Designing Network Design Strategies Through Gradient Path Analysis

Chien-Yao Wang[1], Hong-Yuan Mark Liao[1], and I-Hau Yeh[2]
[1]Institute of Information Science, Academia Sinica, Taiwan
[2]Elan Microelectronics Corporation, Taiwan
kinyiu@iis.sinica.edu.tw, liao@iis.sinica.edu.tw, and ihyeh@emc.com.tw

Abstract

Designing a high-efficiency and high-quality expressive network architecture has always been the most important research topic in the field of deep learning. Most of today's network design strategies focus on how to integrate features extracted from different layers, and how to design computing units to effectively extract these features, thereby enhancing the expressiveness of the network. This paper proposes a new network design strategy, i.e., to design the network architecture based on gradient path analysis. On the whole, most of today's mainstream network design strategies are based on feed forward path, that is, the network architecture is designed based on the data path. In this paper, we hope to enhance the expressive ability of the trained model by improving the network learning ability. Due to the mechanism driving the network parameter learning is the backward propagation algorithm, we design network design strategies based on back propagation path. We propose the gradient path design strategies for the layer-level, the stage-level, and the network-level, and the design strategies are proved to be superior and feasible from theoretical analysis and experiments.

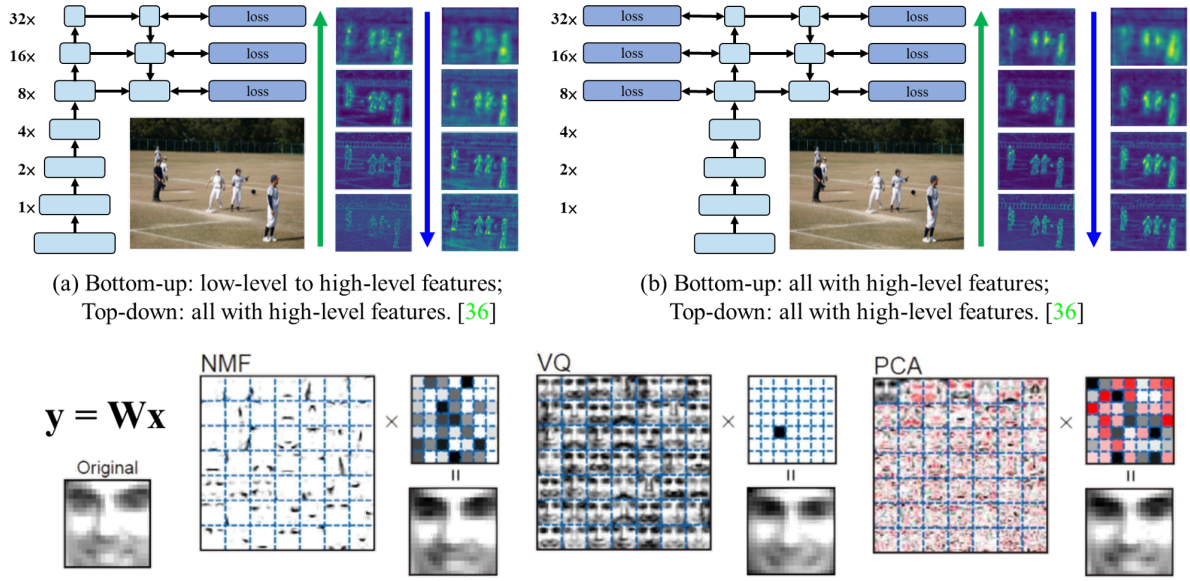
1. Introduction

Deep Neural Networks (DNNs) are now widely used on a variety of devices to solve different kinds of tasks. Millions of scientists, engineers, and researchers are involved in deep learning-related work. They all look forward to designing efficient, accurate, low-cost solutions that can meet their needs. Therefore, how to design network architectures suitable for their products becomes particularly important.

Since 2014, many DNNs have achieved near-human or superior performance than humans on various tasks. For example, Google's GoogLeNet [28] and Microsoft's PReLUNet [4] on image classification, Facebook's Deepface [30] on face verification, and DeepMind's AlphaGo [25] on the Go board, etc. Based on the beginning of the above fields, some researchers continue to develop new architectures or algorithms that are more advanced and can beat the above methods; other researchers focus on how to make DNN-related technologies practical in the daily life of human beings. SqueezeNet [12] proposed by Iandola *et al*. is a representative example, because it reduces the number of parameters of AlexNet [14] by 50 times, but can maintain a comparable accuracy. MobileNet [9, 24, 8] and ShuffleNet [38, 22] are also good examples. The former adds the actual hardware operating latency directly into the consideration of the architecture design, while the latter uses the analysis of hardware characteristics as a reference for designing the neural network architecture.

Just after the ResNet [6], ResNeXt [35], and DenseNet [10] architectures solved the convergence problem encountered in ultra-deep network training, the design of CNN architecture in recent years has focused on the following points: (1) feature fusion, (2) receptive field enhancement, (3) attention mechanism, and (4) branch selection mechanism. In other words, most studies follow the common perception of deep networks, i.e., extract low-level features from shallow layers and high-level features from deep layers. According to the above principles, one can use them to design neural network architectures to effectively combine different levels of features in data path (feed forward path.) However, is such a design strategy necessarily correct? We therefore analyze [16, 36], articles that explore the difference in feature expression between shallow and deep model using different objectives and loss layers. From Figure 1, we found that by adjusting the configuration of objectives and loss layers, we can control the features learned by each layer (shallow or deep). That is to say, what kind of features the weight learns is mainly based on what kind of information we use to teach it, rather than the combination of those layers that the input comes from. Based on this finding, we redefine network design strategies.

Since we propose that the network architecture can be designed with the concept that objective function can guide neural network to learn information, we must first understand how an objective function affects the update of network weights. At present, the main weight update method is the backpropagation algorithm, which uses partial differentiation to generate gradients, and then updates the weights



(a) Bottom-up: low-level to high-level features; Top-down: all with high-level features. [36]

(b) Bottom-up: all with high-level features; Top-down: all with high-level features. [36]

(c) Shallow models with low-level or high-level features. [16]

Figure 1: We can find that no matter for shallow or deep models, and for shallower layers or deeper layers in a deep network, all of them have abilities to extract low-level or high-level features.

by gradient decent. This algorithm propagates gradient information to the shallow layers in chain rule manner, and repeats such steps until the weights of all layers are updated. In other words, the information that an objective function teaches is propagated between layers in the form of gradients. In this paper, we propose that by analyzing the gradient generated through the guidance of objective function, we can design the network architecture by the gradient paths when executing the backpropagation process. We design the network architecture for three different levels of strategies such as layer-level design, stage-level design, and network-level design, which are described below:

1. **Layer-level design:** At this level we design gradient flow shunting strategies and use them to confirm the validity of the hypothesis. We adjust the number of layers and calculate the channel ratio of residual connection, and then design Partial Residual Network (PRN) [32], as described in Section 2.2.
2. **Stage-level design:** We add hardware characters to speed up inference on the network. We maximize gradient combinations while minimizing hardware computational cost, and thus design Cross Stage Partial Network (CSPNet) [33], as described in Section 2.3.
3. **Network-level design:** We add the consideration of gradient propagation efficiency to balance the leaning ability of the network. When we design the network architecture, we also consider the gradient propagation path length of the network as a whole, and therefore design Efficient Layer Aggregation Network (ELAN), as described in Section 2.4.

