Supplementary Material

Training CNNs with Selective Allocation of Channels

A. Detailed Derivation of ECDM Formula

Consider a convolutional layer $\mathrm{Conv}(\mathbf{X};\mathbf{W})$ with $\mathbf{W}\in\mathbb{R}^{I\times O\times K^2}$, and $\mathbf{X}\in\mathbb{R}^{I\times H\times W}$. First, from the assumption that $\mathbf{X}_{i,h,w}\sim\max(\mathcal{N}(\beta_i,\gamma_i^2),0)$ for all i,h,w, we get:

$$\begin{aligned}
\mathbb{E}[\mathbf{X}_{i,h,w}] &= \int_{-\infty}^{\infty}\frac{\max(x,0)}{|\gamma_i|\sqrt{2\pi}}\cdot\exp\left(-\frac{(x-\beta_i)^2}{2\gamma_i^2}\right)dx\\
&= \int_{0}^{\infty}\frac{x}{|\gamma_i|\sqrt{2\pi}}\cdot\exp\left(-\frac{(x-\beta_i)^2}{2\gamma_i^2}\right)dx\\
&= \int_{-\frac{\beta_i}{|\gamma_i|}}^{\infty}\frac{|\gamma_i|y+\beta_i}{\sqrt{2\pi}}\cdot\exp\left(-\frac{y^2}{2}\right)dy\\
&= \frac{|\gamma_i|}{\sqrt{2\pi}}\int_{-\frac{\beta_i}{|\gamma_i|}}^{\infty}y\cdot\exp\left(-\frac{y^2}{2}\right)dy+\beta_i\Phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right)\\
&= |\gamma_i|\phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right)+\beta_i\Phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right)=:f(\mathbf{X})_i,
\end{aligned}\tag{1}$$

where $\phi_{\mathcal{N}}$ and $\Phi_{\mathcal{N}}$ denote the p.d.f. and the c.d.f. of the standard normal distribution, respectively.

Secondly, once noticed that $\mathrm{Conv}(\mathbf{X};\mathbf{W})-\mathrm{Conv}(\mathbf{X};\mathbf{W}_{-i})$ in the definition of ECDM is identical to $\mathrm{Conv}(\mathbf{X}_i;\mathbf{W}_{i,:,:})$, the desired formula follows from the linearity of convolutional layer, the linearity of expectation, and (1):

$$\begin{aligned}
\mathrm{ECDM}(\mathbf{W};\mathbf{X})_i &\coloneqq \frac{1}{HW}\sum_{h,w}\mathbb{E}_{\mathbf{X}}[\mathrm{Conv}(\mathbf{X};\mathbf{W})-\mathrm{Conv}(\mathbf{X};\mathbf{W}_{-i})]_{:,h,w}\\
&= \frac{1}{HW}\sum_{h,w}\mathbb{E}[\mathrm{Conv}(\mathbf{X}_i;\mathbf{W}_{i,:,:})]_{:,h,w}\\
&= \frac{1}{HW}\sum_{h,w}\left(\mathbb{E}\left[\sum_{x=-\lfloor\frac{K}{2}\rfloor}^{\lfloor\frac{K}{2}\rfloor}\sum_{y=-\lfloor\frac{K}{2}\rfloor}^{\lfloor\frac{K}{2}\rfloor}\mathbf{W}_{i,j,\left(\lfloor\frac{K}{2}\rfloor+x\right)\cdot K+\left(\lfloor\frac{K}{2}\rfloor+y\right)}\cdot\mathbf{X}_{i,h+x,w+y}\right]\right)_{j=1}^{O} \qquad (2)\\
&= \frac{1}{HW}\sum_{h,w}\left(\sum_{x=-\lfloor\frac{K}{2}\rfloor}^{\lfloor\frac{K}{2}\rfloor}\sum_{y=-\lfloor\frac{K}{2}\rfloor}^{\lfloor\frac{K}{2}\rfloor}\mathbf{W}_{i,j,\left(\lfloor\frac{K}{2}\rfloor+x\right)\cdot K+\left(\lfloor\frac{K}{2}\rfloor+y\right)}\cdot\mathbb{E}\left[\mathbf{X}_{i,h+x,w+y}\right]\right)_{j=1}^{O}\\
&= \frac{1}{HW}\sum_{h,w}\left(f(\mathbf{X})_i\cdot\sum_{k=1}^{K^2}\mathbf{W}_{i,j,k}\right)_{j=1}^{O}=f(\mathbf{X})_i\cdot\sum_{k=1}^{K^2}\mathbf{W}_{i,:,k}\\
&= \left(|\gamma_i|\phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right)+\beta_i\Phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right)\right)\cdot\sum_{k=1}^{K^2}\mathbf{W}_{i,:,k} \qquad (3)
\end{aligned}$$

