Adaptive Structured Sparse Network for Efficient CNNs with Feature Regularization

Chen Tang*, Wenyu Sun, Zhuqing Yuan, Guijin Wang, Yongpan Liu

*Department of Electronic Engineering
Tsinghua University
*tangc20@mails.tsinghua.edu.cn

Abstract

Neural networks have made great progress in pixel to pixel image processing tasks, e.g. super resolution, style transfer and image denoising. However, recent algorithms have a tendency to be too structurally complex to deploy on embedded systems. Traditional accelerating methods fix the options for pruning network weights to produce unstructured or structured sparsity. Many of them lack flexibility for different inputs. In this paper, we propose a Feature Regularization method that can generate input-dependent structured sparsity for hidden features. Our method can improve sparsity level in intermediate features by 60% to over 95% through pruning along the channel dimension for each pixel, thus relieving the computational and memory burden. On BSD100 dataset, the multiply-accumulate operations can be reduced by over 80% for super resolution tasks. In addition, we propose a method to quantitatively control the level of sparsity and design a way to train one model that supports multi-sparsity. We identify the effectiveness of our method for pixel to pixel tasks by qualitative theoretical analysis and experiments.

1 Introduction

Pixel to pixel problems are a set of tasks that transforms the original image to the targeted image by acting on pixels directly. Through deep neural networks, latent characteristics can be better extracted and their important information can be highlighted. In order to outperform others’ works in the performance competition, there is a tendency to train a deeper and more complex neural network structure. For a typical pixel to pixel neural network like super resolution tasks, number of parameters changes from less than 1M like FSRCNN[9] and VDSR[17] to more than 40M like EDSR[23] and OISR-RK3[12]. In addition, it reduces the real-time performance of the algorithm. For applications like video super resolution, time cost to process one frame ranges from 200ms to 1500ms[40], which is far more than the popular adopted video frame rate. Therefore, methods should be proposed to accelerate computation and save capacity burden especially for pixel to pixel tasks.

Traditional algorithms for vision tasks utilize dense convolution and equal amount of computation is given to all the pixels. This implies a risk of waste. By analyzing traits of pixel to pixel problems, there exists potential to generate sparsity in intermediate layers’ outputs manually due to the spatial redundancy and upper limit of information one pixel can offer. That is, areas composed of similar pixels can be actived with fewer channels while the opposite ones more. Therefore, pixels in background or in areas with similar color or texture will save numerous convolution operations. In order to distinguish which pixel can save more channels. We utilize the fitting ability of neural networks to measure how much extra information one pixel can contain in addition to neighboring pixels. Pixels with less worth can abandon more channels. Then, sparsity along channel dimension at different location is formed and specific hardware architecture can be designed to relieve computational and memory burden.
We call our proposed method FR for Feature Regularization. There are two modules in FR, the Feature Map module and Mask Generating module. The Feature Map module yields a feature map to judge the redundancy and necessity level for each pixel by analyzing the local similarity. Then we design a mechanism to rescale the map to make one model support multi-sparsity. For Mask Generating module, it takes the feature map as input and generates a binarized mask which have the same size with the input tensor. Sparsity for each location is guided by the mask. It is worth mentioning that we make the discarded channels adjacent so that the sparsity is structured. Our FR shares parameters, which means all the intermediate results can use same importance map to cut their channels. It is obvious that our FR treats different input images differently, which shows its flexibility and difference from previous works.

In conclusion, we design a method to discard the unimportant channels for each location, thus generating structured sparsity and relieving the storage burden. Also, calculation related to these channels can be saved, thus relieving the computation burden and energy usage. In addition, since the unactivated channels are essentially controlled by the input tensor and the generated mask, we design a model that can dynamically adjust to the wanted sparsity level. Experiments show that up to 60%-95% channels can be discarded.