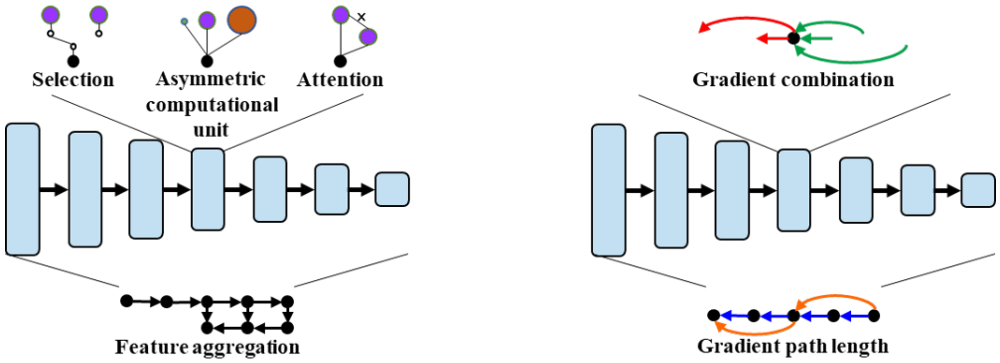
2. Methodology

2.1. Network Design Strategies

In this paper we divide network design strategies into two kinds: (1) data path design strategies, and (2) gradient path design strategies, as shown in Figure 2. Data path design strategy mainly focuses on designing feature extraction, feature selection, and feature fusion operations to extract features with specific properties. These features can help subsequent layers use these features to further obtain better properties for conducting more advanced analysis. The purpose of applying gradient path design strategies is to analyze the source and composition of the gradients, and how they are updated by the driving parameters. Then, one can use the results of the above analysis to design the network architecture. The design concept is to hope that the final parameter utilization rate is higher, and thereby achieve the best learning effect.

Next, we will discuss the advantages and disadvantages of the data path design strategy and the gradient path design strategy, respectively. There are three advantages of the data path design strategy: (1) **can extract features with specific physical meaning.** For example, use asymmetric computational units to extract features with different receptive fields [5, 1, 21]; (2) **can automatically select suitable operation units with parameterized models for different inputs.** For example, using kernel selection to handle inputs with different properties [19, 2]; and (3) **the learned features can be reused directly.** For example, feature pyramid networks can directly utilize features extracted from different layers for more accurate predictions [20]. The data path

1.能提取具有特定物理意义的特征。2.能针对不同的输入自动选择合适的操作单元，并具有参数化模型。 3.学习到的特征可以直接重用。



(a) Data path design strategies (b) Gradient path design strategies

Figure 2: Two main network design strategies: (a) data path design strategy; and (b) gradient path design strategy.

design strategy has two shortcomings: (1) In the process of training, it sometimes leads to unpredictable degradation of the effect, and at this time, a more complex architecture needs to be designed to solve the problem. For example, the pairwise relationship of non-local networks is easy to degenerate into unary information [37]; and (2) various specially designed arithmetic units are easy to cause difficulties in performance optimization. For example, in the ASIC design dedicated to AI, if the designer wants to add an arithmetic unit, an additional set of circuits is required.

As for the gradient path design strategy, there are in total three advantages: (1) **can effectively use network parameters.** In this part, we propose that by adjusting the gradient propagation path, the weights of different computing units can learn various information, and thereby achieve higher parameter utilization efficiency; (2) **has stable model learning ability.** Since gradient path design strategy directly determines and propagates information to update weights to each computing unit, the designed architecture can avoid degradation during training; and (3) **has efficient inference speed.** The gradient path design strategy makes parameter utilization very efficient, so the network can achieve higher accuracy without adding additional complex architecture. Because of the above reasons, the designed network can be lighter and simpler in architecture. The proposed gradient path design strategy only has one shortcoming, i.e., when the gradient update path is not a simple reversed feedforward path of the network, the difficulty of programming will be greatly increased.

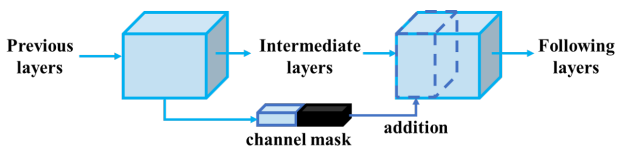


Figure 3: Masked residual layer. Only part of channels will go through the identity connection.

2.2. Partial Residual Networks

Partial Residual Network (PRN) [32] was proposed by our team in 2019, and its design concept belongs to the layer-level design strategy. In the design of PRN, the main concept is to maximize the combination of gradients used to update the weights of each layer. There are two main factors that affect the combination of gradients. The first one is **the source layer of the gradient**. The source layer is composed of the nodes connected the indegree edges of the gradient path. The second factor that affects gradient combination is **the time it takes for the gradient flow to arrive at a particular layer from the loss layer through the operation of the chain rule**. One thing to be noted is that when the gradient changes during the process of the chain rule update, the amount of loss information it covers will gradually fade as the chain grows. We define the above time duration as the number of layers that the gradient flow needs to travel from the loss layer to a specific layer. In PRN, we propose the following two structures to improve ResNet:

Masked residual layer. In the design of ResNet [6], the output of each computational block is added together with an identity connection, and such a structure is called residual layer. In PRN, we multiply identity connection by a binary mask and only allow the feature map of some of the channels to be added to the output of the computational block. We call this structure masked residual layer, and its architecture is shown in Figure 3. Using the mechanism of a masked residual layer allows the feature map to be divided into two parts, in which the weights corresponding to the channels that are masked and the weights corresponding to the channels with identity connection will significantly increase the number of gradient combinations due to the aforementioned masking effect. In addition, differences in gradient sources will simultaneously affect the overall gradient timestamp (time node along time axis), thus making gradient combinations more abundant.

