Improved Regularization of Convolutional Neural Networks with Cutout

Terrance DeVries[1] and Graham W. Taylor[1,2]

[1]University of Guelph
[2]Canadian Institute for Advanced Research and Vector Institute

Abstract

Convolutional neural networks are capable of learning powerful representational spaces, which are necessary for tackling complex learning tasks. However, due to the model capacity required to capture such representations, they are often susceptible to overfitting and therefore require proper regularization in order to generalize well.

In this paper, we show that the simple regularization technique of randomly masking out square regions of input during training, which we call cutout, can be used to improve the robustness and overall performance of convolutional neural networks. Not only is this method extremely easy to implement, but we also demonstrate that it can be used in conjunction with existing forms of data augmentation and other regularizers to further improve model performance. We evaluate this method by applying it to current state-of-the-art architectures on the CIFAR-10, CIFAR-100, and SVHN datasets, yielding new state-of-the-art results of 2.56%, 15.20%, and 1.30% test error respectively. Code available at https://github.com/uoguelph-mlrg/Cutout.

1. Introduction

In recent years deep learning has contributed to considerable advances in the field of computer vision, resulting in state-of-the-art performance in many challenging vision tasks such as object recognition [8], semantic segmentation [11], image captioning [19], and human pose estimation [17]. Much of these improvements can be attributed to the use of convolutional neural networks (CNNs) [9], which are capable of learning complex hierarchical feature representations of images. As the complexity of the task to be solved increases, the resource utilization of such models increases as well: memory footprint, parameters, operations count, inference time and power consumption [2]. Modern networks commonly contain on the order of tens to hundreds of millions of learned parameters which provide the necessary representational power for such tasks, but with

Figure 1: Cutout applied to images from the CIFAR-10 dataset.

the increased representational power also comes increased probability of overfitting, leading to poor generalization.

In order to combat the potential for overfitting, several different regularization techniques can be applied, such as data augmentation or the judicious addition of noise to activations, parameters, or data. In the domain of computer vision, data augmentation is almost ubiquitous due to its ease of implementation and effectiveness. Simple image transforms such as mirroring or cropping can be applied to create new training data which can be used to improve model robustness and increase accuracy [9]. Large models can also be regularized by adding noise during the training process, whether it be added to the input, weights, or gradients. One of the most common uses of noise for improving model accuracy is dropout [6], which stochastically drops neuron activations during training and as a result discourages the co-adaptation of feature detectors.

In this work we consider applying noise in a similar fashion to dropout, but with two important distinctions. The first difference is that units are dropped out only at the input layer of a CNN, rather than in the intermediate feature layers. The second difference is that we drop out contiguous sections of inputs rather than individual pixels, as demon-

strated in Figure 1. In this fashion, dropped out regions are propagated through all subsequent feature maps, producing a final representation of the image which contains no trace of the removed input, other than what can be recovered by its context. This technique encourages the network to better utilize the full context of the image, rather than relying on the presence of a small set of specific visual features. This method, which we call cutout, can be interpreted as applying a spatial prior to dropout in input space, much in the same way that convolutional neural networks leverage information about spatial structure in order to improve performance over that of feed-forward networks.

In the remainder of this paper, we introduce cutout and demonstrate that masking out contiguous sections of the input to convolutional neural networks can improve model robustness and ultimately yield better model performance. We show that this simple method works in conjunction with other current state-of-the-art techniques such as residual networks and batch normalization, and can also be combined with most regularization techniques, including standard dropout and data augmentation. Additionally, cutout can be applied during data loading in parallel with the main training task, making it effectively computationally free. To evaluate this technique we conduct tests on several popular image recognition datasets, achieving state-of-the-art results on CIFAR-10, CIFAR-100, and SVHN. We also achieve competitive results on STL-10, demonstrating the usefulness of cutout for low data and higher resolution problems.

2. Related Work

Our work is most closely related to two common regularization techniques: data augmentation and dropout. Here we examine the use of both methods in the setting of training convolutional neural networks. We also discuss denoising auto-encoders and context encoders, which share some similarities with our work.