We present our experiment on the single image super resolution (SISR), which is a typical pixel to pixel problem. We evaluate our method on multiple benchmark datasets for SISR. We show that: 1) with the application of our method, the results can be comparable with those of the non-sparse one, which means the deterioration of performance when applying our method is tiny. 2) Our method can cut the multiply-accumulate operations of models dramatically, which can be presented by measuring the discarding rates. 3) We put forward a method to train one model that supports multi-sparsity.

2 Related Work

In this part, we will introduce the necessity of acceleration methods for pixel to pixel algorithms and several related trials.

To reduce the burden of storage and computation, many methods have been proposed. For acceleration of computation, researchers make use of sparsity or apply algorithms like Winograd convolution that run faster on hardware. Xingyu Liu [25] overcomes the drawbacks of Winograd to avoid undermining sparsity of hidden layers. Vanhoucke [36] tries to speed up on CPU devices by data layout. Mathieu [28] detects how to realize Fast Fourier Transformation on GPU devices. In addition, other methods try to compress the model. Denil [6] can use a few of weights to predict the whole model. Denton [7] uses matrix decomposition to compress the connection. [15, 24] try to quantize the weights so that not only the model is compressed, but also it is more suitable for mobile platforms to run. Researchers have proposed methods to quantize the network during or after the training. Even methods of binarized quantization is proposed [5].

Network pruning aims to accelerate the inference of deep learning by removing the redundant connections in the model, thus generating sparsity for intermediate data flow. Many works consider this problem from the view of pruning weights or connecting sensitivity [13]. Early works performs unstructured pruning [32, 11], which results in non-structured sparsity of the network. Recently works pay more attention to the structured pruning [19, 13, 26], especially pruning weight channels. Li et al. [19] prune model according to the $l_1$-norm of weight channels. He et al. [13] minimize the local layer output reconstruction error to prune channels. Liu et al. [26] use LASSO regularization to learn the importance of all channels and prunes the model based on a global threshold. As these works only consider the significance of weights for pruning, the pruned model has a same sparse degree regardless of the input. We propose an activation-based sparse training method that can cut the data flow according to an feature map of input.

Many works have generated an inference map to get accelerated. Li found the difference in difficulty to classify pixels in different areas [22]. Then, pixels are first separated into 3 classes and each class will go through different number of layers to get their label in semantic segmentation. Figurnov found that only tiny difference can be made if we skip arbitrary intermediate layer in ResNet [10]. Inspired from this, a halting score generated by input of current layer is devised for whether jump over this layer or not. Veit attached each layer with a gate, which represents the probability to skip the corresponding layer [37]. The probability is calculated by the dependency level for channels in current layers. Our method is different from them in that we do not discard layers. All the
pixels in our method will go through same number of layers with different number of channels active. Ren separated the image into blocks and only block in foreground can be applied with dense convolution\cite{29} while our method is pixel-wise sparsity. Xie proposed a method to divide pixels into two classes\cite{39}. One of them use convolution operation to go forward. Value of the other one is calculated by interpolation, thus relieving the computation burden. What differs our work from them is that we divide pixels into more classes. For different class different number of channels will be left active while previous method only has two classes and all the channels are active.

3 Methodology

3.1 Feature Regularization

Most pixel to pixel algorithms include architectures like feature extraction modules, scale transformation modules and some other manual ones. We call data in the hidden layers, which implicitly or explicitly contains image characteristics, the feature batch. We believe that when a pixel has different characteristics compared to its neighbors, it contains more unique information and should be left with more channels active. In this paper, we show our method can significantly reduce the computation burden by make areas composed of identical pixels have fewer channels active. We present our methodology in Fig1. Functional network refers to structure that complete the certain task, like ResNet. The binary mask is generated from input images, guided by which different locations can discard different number of channels.

Our method mainly refers to the Mask network. To save calculation, the binary mask is generated from a rather light-weight network. Also, the Mask is applied to outputs of all the intermediate layers so as to discard channels at pixel wise. Our method mainly consists of two parts. Feature Map module aims at generating the importance map, called FMAP for each location. The FMAP measures how much distinct information one pixel can offer. Mask Generating module is designed to change FMAP into a binarized mask, which determines the number of active channels for each location. The Mask has the same size with the feature batch. We downsize the FMAP by pooling or interpolation directly if the feature batch goes through downsample operations. It is worth mentioning that the sparse area of our Mask is continuous. To put it more vividly, the black part of the Mask shown in Fig1-(a) is nonzero.

