stride factor of two for reducing the spatial resolution before the 1×1 pointwise convolution and in another, we remove the pooling and run the pointwise convolution with stride 2 as used in (4).

Table 3 shows the corresponding analysis. It can be noticed that without projections, the model suffers with heavy accuracy loss of $\sim 2\%$ relative to when projection is used. Further, removing pooling before the pointwise convolution layer reduces the accuracy by $\sim 1.10\%$. It happens because the pointwise convolution operates only at a single location in the spatial resolution and at a stride of 2, the convolution operates at sparse location thereby sparse gradient are sent during training phase, leading to decreased accuracy. On the otherhand, the pooling operation with 3×3 summarizes the neighbood per channel and then pointwise convolution encodes the cross channel information. It results in better accuracy because in this case dense gradients are sent backward.

4.1.5 Multiple cortical modules

Table 4 shows the analysis of HiPerNet variants with single, and multiple cortical modules with varying width and varying groups. It can be seen that the smaller model (Row-2) with multiple modules performs only $\sim 1\%$ less accurate even at 33% lower parameters and 37% lower FLOPs than the single module variant (Row-1). While the wider variant (Row-3) performs better than the model with single module roughly by 0.40% in similar number of parameters and FLOPs. It shows the significance of multiple cortical modules. For future reference, we name the five models in the table as HiPerNet-A0-M1, HiPerNet-A1-M4, HiPerNet-A1-M5, HiPerNet-A2-M4, and HiPerNet-A2-M5.

Table 3: Effect of inter-stage projections.

CE width	#CE	#Blocks	#M	#Depth	Projection	Pooled	Residual	Stage Type	#Params	#FLOPs	Top-1 (%)
$\{64, 128, 256, 512\}$	3 4 6 3	1 1 1 1	1	26	✗	✗	✓	$S0$	15.4M	2.45B	25.64
	3 4 6 3	1 1 1 1	1	26	✓	✗	✓	$S2$	18.2M	2.81B	25.05
	3 4 6 3	1 1 1 1	1	26	✓	✓	✓	$S2$	18.2M	2.81B	23.95

Table 4: One vs multiple cortical modules

CE width	#CE	#Blocks	#M	#Depth	Projection	Pooled	Residual	Stage Type	#Params	#FLOPs	Top-1 (%)
$\{64, 128, 256, 512\}$	3 4 6 3	1 1 1 1	1	26	✓	✓	✓	$S2$	18.2M	2.81B	23.95
$\{16, 32, 64, 128\}$	3 4 6 3	1 1 1 1	4	26	✓	✓	✓	$S2$	12.1M	1.77B	25.00
	3 4 6 3	1 1 1 1	5	26	✓	✓	✓	$S2$	13.2M	1.95B	24.70
$\{32, 64, 128, 256\}$	3 4 6 3	1 1 1 1	4	26	✓	✓	✓	$S2$	19.8M	3.05B	23.55
	3 4 6 3	1 1 1 1	5	26	✓	✓	✓	$S2$	22.6M	3.51B	23.10

Table 5: Effect of multiple blocks per stage

CE width	#CE	#Blocks	#M	#Depth	Projection	Pooled	Residual	Stage Type	#Params	#FLOPs	Top-1 (%)
$\{16, 32, 64, 128\}$	3 4 6 3	1 1 2 1	4	32	✓	✓	✓	$S2$	13.9M	2.14B	24.52
	3 4 6 3	1 1 2 2	4	37	✓	✓	✓	$S2$	19.6M	2.42B	24.10

4.1.6 MULTIPLE BLOCKS PER STAGE

Table 5 shows the analysis corresponding to the experiment. In one of the variant, we double the number of blocks in stage-4 while in the other, we double the number in both the stage-4 and 5. We observe that by increasing the blocks, depth of the network and parameters increase aggressively. Comparing with Row-2, M5 in Table 4, it is better to stick with smaller depth when we have sufficient CE layers because accuracy are roughly the similar. Also, increased depth increases the latency. Further, from Table 5 it is important to note that while there is a difference of $\sim$ 6M parameters and marginal difference in FLOPs, the accuracy gap is however not that significant. It demonstrates that increases number of parameters in lower resolution doesn't add much to the accuracy.

Discussion. From the above experiments, we see that multiple cortical modules are better than a single module and impact of inter-stage projections with pooling is significant. In addition, multi-CE per block is beneficial to effectively control the number of parameters in contrast to increasing number of blocks. More experiments can be found in supplementary material.

Table 6: HiPerNet vs Existing and dominating CNNs. A '$*$' denotes our re-implementation.