Asymmetric residual layer. Under the ResNet architecture, only feature map of the same size can be added, which is why it is a very restricted architecture. Generally, when the calculation amount and inference speed of the optimized architecture are performed, we are often limited by this architecture and cannot design an architecture that meets the requirements. Under the architecture of PRN, the masked residual layer proposed by us can regard the inconsistency of the number of channels as some channels being blocked, and thus allow feature map with different number of channels to perform masked residual operations. We call the layer that operates in the above manner an asymmetric residual layer. An asymmetric residual layer is designed in such a way that the network architecture is more flexible and more able to maintain the properties of a gradient path-based model. For example, when we are doing feature integration, the general approach requires additional transition layers to project different feature maps to the same dimension, and then perform the addition operation. However, the above-mentioned operation will increase a large number of parameters and amount of computations, and will also make the gradient path longer, and thus affect the convergence of the network. The introduction of asymmetric residual layer can perfectly solve similar issues.

2.3. Cross Stage Partial Networks

CSPNet [33] was proposed by our team in 2019, and it is a stage-level gradient path-based network. Like PRN, CSPNet is based on the concept of maximizing gradient combinations. The difference between CSPNet and PRN is that the latter focuses on confirming the improvement of network learning ability by gradient combination from theoretical perspective, while the former is additionally designed for further architecture optimization for hardware inference speed. Therefore, when designing CSPNet, we extend the architecture from layer-level to stage-level, and optimize the overall architecture. CSPNet mainly has the following two structures:

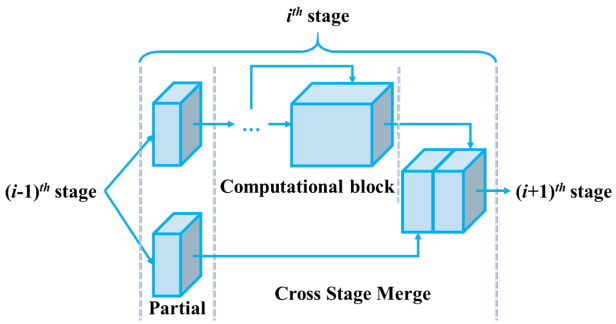


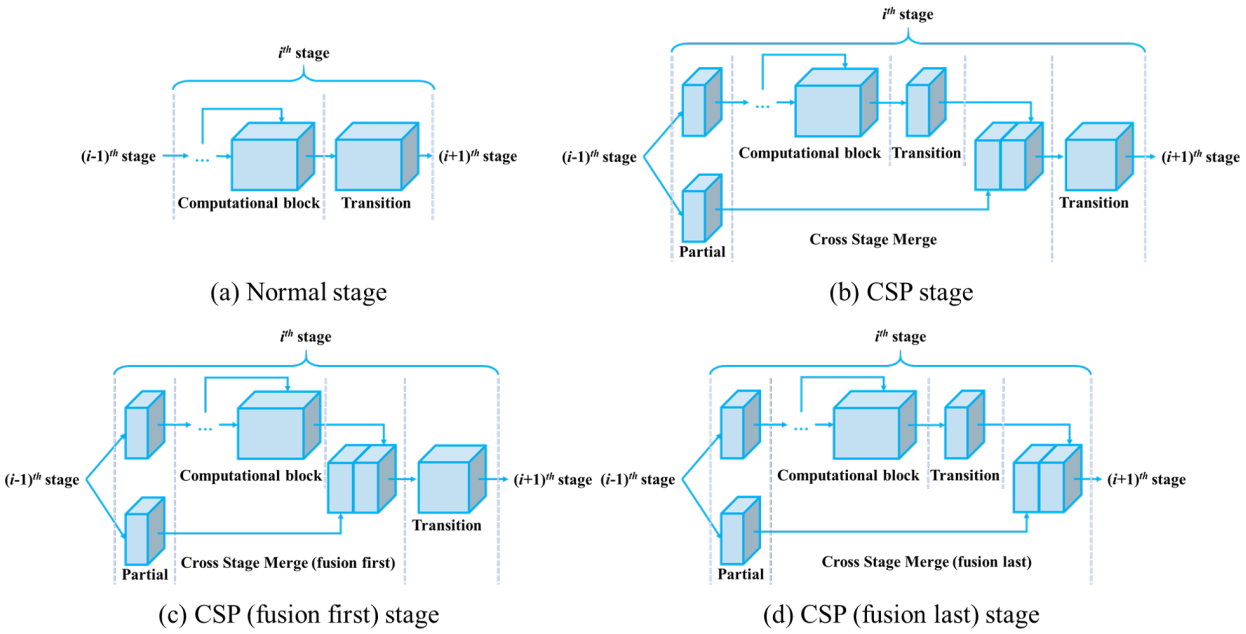
Figure 4: Cross stage partial operation. CSP operation separates feature map of the base layer into two parts, one part will go through a computational block and the other part is then combined with processed feature map to the next stage.

Cross stage partial operation. From the perspective of maximizing the source of the gradient, we can easily find that the source of the gradient can be maximized when each channel has a different gradient path. Also, from the perspective of maximizing gradient timestamps, we know that the number of gradient timestamps can be maximized when each channel has computational blocks of different depths. Following the above concept, we can derive an architecture designed to maximize both the gradient source and gradient timestamp. And this architecture will be the Inception-like architecture [28, 13, 29, 27] and the fractal-like architecture [15] with depth-wise convolution. Although the above design can effectively improve the parameter utilization, it will greatly reduce the parallelization ability. In addition, it will cause the model to significantly reduce the inference speed on inference engines such as GPU and TPU. From the previous analysis, we know that dividing the channel can increase the number of gradient sources, and making the sub-networks connected by different channels with different layers can increase the number of gradient timestamps. The cross stage partial operation we designed can maximize the combination of gradients and increase the inference speed without breaking the architecture and can be parallelized. This architecture is shown in Figure 4. In Figure 4, we divide a stage's input feature map into two parts, and use this manner to increase the number of gradient sources. The detailed procedure is as follows: we first divide the input feature map into two parts and one of them passes through the computational block, and this computational block can be any computational block such as Res block, ResX block, or Dense block. As for the other part, it directly crosses the entire stage, and then integrates with the part that goes through the computational block. Since only part of the feature map enters the computational black for operation, this kind of design can effectively reduce the amount of parameters, operation, memory traffic, and memory peak, allowing the system to achieve faster inference speed.

从前面的分析中我们知道，划分信道可以增加梯度源的数量，使不同信道连接的子网络具有不同的层数可以增加梯度时间戳的数量。我们设计的交叉阶段部分运算可以在不破坏体系结构的情况下最大化梯度组合，提高推理速度，并且可以并行化。

Gradient flow truncate operation. In order to make our designed network architecture more powerful, we further analyze the gradient flow used to update the CSPNet. Since shortcut connections are often used in computational blocks, we know that the gradient sources that provide the two paths are bound to overlap a lot. We know that when a feature map passes through a kernel function, it is equivalent to a spatial projection. Usually we can insert a transition layer at the end of both paths to truncate the duplicated gradient flow. Through the above steps, we can make the information learned from the two paths and adjacent stages have more obvious diversity. We designed three different combinations of duplicate gradient flow truncate operations, as shown in Figure 5. These operations can be matched with different architectures, such as computational blocks and down-sampling blocks to achieve better results.