We will first present our implementation details. Then, some theoretical analyses will be given to explain why it is effective.

3.2 Mask Generating

Let $I_{in} \in \mathbb{R}^{C \times H \times W}$ denotes the input data. We use several convolutional layers to fit the generation of FMAP. Let $\mathcal{H}(\cdot)$ denotes the fitting procedure, and we have $\text{FMAP} = \mathcal{H}(I_{in}) \in \mathbb{R}^{1 \times H \times W}$, which shows the importance level for each pixel. It serves as the input of the next Mask Generating module. The value of FMAP measures the necessity of the corresponding pixel and is in the range of (0, 1), for which the larger one means higher importance. Mask Generating module yields binary mask from FMAP, as is shown in Fig2.
We design three kinds of structure to generate Mask and call them $F_{M_{\text{hard}}}$, $F_{M_{\text{soft}}}$ and $F_{M_{\text{sigmoid}}}$. The implementation of $F_{M_{\text{hard}}}$ mode is inspired from how Li quantize pixels with different bits in image compression task\cite{21}. Since the binary calculation has no gradient, we take tricks from quantization by adding noise\cite{2}. Parameter $n$ refers to the number of classes, noted that pixels with same class will have same number of channels active. A guidance $G$ is defined as

$$G_{i,j} = \begin{cases} l - 1, & \frac{l - 1}{n} \leq \text{FMAP}_{i,j} < \frac{l}{n}, l = 1, \ldots, n \\ n, & \text{FMAP}_{i,j} == 1. \end{cases}$$

As we see in the formula above, each location can be divided into $n + 1$ kinds. The targeted Mask is in the form of $C \times H \times W$ and is generated under the guidance of $G$ as

$$\text{Mask}_{k,i,j} = \begin{cases} 1, & k \leq G_{i,j} \times \lfloor \frac{C}{n} \rfloor \\ 0, & \text{otherwise}, \end{cases}$$

where $C$ denotes the upper limit number of channels. We multiply feature batches by the Mask and achieve the goals of cutting unimportant channels at each pixel. When it comes to back-propagation, the gradient is defined as

$$\frac{\partial \text{Mask}_{k,i,j}}{\partial \text{FMAP}_{i,j}} = \begin{cases} n, & (G_{i,j} - 1) \times \lfloor \frac{C}{n} \rfloor \leq k \leq G_{i,j} \times \lfloor \frac{C}{n} \rfloor \\ 0, & \text{otherwise} \end{cases}$$

For $F_{M_{\text{hard}}}$, we just realize it under the guidance of Equation\cite{1,2} and \cite{3}.

For $F_{M_{\text{soft}}}$, we use exponent function to make the sparsity degree drop more easily. Firstly, we generate a modified distance to the stratification defined in Equation\cite{1} as

$$P_i = e^{-k_1 \times (\text{FMAP} - \frac{1}{n})}, i = 0 \ldots n.$$ Then we have $\text{Prob} = \frac{\text{FMAP}}{(\sum_{i=0}^{n} P_i)}$. $P_i$ means the i-th channel of P and $k_1, k_2$ are predefined parameters. Multiplication and Division are operated in the pixel wise at corresponding position. We get the Mask by putting $\text{Prob}$ into Equation\cite{1} instead of $\text{FMAP}$ and subsequent formulas.

For $F_{M_{\text{sigmoid}}}$, the Mask can be simply generated as

$$\text{Mask}(|\frac{F}{C}|_{s=t\sim|\frac{F}{C}|_{s=t+1}}) = \text{Sign}(2 * \text{Sigmoid}(\alpha * (\text{FMAP} - \frac{l + 0.5}{n}))) - 1),$$

where $\alpha$ is the pre-defined parameter. Experiments show that $F_{M_{\text{sigmoid}}}$ also works well.