2.1. Data Augmentation for Images

Data augmentation has long been used in practice when training convolutional neural networks. When training LeNet5 [9] for optical character recognition, LeCun *et al.* apply various affine transforms, including horizontal and vertical translation, scaling, squeezing, and horizontal shearing to improve their model's accuracy and robustness.

In [1], Bengio *et al.* demonstrate that deep architectures benefit much more from data augmentation than shallow architectures. They apply a large variety of transformations to their handwritten character dataset, including local elastic deformation, motion blur, Gaussian smoothing, Gaussian noise, salt and pepper noise, pixel permutation, and adding fake scratches and other occlusions to the images, in addition to affine transformations.

To improve the performance of AlexNet [8] for the 2012 ImageNet Large Scale Visual Recognition Competition, Krizhevsky *et al.* apply image mirroring, cropping, as well as randomly adjusting colour and intensity values based on ranges determined using principal component analysis on the dataset.

Wu *et al.* take a more aggressive approach with image augmentation when training Deep Image [21] on the ImageNet dataset. In addition to flipping and cropping they apply a wide range of colour casting, vignetting, rotation, and lens distortion (pin cushion and barrel distortion), as well as horizontal and vertical stretching.

Lemley *et al.* tackle the issue of data augmentation with a learned end-to-end approach called Smart Augmentation [10] instead of relying on hard-coded transformations. In this method, a neural network is trained to intelligently combine existing samples in order to generate additional data that is useful for the training process.

Of these techniques ours is closest to the occlusions applied in [1], however their occlusions generally take the form of scratches, dots, or scribbles that overlay the target character, while we use zero-masking to completely obstruct an entire region.

2.2. Dropout in Convolutional Neural Networks

Another common regularization technique is dropout [6, 15], which was first introduced by Hinton *et al.*. Dropout is implemented by setting hidden unit activations to zero with some fixed probability during training. All activations are kept when evaluating the network, but the resulting output is scaled according to the dropout probability. This technique has the effect of approximately averaging over an exponential number of smaller sub-networks, and works well as a robust type of bagging, which discourages the co-adaptation of feature detectors within the network.

While dropout was found to be very effective at regularizing fully-connected layers, it appears to be less powerful when used with convolutional layers [16]. This reduction in potency can largely be attributed to two factors. The first is that convolutional layers already have much fewer parameters than fully-connected layers, and therefore require less regularization. The second factor is that neighbouring pixels in images share much of the same information. If any of them are dropped out then the information they contain will likely still be passed on from the neighbouring pixels that are still active. For these reasons, dropout in convolutional layers simply acts to increase robustness to noisy inputs, rather than having the same model averaging effect that is observed in fully-connected layers.

In an attempt to increase the effectiveness of dropout in convolutional layers, several variations on the original dropout formula have been proposed. Tompson *et al.* introduce SpatialDropout [16], which randomly discards en-

tire feature maps rather than individual pixels, effectively bypassing the issue of neighbouring pixels passing similar information.

Wu and Gu propose probabilistic weighted pooling [20], wherein activations in each pooling region are dropped with some probability. This approach is similar to applying dropout before each pooling layer, except that instead of scaling the output with respect to the dropout probability at test time, the output of each pooling function is selected to be the sum of the activations weighted by the dropout probability. The authors claim that this approach approximates averaging over an exponential number of sub-networks as dropout does.

In a more targeted approach, Park and Kwak introduce max-drop [13], which drops the maximal activation across feature maps or channels with some probability. While this regularization method performed better than conventional dropout on convolutional layers in some cases, they found that when used in CNNs that utilized batch normalization, both max-drop and SpatialDropout performed worse than standard dropout.

2.3. Denoising Auto-encoders & Context Encoders

Denosing auto-encoders [18] and context encoders [14] both rely on self-supervised learning to elicit useful feature representations of images. These models work by corrupting input images and requiring the network to reconstruct them using the remaining pixels as context to determine how best to fill in the blanks. Specifically, denoising auto-encoders that apply Bernoulli noise randomly erase individual pixels in the input image, while context encoders erase larger spatial regions. In order to properly fill in the missing information, the auto-encoders are forced to learn how to extract useful features from the images, rather than simply learning an identity function. As context encoders are required to fill in a larger region of the image they are required to have a better understanding of the global content of the image, and therefore they learn higher-level features compared to denoising auto-encoders [14]. These feature representations have been demonstrated to be useful for pre-training classification, detection, and semantic segmentation models.