通常我们可以在两条路径的末端插入一个过渡层来截断重复的梯度流 。



(a) Normal stage (b) CSP stage

(c) CSP (fusion first) stage (d) CSP (fusion last) stage

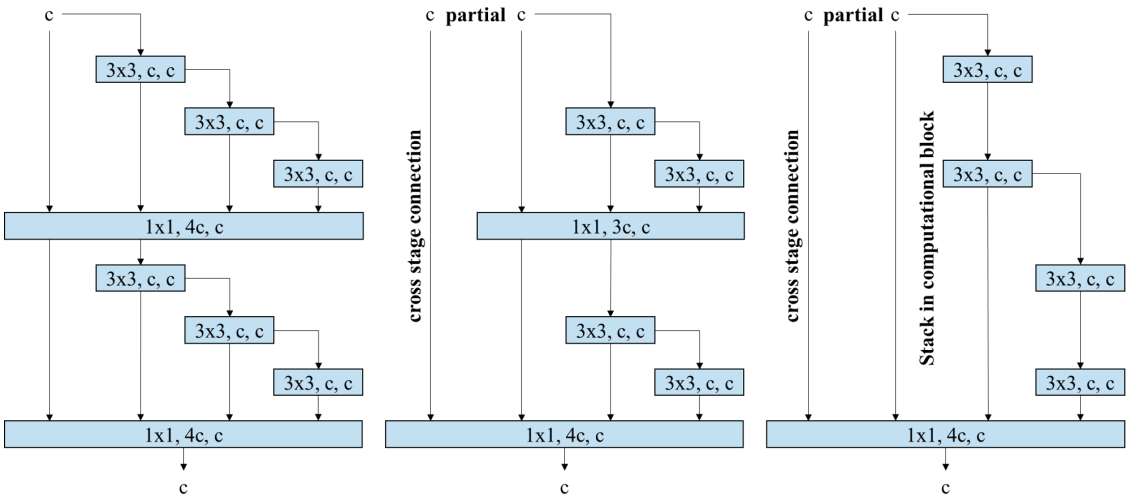
Figure 5: Cross stage partial networks. (a) original network, (b) CSP fusion: transition → concatenation → transition, (c) CSP fusion first: concatenation → transition, and (d) CSP fusion last: transition → concatenation

2.4. Efficient Layer Aggregation Networks

The codes of **Efficient Layer Aggregation Networks (ELAN)** was released by our team in July 2022. It falls into the category of the **gradient path designed network** at the network-level. The main purpose of designing ELAN is to solve the problem that the convergence of the deep model will gradually deteriorate when executing model scaling. We analyze the shortest gradient path and the longest gradient path through each layer in the overall network, thereby designing a layer aggregation architecture with efficient gradient propagation paths. ELAN is mainly composed of VoVNet [17] combined with CSPNet [33], and optimizes the gradient length of the overall network with the structure of **stack in computational block**. In what follows, we will elaborate how stack in computational block works.

Stack in computational block. When we are doing model scaling, there will be a phenomenon, that is, when the network reaches a certain depth, if we continue to stack computational blocks, the accuracy gain will be less and less. To make matters worse, when the network reaches a certain critical depth, its convergence begins to deteriorate, resulting in an overall accuracy that is worse than shallow networks. One of the best examples is scaled-YOLOv4 [31], we see that its P7 model uses expensive parameters and operations, but only a small amount of accuracy gain, and the same phenomenon occurs in many popular networks. For example, ResNet-152 is about three times as computationally intensive as ResNet-50, but offers less than 1% improvement in accuracy on ImageNet [6]. When ResNet is stacked to 200 layers, its accuracy is even worse than ResNet-152 [7]. Also, when VoVNet is stacked to 99 layers, its accuracy is even much lower than that of VoVNet-39 [18]. From the gradient path design strategy point of view, we speculate that the reason why the accuracy of VoVNet degenerates much faster than ResNet is because the stacking of VoVNet is based on the OSA module. We know that every OSA module contains a transition layer, so every time we stack an OSA module, the shortest gradient path of all layers in the network increases by one. As for ResNet, it is stacked by residual blocks, and the stacking of residual layers will only increase the longest gradient path, and will not increase the shortest gradient path. In order to verify the possible effects of model scaling, we did some experiments based on YOLOR-CSP [34]. From the experimental results we found that when the stacking layer reaches 80+ layers, the accuracy of **CSP fusion first** starts to perform better than the normal CSPNet. At this point, the shortest gradient path of the computational block of each stage will be reduced by 1. As the network continues to widen and deepen, **CSP fusion last** will get the highest accuracy, but at this point the shortest gradient path of all layers will be reduced by 1. The above experimental results confirmed our previous hypothesis. With the support of the above experiments, we designed the "stack in computational block" strategy in ELAN, as shown in Figure 6. The purpose of our design is to avoid the problem of using too many transition layers and making the shortest gradient path of the whole network quickly become longer. We hope that the above design strategy allows ELAN to be successfully trained when the network is stacked deeper.

我们设计的目的是避免使用过多的过渡层，使整个网络的最短梯度路径迅速变长的问题。我们希望上述设计策略允许在网络堆叠更深的情况下成功地训练ELAN。



(a) VoVNet [17] (b) CSPVoVNet [31] (c) ELAN

Figure 6: Efficient layer aggregation networks.

Table 1: Analysis of different networks. [39]

Model	Parameters	Shortest gradient path	Aggrgated features
PlainNet	$O(N)$	$O(N)$	$O(1)$
ResNet [6]	$O(N)$	$O(1)$	$O(l)$
DenseNet [10]	$O(N^2)$	$O(1)$	$O(l)$
Sparse ResNet [39]	$O(N)$	$O(\log N)$	$O(\log l)$
Sparse DenseNet [39]	$O(N \log N)$	$O(\log N)$	$O(\log l)$