During training, the task can be defined as

$$\min \mathcal{L}(F \odot \text{Mask}) \text{ s.t. } \text{Mask} = \mathcal{B}(I_{in}),$$

where $F$ means all the hidden layers’ output, $\odot$ denotes the Hadamard product operator and $\mathcal{B}(\cdot)$ means the upper bound channel index of nonzero value in the Mask. $\mathcal{L}(\cdot)$ is the loss function of the network. We call loss function which meets certain tasks’ purpose functional loss $\mathcal{L}_f$. Different pixel to pixel tasks have different $\mathcal{L}_f$. For super resolution problems, $L_1$ loss can contribute to a high Peak Signal to Noise Ratio (PSNR)\cite{31} index. For image style transfer problems, perceptual loss\cite{10} can construct better lifelike images. Then we add a $L_1$ regularization part as our bit rate loss $\mathcal{L}_b = \sum_{k,i,j} |\text{Mask}_{k,i,j}|$. The total loss is defined as

$$\mathcal{L} = \mathcal{L}_f + \alpha * |\mathcal{L}_b - \gamma|,$$

where $\alpha$ is pre-defined parameter and $\gamma$ is the wanted sparsity. We can simply set $\gamma$ with zero to let the bit rate converge freely. Through this loss function, we can train the Mask Generating module and other modules jointly. Implementation details will show in supplementary materials.
3.3 Pixel Wise Sparsity

In this part, we will explain qualitatively why FMAP can be generated through feature regularization. First, we will show why more channels can lead to more precise results. Given a one channel input image $x$ and one channel targeted output $GT$, we have intermediate output $y$ as $y_k = F_{\theta,k}(x), k = 0, 1, ..., BM(A)$. Output of the whole model is $z = \sum_{k=0}^{BM(x)} R_{\phi}(y_k)$, which $\theta$ and $\phi$ means parameters of $F(.)$ and $R(.)$ representing the processing procedures. For typical pixel to pixel problems, the goal is to match $z$ and the ground truth $GT$ as close as possible. If we see the problem from an aspect of probability. We can use $KL$ divergence to measure the proximity of two variable distributions. The task can be explained as making $KL(P(GT))|P_{\theta}(z|y_0, y_1, \ldots y_{BM(x)})$ as small as possible, which $P(.)$ means the probability distribution. For generalization, we have

$$
\begin{align*}
E_y(KL(P(GT))|P_{\theta}(z|y_0, y_1, \ldots y_{BM(x)})) & = \\
E_{GT}(E_y(\log P(y_0, y_1, \ldots y_{BM(x)})) & + \\
E_y(E_{GT}(\log P(GT) - \log P_{\phi}(z)) & + irrelevant const
\end{align*}
$$

(7)

The second part is the weighted distortion between $P(GT)$ and $P(z)$, which directly affects the similarity between output of algorithms and the targeted image. The first part shows that the number of active channels does affect the final accuracy. It can be seen that the greater $BM(x)$ is, the less the first part will be and the distribution of $z$ will be more like $GT$. Also, the impact differs for different pixels, which gives the room for sparsity and cut channels.

Secondly, we will explain that our training strategy can treat pixels differently according to their local diversity, thus generating an importance map. Since the channel importance is difficult to measure and the abandon of existing channels may affect the redundancy of local areas of images, we conduct a learning process and update the importance map dynamically. Taking one targeted place (denote as $tp$) of pixel in $x$ as an example, the intermediate output of $y_{k,tp}$ can be formed as $y_{k,tp} = F_{k,\theta,tp}(x_{tp}) + F_{k,\theta,rf}(x_{rf})$, which $rf$ means receptive field. The output of whole network can be denoted as

$$
\begin{align*}
BM(z)_{tp} & = \sum_{k=0}^{BM(x)_{tp}} R_{\phi,tp}(y_{k,tp}) + \sum_{k=0}^{BM(x)_{rf}} R_{\phi,rf}(y_{k,rf}) \\
BM(z)_{tp} & = \sum_{k=0}^{BM(x)_{tp}} T_{\phi,tp}(x_{k,tp}) + \sum_{k=0}^{BM(x)_{rf}} T_{\phi,rf}(x_{k,rf}) \\
BM(z)_{tp} & = T_{\phi,tp}(x_{tp}) * Mask_{tp} + T_{\phi,rf}(x_{rf}) * Mask_{rf},
\end{align*}
$$