Here, in (2), we assume that the convolutional layer uses the *same padding* scheme, i.e. indexing of $\mathbf{X}$ outside the pixel scope is considered to be the nearest in-scope pixel.

B. Training Details

B.1. Training Setup

We train every model via stochastic gradient descent (SGD) with Nesterov momentum of weight 0.9 without dampening. We use a cosine shape learning rate schedule (Loshchilov & Hutter, 2016) which starts from 0.1 and decreases gradually to

0 throughout the training. We set a weight decay of 10^{-4}, except for the spatial shifting biases $\mathbf{b}$ in which 10^{-5} is used instead. During the training, we call `dealloc` and `realloc` at each epoch for the half of the total epochs. This is mainly for two reasons: (a) usually, most of `dealloc` is done before that time, and (b) we found this makes the training less sensitive on the choice of γ. When a spatial bias is re-initialized, we sample a point from $[-1.5, 1.5] \times [-1.5, 1.5]$ pixels uniformly.

B.2. Datasets

CIFAR-10/100 datasets (Krizhevsky, 2009) consist of 60,000 RGB images of size 32×32 pixels, 50,000 for a training set and 10,000 for a test set. Each image in the two datasets is corresponded to one of 10 and 100 classes, respectively, and the number of data is set evenly for each of the classes. We use a standard data-augmentation scheme that is common for this dataset (Srivastava et al., 2015; Lin et al., 2013; He et al., 2016a; Huang et al., 2017), i.e. random horizontal flip and random translation up to 4 pixels. We also normalize the images in pixel-wise by the mean and the standard deviation calculated from the training set. Each model is trained for 300 epochs with mini-batch size 64.

Fashion-MNIST dataset (Xiao et al., 2017) consists 70,000 gray-scale 28×28 images, 60,000 for a training set and 10,000 for a test set. Each of the labels is associated with one of 10 fashion objects. We use the same data-augmentation scheme with that of CIFAR datasets, along with the dataset normalization scheme. For this dataset, each model is trained for 300 epochs with mini-batch size 128.

Tiny-ImageNet[1] dataset is a subset of ImageNet classification dataset (Russakovsky et al., 2015). It consists 200 classes in total, each of which has 500 and 50 images for training and validation respectively. Unlike the ImageNet dataset, each image in this dataset has the spatial resolution of 64×64. We also use the data-augmentation scheme of CIFAR datasets, but using random translation of 8 pixels due to the doubled resolution. In order to process the larger images without changing the architecture configurations, we simply doubled the stride of the first convolution in each model. In this dataset, each model is trained for 200 epochs with mini-batch size 128.

ImageNet classification dataset (Russakovsky et al., 2015) consists of 1.2 million training images and 50,000 validation images, which are labeled with 1,000 classes. For data-augmentation, we perform 224×224 random cropping with random resizing and horizontal flipping to the training images. At test time, on the other hand, 224×224 center cropping is performed after rescailing the images into 256×256. All the images are normalized in pixel-wise by the pre-computed mean and standard deviation. Each model is trained for 120 epochs with mini-batch size 128.