3. Analysis

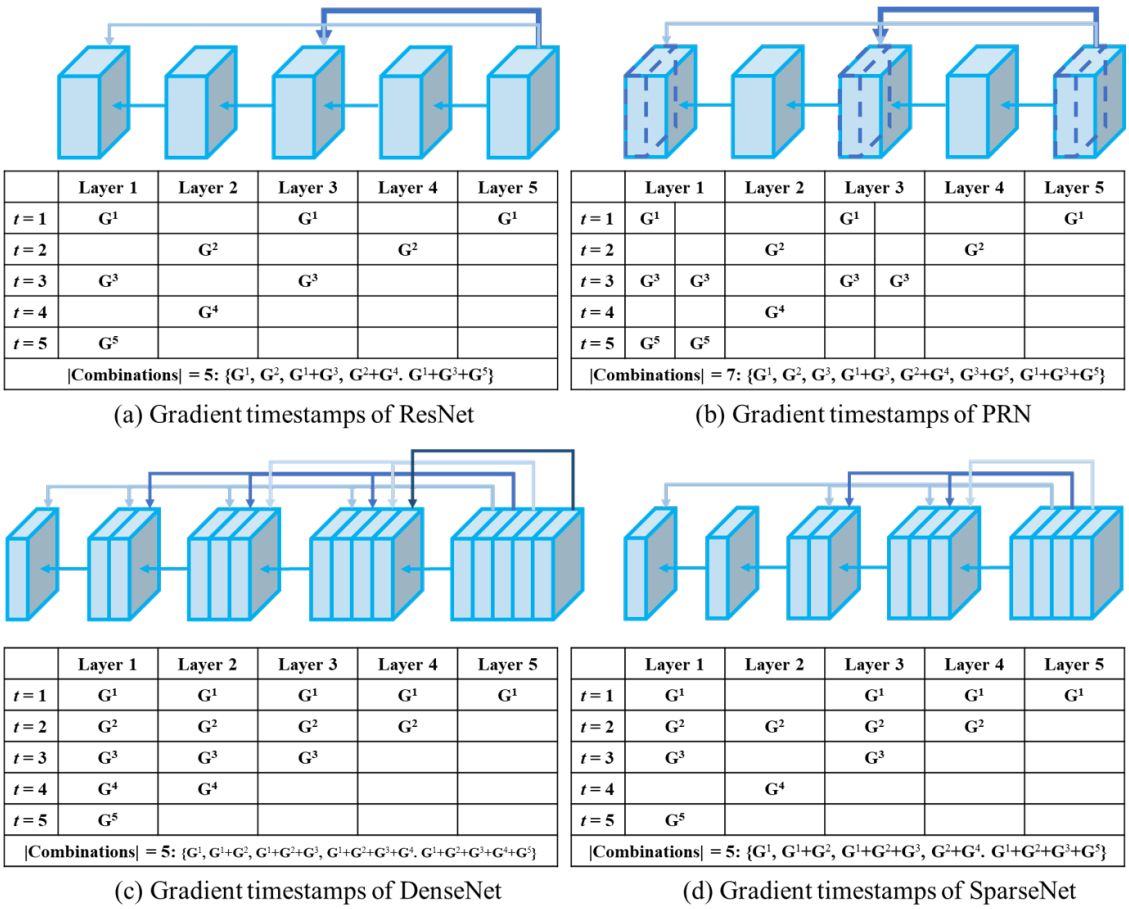
In this section we will analyze the proposed gradient path design strategies based on the classical network architecture. First, we will analyze the existing network architecture and the proposed PRN with the concept of gradient combination, and this example shows that the network architecture that performs well does have a richer gradient combination. Then we will analyze how the proposed CSPNet brings richer gradient combinations and other benefits. Finally, we analyze the importance of length of gradient path by stop gradient, and thus confirm that the proposed ELAN has a design concept advantage.

3.1. Analysis of gradient combination

General researchers often use the shortest gradient path and the number of integrated features to measure the learning efficiency and ability of network architectures. However, from the literature [39] we can find that these metrics are not completely related to accuracy and parameter usage, as shown in Table 1. We observe the process of gradient propagation and find that the gradient combination used to update the weights of different layers matches the learning ability of the network well, and in this section we will analyze the gradient combination. Gradient combinations are composed by two types of component, namely **gradient timestamp** and **gradient source**. Next we will analyze them separately.

Gradient Timestamp. Figure 7 shows the architecture of ResNet [6], PRN, DenseNet [10], and SparseNet [39]. Among them, we unfold the cascaded residual connection and concatenation connection to facilitate the observation of the gradient propagation process. In addition, the gradient flow delivery timestamps on each architecture is also shown in Figure 7. The gradient sequence is equivalent to a breadth first search process, and each sequence will visit all the outdegree nodes reached by the previous round of traverse. From Figure 7, we can see that PRN uses the channel splitting strategy to enrich the gradient timestamps received by the weights corresponding to different channels. As for SparseNet, it uses sparse connections to make the timestamps received by the weight connections corresponding to different layers more variable. Both of the above methods can learn more diverse information with different weights, which makes our proposed architecture more powerful.

Gradient Source: Figure 8 shows the gradient sources from ResNet [6], PRN and DenseNet [10] at the first gradient timestamp. It can be seen from Figure 8 that the concatenation connection-based architectures, such as DenseNet and SparseNet [39], belong to the network that must be specially handled. This is because in the gradient propagation process, if it is the gradient information propagated by the same layer at a certain gradient timestamp, because the gradient flow has been split beforehand, it cannot be processed like a general network. As for the residual connection-based architectures, such as ResNet [6] and PRN, the exact same gradient information is propagated to all layers of outdegree. Since the outdegree of PRN is only connected to some channels of other layers, it can have a richer combination



	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
$t=1$	G^1		G^1		G^1
$t=2$		G^2		G^2	
$t=3$	G^3		G^3		
$t=4$		G^4			
$t=5$	G^5				
|Combinations| = 5: $\{G^1, G^2, G^1{+}G^3, G^2{+}G^4, G^1{+}G^3{+}G^5\}$					

(a) Gradient timestamps of ResNet

	Layer 1		Layer 2	Layer 3		Layer 4	Layer 5
$t=1$	G^1			G^1			G^1
$t=2$			G^2			G^2	
$t=3$	G^3	G^3		G^3	G^3		
$t=4$			G^4				
$t=5$	G^5	G^5					
|Combinations| = 7: $\{G^1, G^2, G^3, G^1{+}G^3, G^2{+}G^4, G^3{+}G^5, G^1{+}G^3{+}G^5\}$							

(b) Gradient timestamps of PRN

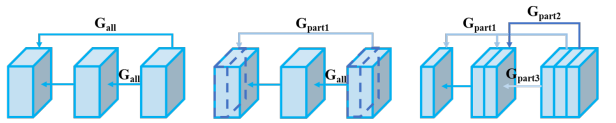
	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
$t=1$	G^1	G^1	G^1	G^1	G^1
$t=2$	G^2	G^2	G^2	G^2	
$t=3$	G^3	G^3	G^3		
$t=4$	G^4	G^4			
$t=5$	G^5				
|Combinations| = 5: $\{G^1, G^1{+}G^2, G^1{+}G^2{+}G^3, G^1{+}G^2{+}G^3{+}G^4, G^1{+}G^2{+}G^3{+}G^4{+}G^5\}$					

(c) Gradient timestamps of DenseNet

	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
$t=1$	G^1		G^1	G^1	G^1
$t=2$	G^2	G^2	G^2	G^2	
$t=3$	G^3		G^3		
$t=4$		G^4			
$t=5$	G^5				
|Combinations| = 5: $\{G^1, G^1{+}G^2, G^1{+}G^2{+}G^3, G^2{+}G^4, G^1{+}G^2{+}G^3{+}G^5\}$					

(d) Gradient timestamps of SparseNet

Figure 7: Gradient timestamps. (a) ResNet, (b) PRN, (c) DenseNet, and (d) SparseNet.