While removing contiguous sections of the input has previously been used as an image corruption technique, like in context encoders, to our knowledge it has not previously been applied directly to the training of supervised models.

3. Cutout

Cutout is a simple regularization technique for convolutional neural networks that involves removing contiguous sections of input images, effectively augmenting the dataset with partially occluded versions of existing samples. This technique can be interpreted as an extension of dropout in input space, but with a spatial prior applied, much in the same way that CNNs apply a spatial prior to achieve improved performance over feed-forward networks on image data.

From the comparison between dropout and cutout, we can also draw parallels to denoising autoencoders and context encoders. While both models have the same goal, context encoders are more effective at representation learning, as they force the model to understand the content of the image in a global sense, rather than a local sense as denoising auto-encoders do. In the same way, cutout forces models to take more of the full image context into consideration, rather than focusing on a few key visual features, which may not always be present.

One of the major differences between cutout and other dropout variants is that units are dropped at the input stage of the network rather than in the intermediate layers. This approach has the effect that visual features, including objects that are removed from the input image, are correspondingly removed from all subsequent feature maps. Other dropout variants generally consider each feature map individually, and as a result, features that are randomly removed from one feature map may still be present in others. These inconsistencies produce a noisy representation of the input image, thereby forcing the network to become more robust to noisy inputs. In this sense, cutout is much closer to data augmentation than dropout, as it is not creating noise, but instead generating images that appear novel to the network.

3.1. Motivation

The main motivation for cutout comes from the problem of object occlusion, which is commonly encountered in many computer vision tasks, such as object recognition, tracking, or human pose estimation. By generating new images which simulate occluded examples, we not only better prepare the model for encounters with occlusions in the real world, but the model also learns to take more of the image context into consideration when making decisions.

We initially developed cutout as a targeted approach that specifically removed important visual features from the input of the image. This approach was similar to max-drop [13], in that we aimed to remove maximally activated features in order to encourage the network to consider less prominent features. To accomplish this goal, we extracted and stored the maximally activated feature map for each image in the dataset at each epoch. During the next epoch we then upsampled the saved feature maps back to the input resolution, and thresholded them at the mean feature map value to obtain a binary mask, which was finally overlaid on the original image before being passed through the CNN. Figure 2 demonstrates this early version of cutout.

While this targeted cutout method performed well, we found that randomly removing regions of a fixed size per-

Figure 2: An early version of cutout applied to images from the CIFAR-10 dataset. This targeted approach often occludes part-level features of the image, such as heads, legs, or wheels.

formed just as well as the targeted approach, without requiring any manipulation of the feature maps. Due to the inherent simplicity of this alternative approach, we focus on removing fixed-size regions for all of our experiments.

3.2. Implementation Details

To implement cutout, we simply apply a fixed-size zero-mask to a random location of each input image during each epoch of training, as shown in Figure 1. Unlike dropout and its variants, we do not apply any rescaling of weights at test time. For best performance, the dataset should be normalized about zero so that modified images will not have a large effect on the expected batch statistics.

In general, we found that the size of the cutout region is a more important hyperparameter than the shape, so for simplicity, we conduct all of our experiments using a square patch as the cutout region. When cutout is applied to an image, we randomly select a pixel coordinate within the image as a center point and then place the cutout mask around that location. This method allows for the possibility that not all parts of the cutout mask are contained within the image. Interestingly, we found that allowing portions of the patches to lay outside the borders of the image (rather than constraining the entire patch to be within the image) was critical to achieving good performance. Our explanation for this phenomenon is that it is important for the model to receive some examples where a large portion of the image is visible during training. An alternative approach that achieves similar performance is to randomly apply cutout constrained within the image region, but with 50% probability so that the network sometimes receives unmodified images.