(8)

where $T_{\phi}(.)$ refers to the whole process. If we only have functional loss and set $L(.) = L_f$, then we reach

$$
\frac{\partial L}{\partial w} = \frac{\partial L_f}{\partial z_{tp}} * \frac{\partial z_{tp}}{\partial w}
$$

(9)

$$
\frac{\partial z_{tp}}{\partial w} = \frac{\partial T_{\phi,tp}}{\partial w} * Mask_{tp} + \frac{\partial T_{\phi,rf}}{\partial w} * Mask_{rf} + T_{\phi,tp} * \frac{\partial Mask_{tp}}{\partial w} + T_{\phi,rf} * \frac{\partial Mask_{rf}}{\partial w}
$$

(10)

Equation (10) and (11) tell us that the number of channels will certainly increase to its upper bound if we only set the functional loss $L_f$. Therefore, we need to set another loss function to make the Mask Network generate a sparser mask. We add a L1-regularization part as our bit rate loss $L_b = \sum_{k,i,j} |Mask_{k,i,j}|$ and we have

$$
\begin{align*}
\frac{\partial L_b}{\partial w} & = \frac{\partial L_b}{\partial Mask_{tp}} * \frac{\partial Mask_{tp}}{\partial w} + \frac{\partial L_b}{\partial Mask_{rf}} * \frac{\partial Mask_{rf}}{\partial w}
\end{align*}
$$

(11)

If we only pick the part related to Mask, we can have

$$
\frac{\partial z_{tp}}{\partial w} = \frac{\partial T_{\phi,tp}}{\partial w} * Mask_{tp} + \frac{\partial T_{\phi,rf}}{\partial w} * Mask_{rf} + T_{\phi,tp} * \frac{\partial Mask_{tp}}{\partial w} + T_{\phi,rf} * \frac{\partial Mask_{rf}}{\partial w}
$$

(12)

During backward propagation, if local $x_{tp}$ and $x_{rf}$ shows similar characteristics, their gradient will behave similarly, which means their value of FMAP will be close and drops sharply together.
When they differ greatly, their corresponding value in FMAP will drop in different speeds, which contributes to the heterogeneity in number of channels active. Therefore, areas composed of diverse pixels can keep some pixels more channels than others.

In conclusion, \( L_b \) leads to the reduction of bit rate of data batch while \( L_f \) tends to keep more redundant information and thus increase the bit rate. It is a tradeoff between sparsity and performance. We can set \( \alpha \) and \( \gamma \) to change the degree of sparsity we need. It is worth mentioning that the training of formulation of feature mask and the training of generating targeted image are done jointly.

\[
L = L_f + \alpha_1 |L_b - \gamma| + \alpha_2 |avg(pdf \odot V) - \beta| \tag{13}
\]

It is like supervised learning with \( Q \) and \( \beta \). Through the adjustment of FMAP, we can generate the Mask with different sparsity degree by changing \( Q \). Also, we can set the changing rate of \( V \) while training. The total procedure is summarized in algorithm 1.

**Algorithm 1 Feature Sparse Regularization: Sparsity for Pixel to Pixel Problems**

**Input:** \( I \): training images, \( \hat{W}_f \): parameters from Mask Network, \( \hat{W}_\varphi \): parameters from functional network, \( Q \): the degree of wanted sparsity, \( Q_s \): range of \( Q \).

**Output:** \( W_f \): updated parameters from Mask Network, \( W_\varphi \): updated parameters from functional network.