(a) Gradient source of ResNet (b) Gradient source of PRN (c) Gradient source of DenseNet

Figure 8: Gradient source. (a) ResNet, (b) PRN, and (c) DenseNet.

of gradients than ResNet as a whole. In addition, there are network architectures that use other split-transform-merge strategies, such as group convolution-based ResNeXt [35] and depth-wise convolution-based MobileNet [9], etc., which can also increase the number of gradient sources.

Summary. In summary, through the analysis of gradient timestamp and gradient sources generated in the process of gradient backward propagation, we can clearly explain the existing popular network architectures and the information learned by our proposed PRN and the utilization efficiency of parameters. In ResNet, different layers share many gradients of the same timestamp and the same gradient source, and DenseNet passes the gradient information of the same timestamp but different sources to the corresponding layers. This part clearly explains why the concatenation connection-based DenseNet can avoid the problem of easily learning a lot of useless information like the residual connection-based ResNet. Our proposed PRN uses a simple masked residual layer to increase the number of gradient combinations along time axis while maintaining the ResNet network topology, and to divert the gradient sources, thereby increasing the variability of the gradient sources.

在Resnet中，不同的层共享同一时间戳的多个梯度和同一梯度源，DenseNet将同一时间戳但不同源的梯度信息传递给相应的层。这一部分清楚地解释了为什么基于连接的DenseNet可以避免像基于剩余连接的ResNet那样容易学习大量无用信息的问题。我们提出的PRN使用一个简单的掩蔽残差层，在保持RESNET网络拓扑结构的同时，增加沿时间轴的梯度组合数，并对梯度源进行转移，从而增加梯度源的可变性。

3.2. Analysis of cross stage partial strategy

CSPNet is designed to enhance online learning ability and speed up inference at the same time, so we will discuss the advantages of the CSPNet strategy from these two aspects separately. In the analysis conducted in Section 3.1, we observed that even if the number of combinations generated by the gradient sources is the same, when the common components received between different combinations are reduced, which makes the gradient components more abundant, and also makes the network learn better. This phenomenon actually occurs in the process of learning a large number of parameters for singe-layer weights. For example, dropout [26] uses random Bernoulli masking neu-

rons to prevent parameters to learn co-adaptation information. From a mathematical model point of view, dropout is to update the weights of different parts by using the gradients generated by different inputs, which is equivalent to a random ensemble structure. As for CSPNet, it directly increases the richness of the gradient combination through the difference in time and the spatial transformation of the gradient on the gradient path. Next, we will introduce what strategy the CSPNet uses to solve the problem of duplicated gradient information, and how it improves resource usage.

Duplicated Gradient Information: In Section 3.1 we analyzed the number of gradient combinations and the effect of diversity on the learning ability of the network. In CSPNet, we further analyze the gradient information content received by different layers, and design the architecture to improve the efficiency of parameter usage. From the gradient combination of PRN and SparseNet, it can be found that they have a commonality in the process of increasing the richness of gradient combination, that is, the situation of receiving a large number of duplicated gradient information through residual connection or dense connection is significantly reduced. We speculate that these duplicated gradients are the main reason for the large number of weights to easily learn the co-adaptation information. As for PRN, it utilizes gradient timing differences to update the weights of local channels. With the update process of chain rule, the above timing difference will spread to the entire network, and then achieve a richer gradient combination. In addition, CSPNet directly uses cross stage connection to make the two paths of the entire stage have a great timing difference, and uses different fusion structures to reduce the duplicated gradient information between stage and stage, or between computational block path and cross stage connection path.

Resource Usage Efficiency: Taking Darknet-53 as an example, suppose that cross stage partial operation divides the feature map into two equal parts according to the direction of the channel. At this time, the number of input channel and output channel of residual block is halved, while the number of channels in the middle remains unchanged. According to the above structure, the overall calculation and parameter amount of computational blocks will be reduced to half of the original, and the memory peak is the sum of the size of input feature map and output feature map, so it will be reduced to 2/3 of the original. In addition, since the input channel and output channel of the convolution layer in the entire computational blocks are equal, the memory access cost at this time will be the smallest.

Summary. In summary, CSPNet successfully combines the concept of gradient combination with the efficiency of hardware utilization so that the designed network architecture improves the learning ability and inference speed at the same time. CSPNet uses only simple channel split, cross stage connection and a small amount of extra transition layers, and successfully completes the preset goal without changing the original network computing units. Another benefit of the CSPNet is that it can be applied to many popular network architectures and improve overall network efficiency in all aspects. In Table 2 we show the excellent performance of the CSPNet applied to several popular network architectures. Finally, because the CSPNet has lower requirements on many hardware resources, it is suitable for high-speed inference on devices with more stringent hardware constraints.

Table 2: Apply CSPNet on different Networks.

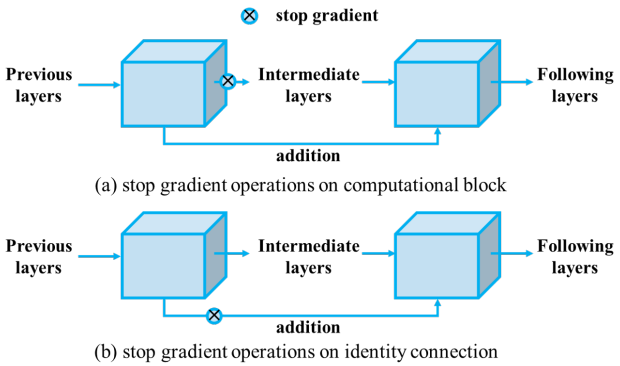
Model	FLOPs	#Params	Top-1
Darknet-53 [23]	18.57G	41.57M	77.2%
+ CSP	**13.07G** (-30%)	**27.61M** (-34%)	**77.2%** (=)
ResNet-50 [6]	9.74G	22.73M	75.8%
+ CSP	**8.97G** (-8%)	**21.57M** (-5%)	**76.6%** (+0.8)
ResNeXt-50 [35]	10.11G	22.19M	77.8%
+ CSP	**7.93G** (-22%)	**20.50M** (-8%)	**77.9%** (+0.1)

* Results are obtained on ImageNet validation set.