The cutout operation can easily be applied on the CPU along with any other data augmentation steps during data loading. By implementing this operation on the CPU in parallel with the main GPU training task, we can hide the computation and obtain performance improvements for virtually free.

4. Experiments

To evaluate the performance of cutout, we apply it to a variety of natural image recognition datasets: CIFAR-10, CIFAR-100, SVHN, and STL-10.

4.1. CIFAR-10 and CIFAR-100

Both of the CIFAR datasets [7] consist of 60,000 colour images of size 32×32 pixels. CIFAR-10 has 10 distinct classes, such as cat, dog, car, and boat. CIFAR-100 contains 100 classes, but requires much more fine-grained recognition compared to CIFAR-10 as some classes are very visually similar. For example, it contains five different classes of trees: maple, oak, palm, pine, and willow. Each dataset is split into a training set with 50,000 images and a test set with 10,000 images.

Both datasets were normalized using per-channel mean and standard deviation. When required, we apply the standard data augmentation scheme for these datasets [5]. Images are first zero-padded with 4 pixels on each side to obtain a 40×40 pixel image, then a 32×32 crop is randomly extracted. Images are also randomly mirrored horizontally with 50% probability.

To evaluate cutout on the CIFAR datasets, we train models using two modern architectures: a deep residual network [5] with a depth of 18 (ResNet18), and a wide residual network [22] with a depth of 28, a widening factor of 10, and dropout with a drop probability of $p = 0.3$ in the convolutional layers (WRN-28-10). For both of these experiments, we use the same training procedure as specified in [22]. That is, we train for 200 epochs with batches of 128 images using SGD, Nesterov momentum of 0.9, and weight decay of 5e-4. The learning rate is initially set to 0.1, but is scheduled to decrease by a factor of 5x after each of the 60th, 120th, and 160th epochs. We also apply cutout to shake-shake regularization models [4] that currently achieve state-of-the-art performance on the CIFAR datasets, specifically a 26 2 $\times$ 96d "Shake-Shake-Image" ResNet for CIFAR-10 and a 29 2 $\times$ 4 $\times$ 64d "Shake-Even-Image" ResNeXt for CIFAR-100. For our tests, we use the original code and training settings provided by the author of [4], with the only change being the addition of cutout.

To find the best parameters for cutout we isolate 10% of the training set to use as a validation set and train on the remaining images. As our cutout shape is square, we perform a grid search over the side length parameter to find the optimal size. We find that model accuracy follows a parabolic trend, increasing proportionally to the cutout size until an optimal point, after which accuracy again decreases and eventually drops below that of the baseline model. This behaviour can be observed in Figure 3a and 3b, which depict the grid searches conducted on CIFAR-10 and CIFAR-100 respectively. Based on these validation results we select a cutout size of 16×16 pixels to use on CIFAR-10 and a
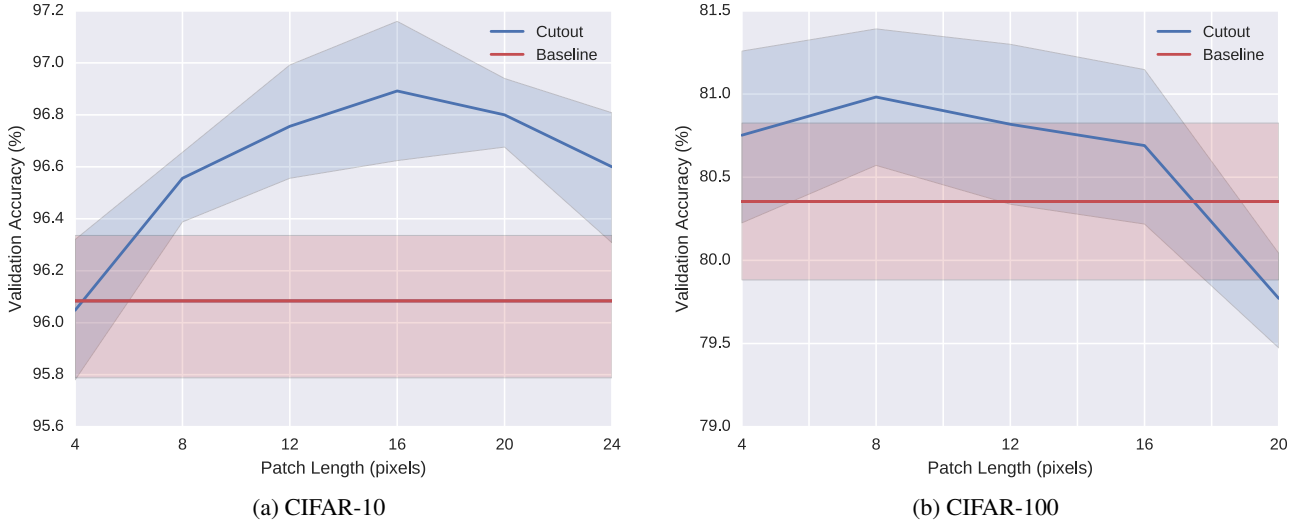
(a) CIFAR-10