Architecture	#Depth	#Epochs	#Params	#SR_Params	#FLOPs	#SR_FLOPs	Latency	SR_Latency	Top-1 (%)
ResNet-18 (4)	18	120	11.6M	–	1.83B	–	4ms	–	28.84
RepVGG-A0 (3)	22	120	9.10M	8.30M	1.51B	1.46B	8ms	4ms	27.59
ResNet-34 (4)	34	120	21.7M	–	3.68B	–	8ms	–	25.83
RepVGG-A1 (3)	22	120	14.0M	12.7M	2.63B	2.36B	7ms	5ms	25.54
HiPerNet-A1-M4	26	100	12.1M	12.1M	1.77B	1.77B	7ms	5ms	25.00
ResNet-50 (4)	50	120	25.5M	–	4.12B	–	10ms	–	24.70
ResNet-50 (15)	50	120	25.5M	–	4.12B	–	10ms	–	23.85
ResNet-50* (4)	50	100	25.5M	–	4.12B	–	10ms	–	23.80
RepVGG-B0 (3)	28	120	15.8M	14.3M	3.06B	3.40B	7ms	5ms	24.86
HiPerNet-A1-M5	26	100	13.2M	13.2M	1.95B	1.95B	6ms	5ms	24.70
RepVGG-A2 (3)	22	120	28.1M	25.5M	5.69B	5.12B	9ms	7ms	23.52
EfficientNet-B0 (3)	49	120	5.26M	–	0.40B	–	8ms	–	24.89
ResNet-101 (4)	101	120	44.5M	–	7.85B	–	15ms	–	23.60
ResNet-101* (4)	101	100	44.5M	–	7.85B	–	15ms	–	22.94
HiPerNet-A0-M1	26	100	18.2M	18.2M	2.81B	2.81B	7ms	6ms	23.95
HiPerNet-A2-M4	26	100	19.8M	19.8M	3.05B	3.05B	7ms	6ms	23.55
HiPerNet-A2-M5	26	100	22.6M	22.6M	3.51B	3.51B	7ms	6ms	23.10

4.1.7 COMPARISON WITH EXISTING CNNS

ResNet (4). ResNets are indeed very powerful networks. Despite recent architectures (22), ResNet still remains a choice of deployment. Though, many newer variants such as (17) have been proposed but they are still the slight modifications of ResNet. As our motive is to achieve higher performance in

almost every dimension i.e. accuracy, FLOPs, parameters, latency, depth, and lesser training epochs, we show that HiPerNet definitely achieves the desired goals via biological observations.

From the Table 6 shows the accuracy the analysis and comparison with most popular models. If we compare HiPerNet variants with ResNet, we observe that HiPerNet outperforms ResNet in all cases while being alot smaller, faster. In particular, HiPerNet-A2-M4 being 50% lesser layers, 22% lower parameters, 25% lower FLOPs and in 16% less training Epochs outperforms ResNet-50. Similarly the comparison can also be done with other models.

RepVGG (3). RepVGG proposes structural reparameterization (SR) during inference phase which converts a CNN into a plain VGG (19) like structure in order achieve smaller runtime. Our HiPerNet is also fully compatible with reparameterization and can achieve lower latency. From the Table 6, it can be seen that HiPerNet consistently outperforms the recent efficient RepVGG in all domains whether accuracy, FLOPs, depth, latency or training time, parameters. Interestingly, RepVGG offers lower architectural complexity only during inference phase which increases training time, also noticeable from the differences in the parameters with and without SR. On the otherhand, HiPerNet offers extremely lower complexity at both training and testing phase. In addition, HiPerNet can be converted to plain VGG (19) like structure by exploiting SR.

4.2 Latency analysis

Table 6 also provides the FLOPs and latency of various models on RTX2070 GPU with and without structural reprameterization (SR). The latency may vary on a different GPU device therefore the numbers are for reference purpose. Interestingly, HiPerNet performs extremely better relative to all other variants such as RepVGG, ResNet or EfficientNet in a multi-dimensional comparision. More details can be found in supplementary material.

5 Conclusion

In this work, we present a novel high performance deployable CNN architecture HiPerNet. Primary goals to design HiPerNet has been to achieve multi dimensional efficiencies such as parameter, FLOPs, accuracy, latency, lesser training epochs while very low architectural complexity. Towards our goals, we carefully study the biological visual cortex design. From the study, we bring the concept of cortical modules into CNNs which helps achieving us the above goals simultaneously. In addition, we also propose the concept of one-block-one-stage which turn the network extremely faster and less deeper. HiPerNet also benefits from recent structural reparameterization offering very high speeds. In the context of state-of-the-art CNNs, HiPerNet outperforms the most popular and deployable networks such as ResNet, RepVGG in multiple dimensions.

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

[3] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13733–13742, 2021.

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

[5] 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.

[6] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *CVPR*, 2017.

[7] David H Hubel and TN Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3):559–568, 1963.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[9] Ashish Kumar and Laxmidhar Behera. Semi supervised deep quick instance detection and segmentation. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8325–8331. IEEE, 2019.

[10] Ashish Kumar and Laxmidhar Behera. Semi supervised deep quick instance detection and segmentation. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8325–8331. IEEE, 2019.

[11] Ashish Kumar, Mohit Vohra, Ravi Prakash, and Laxmidhar Behera. Towards deep learning assisted autonomous uavs for manipulation tasks in gps-denied environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1613–1620. IEEE, 2020.

[12] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

[13] Vernon B Mountcastle. Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of neurophysiology*, 20(4):408–434, 1957.

[14] Vernon B Mountcastle. The columnar organization of the neocortex. *Brain: a journal of neurology*, 120(4):701–722, 1997.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[16] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[17] Nick Schneider, Florian Piewak, Christoph Stiller, and Uwe Franke. Regnet: Multimodal sensor registration using deep neural networks. In *2017 IEEE intelligent vehicles symposium (IV)*, pp. 1803–1810. IEEE, 2017.

[18] L Sifre and S Mallat. Rigid-motion scattering for image classification. arxiv 2014. *arXiv preprint arXiv:1403.1687*.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[22] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

[23] Keiji Tanaka. Inferotemporal cortex and object vision. *Annual review of neuroscience*, 19(1): 109–139, 1996.

[24] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

[25] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[26] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

[27] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

[28] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.