1: initialize \( W_f \leftarrow \hat{W}_f \), \( W_\varphi \leftarrow \hat{W}_\varphi \) and \( \text{iter} \leftarrow 0 \)
2: repeat
3: Choose a minibatch of network input from \( I \), generate a random \( Q \) from \( Q_s \)
4: Generate the Mask from Mask Network
5: Through Forward propagation, apply Mask to output from hidden layers.
6: Map \( Q \) to targeted \( \gamma \), update \( L \) by Equation 13, backward propagation and generate \( \Delta L \)
7: Update \( W_f \) and \( W_\varphi \) by the current loss function gradient \( \Delta L \)
8: until
9: \( \text{iter} \) reaches desired maximum

3.4 One Model for Multi-sparsity

For hardware realization, it will be interesting if the sparsity degree can be adaptively adjusted to match the computing and storage capacity of the embedded systems. Training different models for different requirements respectively is inconvenient. Therefore, we design a method to train a model that supports different sparsity degree. The Mask derives from FMAP, adjustment of which can contribute to different level of sparsity. We find that pixels share same \( G \) in equation 1 are aggregational for space. Then we devise an adjustment module to revise the FMAP, as Fig 3 shows. Input is the statistic property of original FMAP named as pdf, which measures the frequency of pixels with same \( G \), and an integer \( Q \) which represents the sparsity degree and \( \beta \) is mapped from \( Q \). Output is an array \( V \) with length of \( n \), in which value means a rescaling for pixels in FMAP with same . The loss function is adjusted as

![Figure 3: Structure for Multi-sparsity Control](image)
4 Experiment

4.1 Baseline and Model Training

To demonstrate the effectiveness of our network structure, we apply our Feature Regularization modules on the classic Single Image Super Resolution(SISR) problem, which aims at reconstructing high resolution images from low resolution images. There are many artilf designs of algorithms in this field. For fair comparison, we refer to the most ordinary one to build our baseline, which only has several convolutional layers and the common pixel-shuffle method\cite{30}. The baseline is like SRCNN\cite{8}, which is the first CNN-based method in this area, with adequate more convolutional layers. We apply mask generated by our FR on the intermediate output to form our model. Our experiments are trained on dataset DIV2K\cite{35}. We test our performance on four standard datasets, including Set5\cite{3}, Set14\cite{41}, B100\cite{27} and Urban100\cite{14}. The four datasets are universally used as benchmarks for super resolution problems. As for the evaluation method, we use PSNR and structural similarity index(SSIM)\cite{38} to measure the distortion of the recovered images. Also, we estimate Multiply accumulate Operations (MACs) of the SISR algorithm to judge the computing burden. We set upscaling factors for X2 and X4 to train and test our model.

First, we train five models with different rate weights($\alpha_1=4e-6, 2e-5, 1e-4, 5e-4, 2.5e-3$, $\alpha_2 = 0$). We let the model freely drop to a stable low rank of sparsity by setting $\gamma = 0$. To describe the results more clearly, we denote the original SR method by Res-SR and call the model with our FR Mask-SR. The only difference between two is that Mask-SR applies FR to cut channels at every intermediate output. Specifically, we set $n = 16$ in equation and the upper bound number of channels is 128. We set convolutional layers for Res-SR is 24.

![Figure 4: Baseline Model for Super Resolution](image)