CSPNet的另一个好处是它可以应用许多流行的网络体结构，从各个方面高网络的整体效率在表2中，我们展示了应用于几种流网络体系结构的CSPNet的优异性能。最后，由于CSPNet对许多硬件资源的要求较低，此适合在硬件约束严格的设备上进行速推理。

3.3. Analysis of length of gradient path

As discussed in Section 3.1, we understand that the shorter the gradient path of the overall network does not mean the stronger the learning ability. Furthermore, even if the length of the overall gradient combination path is fixed, we find that the learning ability of the ResNet still degrades when the stacking is very deep. However, we found that the above problem can be used to disassemble the ResNet into shallower random sub-networks for training during the training phase using stochastic depth [11], which can make the ultra-deep ResNet converge to better results. The above phenomenon tells us that when analyzing the gradient path, we can not only look at the shortest gradient path and the longest gradient path of the overall network, but need a more detailed gradient path analysis. In what follows, we will control the gradient path length by adjusting gradient flow during training, and then discuss the gradient length strategy when designing the network architecture from the results.



(a) stop gradient operations on computational block

(b) stop gradient operations on identity connection

Figure 9: Architectures for stop gradient ablation studies.

总之，从以上的实验和分析中，我们推断在规划整体网络的梯度路径时，不能只考虑最短的梯度路径，而应该保证每一层的最短梯度路径都能得到有效的训练。至于整体网络最长梯度路径的长度，将大于或等于任一层的最长梯度路径。因此，在实施网络级梯度路径设计策略时，需要考虑网络中所有层的最长最短梯度路径长度，以及整个网络的最长梯度路径。

Table 3: Results of stop gradient ablation study.

Model	FLOPs	AP^{box}	AP^{mask}
YOLOR-CSP [34]	159.0G	**51.0%**	**41.1%**
YOLOR-CSP [34] + Figure 9 (a)	159.0G	48.8%	39.5%
YOLOR-CSP [34] + Figure 9 (b)	159.0G	47.7%	38.7%

* Results are obtained on MS COCO validation set.

Stop gradient: First we explore the importance of the shortest gradient length based on ResNet. Compared to PlainNet, each residual block in ResNet has a part of gradient across the computational block through identity connection in addition to the gradient passing through the computational block. Here, we perform stop gradient operations on computational block and identity connection respectively, as shown in Figure 9. When we execute stop gradient on identity connection, the gradient path of the overall network will be like PlainNet. That is to say, the longest gradient path is the same length as the shortest gradient path, and the network depth is also the same. When we perform stop gradient on a computational block, the shortest gradient path will go directly through all residual connection and directly to the starting layer, and the shortest gradient path length is 1 at this time. Since each computational block has two layers, its longest gradient path is 2. We can use these two sets of settings to observe the benefits of residual learning itself and the reduction of gradient path. We use object detection and instance segmentation in Microsoft COCO dataset as the baseline model to perform ablation study on YOLOR-CSP [34] and show the results in Table 3. Experimental results show that performing a shortened gradient path in ResNet is indeed an important factor for better convergence of deep networks.

Table 4: Apply ELAN concept on VoVNet.

Model	FLOPs	AP^{box}	AP^{mask}
Deep VoVNet [17] + ELAN	253.4G	53.3%	42.9%
Deep VoVNet [17] + ELAN + CSP	236.5G	**53.4%**	**42.9%**

* Deep VoVNet is a VoVNet with 99 convolutional layers.
* Results are obtained on MS COCO validation set.

Gradient path planning: From the above analysis and our experiment of model scaling using CSP fusion in YOLOR-CSP, we re-plan the transition layer of VoVNet and conduct experiment. We first remove the transition layer of each OSA module of the deep VoVNet, leaving only the transition layer of the last OSA module in each stage. We organize both the longest gradient path of the network and the shortest gradient path through each layer in the same way as described above. At the same time, we also apply the CSPNet structure to the above network to further observe the versatility of CSPNet, and the related experimental results are shown in Table 4. We clearly see that deep VoVNet has changed from failing to converge to one that can converge well and achieve very good accuracy.

Summary. In short, from the above experiments and analysis, we infer that when planning the gradient path of the overall network, we should not only consider the shortest gradient path, but should ensure that the shortest gradient path of each layer can effectively been trained. As for the length of the longest gradient path of the overall network, it will be greater than or equal to the longest gradient path of any layer. Therefore, when practicing network-level gradient path design strategies, we need to consider the longest shortest gradient path length for all of layers in the network, and the longest gradient path for the overall network.

4. Experiments

4.1. Experimental setup

We use the Microsoft COCO dataset as the basis for performing validation on object detection and instance segmentation. As for baseline architecture we chose residual-based YOLOv3-SPP [23], and for baseline decoder we chose a combination of YOLOR [34] and YOLO-v5 (r6.2) [3]. As for baseline training strategy and all methods of training hyperparameters, we follow the rules adopted by YOLOR [34]. We name the baseline model trained in the above YOLOR-v3 [34]. In the following experiments, we will verify one-by-one the effect of our proposed layer-level, stage-level, and network-level architecture based on the gradient path design strategies. Finally, we compare the proposed method with baseline-related methods such as YOLOR-v3 [34] and YOLO-v5 (r6.2) [3].

4.2. Layer-level gradient path design strategies

Table 5: Ablation study of PRN.

Model	FLOPs	AP^{box}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{75}
YOLOR-v3 [34]	194.6G	49.5%	53.9%	40.9%	43.1%
+ PRN	194.6G	**50.0%**	**54.4%**	**41.0%**	**43.4%**

* YOLO-v3 + PRN equals to YOLOv3-FPRN [32].
* Results are obtained on MS COCO validation set.

In the experiment of PRN, we set the number of channels shaded by the masked residual layer to half of the original number of channels, and the results obtained in the experiment are shown in Table 5. Since the design of PRN maintains all parameters and topology of the entire network, only the addition operation in residual connection is reduced by half, so the overall calculation amount is almost unchanged. However, YOLOR-PRN gets a significant improvement in accuracy due to the addition of the combination of gradients that each layer uses to update the weights. Compared to YOLOR-v3, PRN improves 0.5% AP on object detection, and we can also observe high quality and significant improvement. On instance segmentation, we improved AP by 0.1% and AP_{75} by 0.3%.

4.3. Stage-level gradient path design strategies

Table 6: Ablation study of CSPNet.

Model	FLOPs	AP^{box}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{75}
YOLOR-v3 [34]	194.6G	49.5%	53.9%	40.9%	43.1%
+ CSP	159.0G	**51.0%**	**55.5%**	**41.1%**	**43.4%**
+ CSP fusion first	158.1G	**50.8%**	**55.3%**	**41.0%**	**43.3%**
+ CSP fusion last	155.6G	**50.6%**	**55.3%**	**40.9%**	**43.3%**
+ CSP no fusion	154.8G	**50.5%**	**55.2%**	**40.9%**	**43.2%**

* YOLOR-v3 + CSP equals to YOLOR-v4-CSP [34].
* Results are obtained on MS COCO validation set.