B.3. Architectures

DenseNet. In our experiments, we consider four DenseNet (Huang et al., 2017) models: DenseNet-40, DenseNet-100, DenseNet-121, and DenseNet-BC-190. DenseNet-40 and DenseNet-100 consist 3 *dense blocks*, each of which consists of N consecutive *dense units*, where each of N is set by 6 and 16, respectively. A dense unit is designed by a 2-layer CNN that produces $k = 12$ features, where k is called the *growth rate*. The output of each dense unit is *concatenated* to the input, which defines the main characteristic of the DenseNet architecture. There exist a average pooling layer between the dense blocks for down-sampling, and the final feature maps are pooled into 1×1 features using a global average pooling layer. Unlike Huang et al. (2017), we do not place a feature compression layer between the dense blocks for simplicity. DenseNet-121 and DenseNet-BC-190 are considered only for the ImageNet experiments and model compression experiments, respectively, and we follow the original architectures specified in Huang et al. (2017).

ResNet and ResNeXt. We also evaluate our method on pre-activation ResNet-164 (He et al., 2016b) and ResNeXt-29 ($8 \times 64d$) (Xie et al., 2017) models designed for CIFAR-10/100 datasets, and ResNet-50 for ImageNet dataset (He et al., 2016a). Similar to DenseNet, both architectures consists 3 *residual blocks*. We generally follow the model configurations specified by the original papers (He et al., 2016a;b; Xie et al., 2017), with some minor modifications for architectural simplicity. Originally, both architectures place a 1×1 convolutional layer of stride 2 between two residual blocks to perform down-sampling and doubling the number of channels. Instead of it, we place a 2×2 average pooling layer for down-sampling, and use a simple zero-padding scheme for the doubling.

LGC in CondenseNet. CondenseNet is designed upon DenseNet (Huang et al., 2017) architecture, consisting several components to improve the computational efficiency of DenseNet. We primarily focus on the *learned group convolution*

[1] https://tiny-imagenet.herokuapp.com/

(LGC) layer among the components. Architecturally, LGC is a group convolutional layer with 1×1 kernel. During training, however, LGC prunes out $\frac{3}{4}$ of its weights through 3 *condensing stages* at $\frac{1}{6}$, $\frac{2}{6}$, and $\frac{3}{6}$ of the total training epochs, $\frac{1}{4}$ for each. At each of the condensing stages, channels are scored by the ℓ^1-norm of the corresponding weights, and the pruning is done according to the scores. Also, LGC adopts channel-wise group-lasso regularization (Wen et al., 2016) during training to induce more sparsity over channels. In our experiment, we consider CondenseNet-182 model. We follow the original setting by Huang et al. (2018) for training this model and our counterpart CondenseNet-SConv-182 as well. Namely, we train them via SGD with mini-batch size 64 for 600 epochs, and dropout (Srivastava et al., 2014) with a drop rate 0.1 is used.