4.2 Comparison with SR Methods

We compare our work with six state-of-the-art SR methods: FSRCNN\cite{9}, DRRN\cite{33}, MemNet\cite{34}, SelNet\cite{4}, CARN\cite{1}, OISR-RK2-s, OISR-LF-s\cite{12} and MSRN\cite{20}. MACs of our SR algorithm with our method mainly depends on the basic network we choose. As it is shown in Tab1, we achieve most of the better performance for SSIM and have comparable results in PSNR between methods that have same level of MACs. The interaction between MACs and PSNR, MACs and SSIM test on B100 is shown in Fig5, which infers that simple design with our method can outperform those delicate designed ones when we take the tradeoff into account. Our SSIM generally can have a 0.01-0.1 gain. Moreover, our method can be easily applied on other algorithms as we add sparsity to the universally used convolutional layers. Our FR can help other algorithms further reduce their number of MACs. As is shown in Tab1 Mask-SR saves MACs dramatically compared with Res-SR, which results from an increase in sparsity. The biggest three rate weights models remain only 20% and the $\alpha = 2.5e-3$ one further drops 90% of the calculation. Furthermore, it does not distort the result much. In our experiment, we show a 0.1dB decline in PSNR and a 0.01 decline in SSIM exchanges for over 80% reduction of MACs. With little drop in performance, our FR saves huge computation burden. Therefore, it is worthwhile to apply our method to reduce the storage and computation burden.
We train a model that supports four kinds of sparsity degree. We train two models with different \( \gamma \) and mapping strategy between \( Q \) and \( V \) in Equation 13. Performance on datasets BSD100 with upscale factor X2 is shown on Fig6. As it can be seen, one model that supports multi-sparsity will sacrifice some performance for extension of range in sparsity. Generally speaking, the sparsity degree is controlled by the \( F_{\text{map}} \), which is synergistically generated as is shown in Fig6.
Figure 6: Results for Multi-sparsity Control from One Model

4.4 Discussion
We show examples of $F_{map}$ in Fig[7]. It can be seen that the Feature Map module extracts the details and complex texture areas of the image. Under the guidance of $F_{map}$, we can allow the discarding of some values along the channel dimension. Our experiment shows that the performance between each rate weights has little difference. However, models with bigger $\alpha$ in equation discard more channels, which should have distorted the results more. This phenomenon implies that when we make our models deeper and wider, we mainly increase unnecessary redundant information. We believe our Feature Regularization can give algorithm designer a new view towards network structure and can help design a more efficient model.

Figure 7: Input Image and Their $F_{map}$

5 Conclusion
In this paper, we propose a Feature Regularization method to generate structured sparsity in CNN-based models, which aims to apply on pixel to pixel application scenarios. Through abandoning values along the channel dimension at activation-base, models can run faster in their hardware implementation. In addition, the sparsity is input dependent, which assures its generalization ability. We also design a training mechanism to train a model which is capable for adjusting sparsity level. We verify the effectiveness of our method on classical pixel to pixel tasks super resolution. Experiments show that our method can have less computational complexity without sacrificing the precision of results. Since most of the pixel to pixel Neural Network shares similar blocks and structure, we believe our Feature Regularization can be universally applied for pixel to pixel problems to relieve the computing burden of hardware.

6 Broader Impact
Our Feature Regularization can be combined with other modules easily. We just need to generate a corresponding mask of the input data batch and apply multiplication with intermediate feature batches, as shown in Fig[1]. It can reduce the computation burden because there is no need to calculate at the position where channels are discarded and the storage burden will be also relieved. Our work
is inspired from Mu Li’s Content-weighted Image Compression[21], which increases compression rate by applying different Quantization bits to different pixels. However, We have different training strategy and usage. We identify that our FR has potential for all the pixel to pixel tasks. It contributes to the re-simplifying of each data patch at the pixel level. In addition, we design a method for multi-sparsity model, which also differs our work from them.

Traditional methods focus on pruning weights and generating sparsity, which lacks flexibility for different kinds of inputs. Our method can provide a different view from pruning channels and can generate sparsity according to the input image. However, we only test its effectiveness for super resolution problems. Whether other pixel to pixel problems can apply our FR remains in the theoretical stage. As more and more algorithms design networks that have tremendous channels, FR can mend the consequence of huge number of parameters brought by such design concept.