In the CSPNet experiment, we follow the principle of optimizing the inference speed and set the gradient split ratio to 50%-to-50%, and we show the experimental results in Table 6. Since only half of the channel's feature maps will enter the computational block, we can clearly see that YOLOR-CSP significantly reduces the amount of calculations by 22% compared to YOLOR-v3. However, with rich gradient combinations, YOLOR-CSP significantly improves the AP by 1.5% on the object detection. Compared to YOLOR-v3, the combination of YOLOR and CSPNet (YOLOR-CSP) added more high-quality results. We further compare gradient flow truncate operations for reducing repeated gradient information, and we clearly see that the strategy of YOLOR-CSP does learn better than CSP fusion first and CSP fusion last. It is worth mentioning that no matter what fusion strategy is adopted, the CSP-based architecture has a much lower computational load than YOLOR-v3 and an accuracy far better than YOLOR-v3.

4.4. Network-level gradient path design strategies

Table 7: Ablation study of ELAN.

Model	FLOPs	AP^{box}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{75}
YOLOR-v3 [34]	194.6G	49.5%	53.9%	40.9%	43.1%
+ ELAN-{1,1}s	126.4G	**50.2%**	**54.5%**	40.6%	42.9%
+ ELAN-{2,1}s	143.2G	**51.4%**	**55.8%**	**41.5%**	**43.7%**
+ ELAN-{2,2}s	164.0G	**51.8%**	**56.5%**	**41.6%**	**43.3%**

* $\{a,b\}$s means stack in computational blocks a and b times on backbone and neck, respectively.
* Results are obtained on MS COCO validation set.

In the ELAN experiment, we test the stacking times of computational blocks in backbone and neck respectively, and we show the results in Table 7. From this table, we can clearly see that ELAN can still improve the performance of object detection by 0.7% AP under 35% less amount of computation than YOLOR-v3. In ELAN, we can flexibly set the number of stacks to make a trade-off between accuracy and computation. From the experimental results listed in Table 7, we can see that under the stack setting of 2,1s, YOLOR-ELAN can significantly improve the performance of object detection and instance segmentation by 1.9% AP and 0.6% AP, respectively, under the condition of reducing the amount of computation by 26%.

4.5. Comparison

Table 8: Compariso with baseline methods.

Model	FLOPs	#Params	AP^{box}	AP^{mask}
YOLO-v5l (r6.2) [3]	147.7G	47.9M	49.1%	40.0%
YOLO-v5x (r6.2) [3]	265.7G	88.8M	50.9%	41.4%
YOLOR-v3 [34]	194.6G	64.3M	49.5%	40.9%
YOLOR-PRN	194.6G	64.3M	50.0%	41.0%
YOLOR-CSP	159.0G	54.3M	51.0%	41.1%
YOLOR-ELAN	**143.2G**	**34.5M**	**51.4%**	**41.5%**

* Results are obtained on MS COCO validation set.

Finally, we comprehensively compare the three proposed methods, that is, YOLOR-PRN designed by layer-level design strategies, YOLOR-CSP designed by stage-level design strategies, and YOLOR-ELAN designed by network-level design strategies, with baseline YOLOR-v3 and YOLOv5 (r6.2), and the results are shown in Table 8. From the table, we see that the model designed based on gradient path design strategy outperforms the baseline-based methods in all aspects. In addition, regardless of the amount of computation. The amount of parameters, and the accuracy, the YOLOR-ELAN designed by network-level design strategy can obtain the most outstanding performance in an all-round way. From the results we confirm that based on the gradient path analysis, we are able to devise better network architecture design strategies. If compared with general data path-based strategies, the architecture designed by data path strategy usually requires additional parameter or computational cost to achieve better accuracy. In contrast, the three proposed architectures based on gradient path design strategy can significantly improve the overall performance.

5. Conclusions

In this paper we propose a strategy for designing network architectures with gradient paths. We propose three different gradient path design strategies and these strategies confirm that no matter designing from layer-level, stage-level, or network-level, it can effectively and comprehensively improve the network architecture to achieve great learning ability. Compared with data path-based design strategies, data path-based strategy often needs to design additional computing units and complex topology to achieve better learning results. As for gradient path design strategies, it can completely rely on the existing computing units, and re-planning through a simple gradient path can reduce the amount of parameters, computing, hardware resources, and improve the inference speed and network learning effect simultaneously. In this paper we redefine the strategy for designing a network and create an effective and concise architectural design rule.

6. Acknowledgements

The authors wish to thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017. 2

[2] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11030–11039, 2020. 2

[3] Jocher Glenn. YOLOv5 release v6.2. https://github.com/ultralytics/yolov5/releases/tag/v6.2, 2022. 9, 10

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1904–1916, 2015. 2

[6] 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 (CVPR)*, pages 770–778, 2016. 1, 3, 5, 6, 8

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 630–645. Springer, 2016. 5

[8] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 1

[9] 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. 1, 7

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. 1, 6

[11] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proceedings of the European conference on computer vision (ECCV)*, pages 646–661. Springer, 2016. 8

[12] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size. *arXiv preprint arXiv:1602.07360*, 2016. 1

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015. 4

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

[15] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 4

[16] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 1

[17] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and GPU-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop)*, 2019. 5, 9

[18] Youngwan Lee and Jongyoul Park. CenterMask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5

[19] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–519, 2019. 2

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature

pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017. 2

[21] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 385–400, 2018. 2

[22] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNetV2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. 1

[23] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 8, 9

[24] 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 (CVPR)*, pages 4510–4520, 2018. 1

[25] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 1

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

[27] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017. 4

[28] 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 (CVPR)*, pages 1–9, 2015. 1, 4

[29] 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 (CVPR)*, pages 2818–2826, 2016. 4

[30] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014. 1

[31] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, 2021. 5

[32] Chien-Yao Wang, Hong-Yuan Mark Liao, Ping-Yang Chen, and Jun-Wei Hsieh. Enriching variety of layer-wise learning information by gradient combination. *Proceedings of the IEEE International Conference on Computer Vision Workshop (ICCV Workshop)*, 2019. 2, 3, 9

[33] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of CNN. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop)*, 2020. 2, 4, 5

[34] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *arXiv preprint arXiv:2105.04206*, 2021. 5, 9, 10

[35] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017. 1, 7, 8

[36] Fan Yang, Cheng Lu, Yandong Guo, Longin Jan Latecki, and Haibin Ling. Dually supervised feature pyramid for object detection and segmentation. *arXiv preprint arXiv:1912.03730*, 2019. 1

[37] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 191–207. Springer, 2020. 3

[38] 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 (CVPR)*, pages 6848–6856, 2018. 1

[39] Ligeng Zhu, Ruizhi Deng, Michael Maire, Zhiwei Deng, Greg Mori, and Ping Tan. Sparsely aggregated convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 186–201, 2018. 6