(b) CIFAR-100

Figure 3: Cutout patch length with respect to validation accuracy with 95% confidence intervals (average of five runs). Tests run on CIFAR-10 and CIFAR-100 datasets using WRN-28-10 and standard data augmentation. Baseline indicates a model trained with no cutout.

Method	C10	C10+	C100	C100+	SVHN
ResNet18 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	-
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	-
WideResNet [22]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	$\mathbf{5.54 \pm 0.08}$	3.08 ± 0.16	$\mathbf{23.94 \pm 0.15}$	18.41 ± 0.27	$\mathbf{1.30 \pm 0.03}$
Shake-shake regularization [4]	-	2.86	-	15.85	-
Shake-shake regularization + cutout	-	$\mathbf{2.56 \pm 0.07}$	-	$\mathbf{15.20 \pm 0.21}$	-

Table 1: Test error rates (%) on CIFAR (C10, C100) and SVHN datasets. "+" indicates standard data augmentation (mirror + crop). Results averaged over five runs, with the exception of shake-shake regularization which only had three runs each. Baseline shake-shake regularization results taken from [4].

cutout size of 8×8 pixels for CIFAR-100 when training on the full datasets. Interestingly, it appears that as the number of classes increases, the optimal cutout size decreases. This makes sense, as when more fine-grained detection is required then the context of the image will be less useful for identifying the category. Instead, smaller and more nuanced details are important.

As shown in Table 1, the addition of cutout to the ResNet18 and WRN-28-10 models increased their accuracy on CIFAR-10 and CIFAR-100 by between 0.4 to 2.0 percentage points. We draw attention to the fact that cutout yields these performance improvements even when applied to complex models that already utilize batch normalization, dropout, and data augmentation. Adding cutout to the current state-of-the-art shake-shake regularization models improves performance by 0.3 and 0.6 percentage points on CIFAR-10 and CIFAR-100 respectively, yielding new state-of-the-art results of 2.56% and 15.20% test error.

4.2. SVHN

The Street View House Numbers (SVHN) dataset [12] contains a total of 630,420 colour images with a resolution of 32×32 pixels. Each image is centered about a number from one to ten, which needs to be identified. The official dataset split contains 73,257 training images and 26,032 test images, but there are also 531,131 additional training images available. Following standard procedure for this dataset [22], we use both available training sets when training our models, and do not apply any data augmentation. All images are normalized using per-channel mean and standard deviation.

To evaluate cutout on the SVHN dataset we apply it to a WideResNet with a depth of 16, a widening factor of 8, and dropout on the convolutional layers with a dropout rate of $p = 0.4$ (WRN-16-8). This particular configuration currently holds state-of-the-art performance on the SVHN dataset with a test error of 1.54% [22]. We repeat the same training procedure as specified in [22] by training for 160

epochs with batches of 128 images. The network is optimized using SGD with Nesterov momentum of 0.9 and weight decay of 5e-4. The learning rate is initially set to 0.01, but is reduced by a factor of 10x after the 80th and 120th epochs. The one change we do make to the original training procedure (for both baseline and cutout) is to normalize the data so that it is compatible with cutout (see § 3.2). The original implementation scales data to lie between 0 and 1.