References

[1] Ahn, N., Kang, B., Sohn, K.: Fast, accurate, and lightweight super-resolution with cascading residual network pp. 256–272 (2018)
[2] Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression (2016)
[3] Bevilacqua, M., Roumy, A., Guillemin, C., Alberimorel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding pp. 1–10 (2012)
[4] Choi, J., Kim, M.: A deep convolutional neural network with selection units for super-resolution pp. 1150–1156 (2017)
[5] Courbariaux, M., Bengio, Y., David, J.: Binaryconnect: training deep neural networks with binary weights during propagations pp. 3123–3131 (2015)
[6] Denil, M., Shakibi, B., Dinh, L., Ranzato, M., De Freitas, N.: Predicting parameters in deep learning pp. 2148–2156 (2013)
[7] Denton, E., Zaremba, W., Bruna, J., Lecun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation pp. 1269–1277 (2014)
[8] Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Trans Pattern Anal Mach Intell 38(2), 295–307 (2016)
[9] Dong, C., Loy, C.C., Tang, X.: Accelerating the super-resolution convolutional neural network. In: European conference on computer vision. pp. 391–407. Springer (2016)
[10] Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1790–1799 (2017)
[11] HASSIBI, B.: Second order derivatives for network pruning: Optimal brain surgeon. Advances in Neural Information Processing Systems 5 (1993)
[12] He, X., Mo, Z., Wang, P., Liu, Y., Yang, M., Cheng, J.: Ode-inspired network design for single image super-resolution pp. 1732–1741 (2019)
[13] He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. arXiv: Computer Vision and Pattern Recognition (2017)
[14] Huang, J., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars pp. 5197–5206 (2015)
[15] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference pp. 2704–2713 (2018)
[16] Johnson, J., Alahi, A., Feifei, L.: Perceptual losses for real-time style transfer and super-resolution pp. 694–711 (2016)
[17] Kim, J., Lee, J.K., Lee, K.M.: Accurate image super-resolution using very deep convolutional networks pp. 1646–1654 (2016)
[18] Lee, N., Ajanthan, T., Torr, P.H.S.: Snip: Single-shot network pruning based on connection sensitivity (2019)
[19] Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv: Computer Vision and Pattern Recognition (2016)
[20] Li, J., Fang, F., Mei, K., Zhang, G.: Multi-scale residual network for image super-resolution pp. 527–542 (2018)
[21] Li, M., Zuo, W., Gu, S., Zhao, D., Zhang, D.: Learning convolutional networks for content-weighted image compression pp. 3214–3223 (2018)
[22] Li, X., Liu, Z., Luo, P., Loy, C.C., Tang, X.: Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6459–6468 (2017)
[23] Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution (2017)
[24] Lin, D.D., Talathi, S.S., Annapureddy, V.S.: Fixed point quantization of deep convolutional networks pp. 2849–2858 (2016)
[25] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. arXiv: Computer Vision and Pattern Recognition (2017)
[26] Martin, D., Fowlkes, C.C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics 2, 416–423 (2001)
[27] Mathieu, M., Henaff, M., Lecun, Y.: Fast training of convolutional networks through ffts. arXiv: Computer Vision and Pattern Recognition (2013)
[28] Ren, M., Pokrovsky, A., Yang, B., Urtasun, R.: Sbnnet: Sparse blocks network for fast inference (2018)
[29] Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network pp. 1874–1883 (2016)
[30] Soh, J.W., Park, G.Y., Jo, J., Cho, N.I.: Natural and realistic single image super-resolution with explicit natural manifold discrimination pp. 8122–8131 (2019)
[31] Song, H., Liu, X., Mao, H., Jing, P., Dally, B.: Deep compression and eie: Efficient inference engine on compressed deep neural network. In: 2016 IEEE Hot Chips 28 Symposium (HCS) (2016)
[32] Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3147–3155 (2017)
[33] Tai, Y., Yang, J., Liu, X.: Memnet: A persistent memory network for image restoration pp. 4549–4557 (2017)
[34] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13(4), 600–612 (2004)
[35] Xie, Z., Zhang, Z., Zhu, X., Huang, G., Lin, S.: Spatially adaptive inference with stochastic feature sampling and interpolation. arXiv e-prints (2020)
[36] Yi, P., Wang, Z., Jiang, K., Jiang, J., Ma, J.: Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations (12 2019). https://doi.org/10.1109/ICCV.2019.00320
[37] Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations pp. 711–730 (2010)