C. Additional Illustrations of De/Re-allocated Channels in DenseNet-40

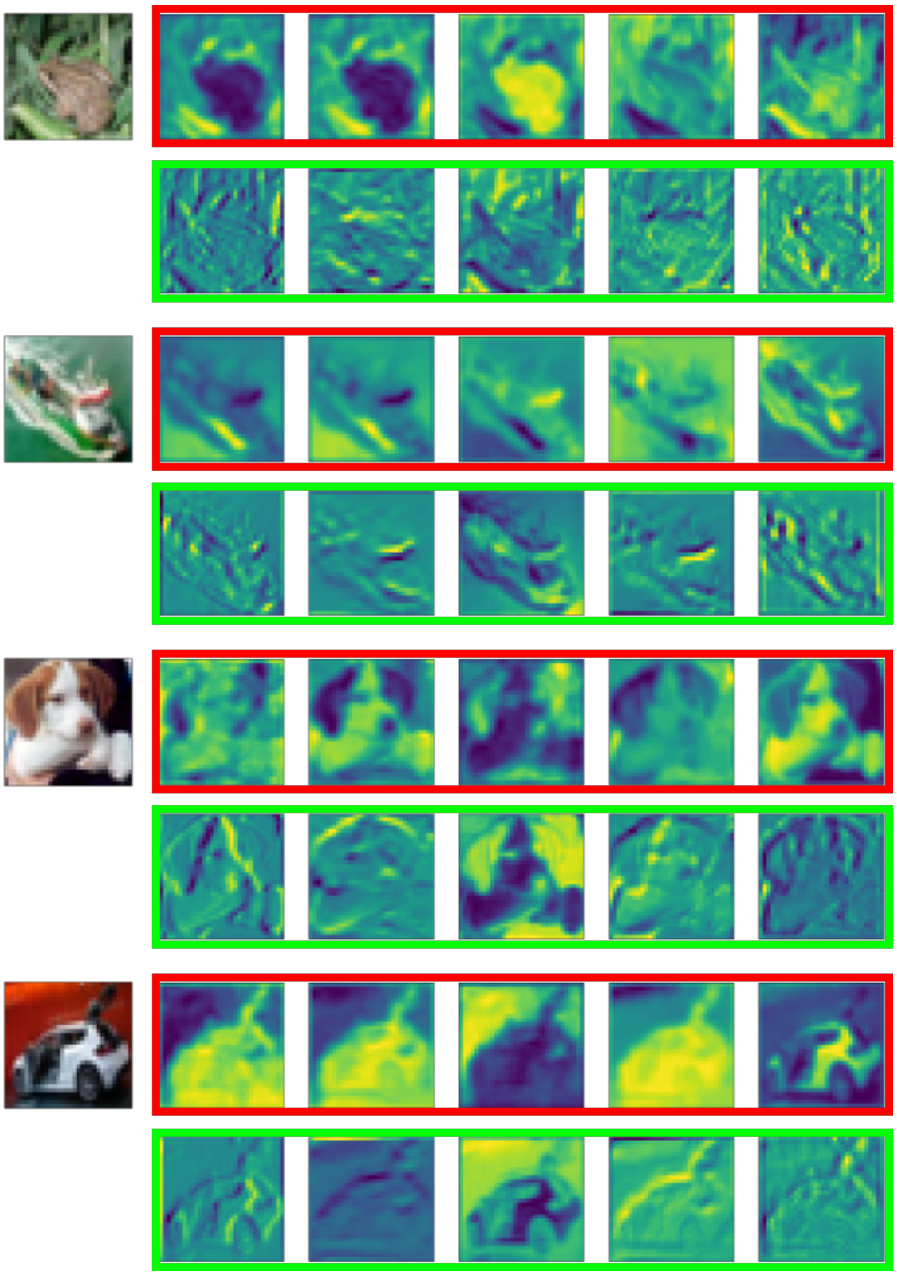

Figure 1. Illustrations of the top-5 feature maps that are most frequently de/re-allocated for each at the 1st dense block of a DenseNet-40 model. The results are shown across four different CIFAR-10 test images. The red and green boxes indicate the top-5 de-allocated and re-allocated channels, respectively.

References

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE Computer Society, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, volume 9908 of *Lecture Notes in Computer Science*, pp. 630–645. Springer, 2016b.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269. IEEE Computer Society, 2017.

Huang, G., Liu, S., Van der Maaten, L., and Weinberger, K. Q. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2752–2761, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.

Lin, M., Chen, Q., and Yan, S. Network in network. *CoRR*, abs/1312.4400, 2013.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Training very deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2377–2385, 2015.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2074–2082, 2016.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL http://arxiv.org/abs/1708.07747.

Xie, S., Girshick, R. B., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995. IEEE Computer Society, 2017.