To find the optimal size for the cutout region we conduct a grid search using 10% of the training set for validation and ultimately select a cutout size of 20×20 pixels. While this may seem like a large portion of the image to remove, it is important to remember that the cutout patches are not constrained to lie fully within the bounds of the image.

Using these settings we train the WRN-16-8 and observe an average reduction in test error of 0.3 percentage points, resulting in a new state-of-the-art performance of 1.30% test error, as shown in Table 1.

4.3. STL-10

The STL-10 dataset [3] consists of a total of 113,000 colour images with a resolution of 96×96 pixels. The training set only contains 5,000 images while the test set consists of 8,000 images. All training and test set images belong to one of ten classes, such as airplane, bird, or horse. The remainder of the dataset is composed of 100,000 unlabeled images belonging to the target ten classes, plus additional but visually similar classes. While the main purpose of the STL-10 dataset is to test semi-supervised learning algorithms, we use it to observe how cutout performs when applied to higher resolution images in a low data setting. For this reason, we discard the unlabeled portion of the dataset and only use the labeled training set.

The dataset was normalized by subtracting the per-channel mean and dividing by the per-channel standard deviation. Simple data augmentation was also applied in a similar fashion to the CIFAR datasets. Specifically, images were zero-padded with 12 pixels on each side and then a 96×96 crop was randomly extracted. Mirroring horizontally was also applied with 50% probability.

To evaluate the performance of cutout on the STL-10 dataset we use a WideResNet with a depth of 16, a widening factor of 8, and dropout with a drop rate of $p = 0.3$ in the convolutional layers. We train the model for 1000 epochs with batches of 128 images using SGD with Nesterov momentum of 0.9 and weight decay of 5e-4. The learning rate is initially set to 0.1 but is reduced by a factor of 5x after the 300th, 400th, 600th, and 800th epochs.

We perform a grid search over the cutout size parameter using 10% of the training images as a validation set and select a square size of 24×24 pixels for the no data-augmentation case and 32×32 pixels for training STL-10

with data augmentation. Training the model using these values yields a reduction in test error of 2.7 percentage points in the no data augmentation case, and 1.5 percentage points when also using data augmentation, as shown in Table 2.

Model	STL10	STL10+
WideResNet	23.48 ± 0.68	14.21 ± 0.29
WideResNet + cutout	$\mathbf{20.77 \pm 0.38}$	$\mathbf{12.74 \pm 0.23}$

Table 2: Test error rates on STL-10 dataset. "+" indicates standard data augmentation (mirror + crop). Results averaged over five runs on full training set.

4.4. Analysis of Cutout's Effect on Activations

In order to better understand the effect of cutout, we compare the average magnitude of feature activations in a ResNet18 when trained with and without cutout on CIFAR-10. The models were trained with data augmentation using the same settings as defined in Section 4.1, achieving scores of 3.89% and 4.94% test error respectively.

In Figure 4, we sort the activations within each layer by ascending magnitude, averaged over all samples in the test set. We observe that the shallow layers of the network experience a general increase in activation strength, while in deeper layers, we see more activations in the tail end of the distribution. The latter observation illustrates that cutout is indeed encouraging the network to take into account a wider variety of features when making predictions, rather than relying on the presence of a smaller number of features. Figure 5 demonstrates similar observations for individual samples, where the effects of cutout are more pronounced.

5. Conclusion

Cutout was originally conceived as a targeted method for removing visual features with high activations in later layers of a CNN. Our motivation was to encourage the network to focus more on complimentary and less prominent features, in order to generalize to situations like occlusion. However, we discovered that the conceptually and computationally simpler approach of randomly masking square sections of the image performed equivalently in the experiments we conducted. Importantly, this simple regularizer proved to be complementary to existing forms of data augmentation and regularization. Applied to modern architectures, such as wide residual networks or shake-shake regularization models, it achieves state-of-the-art performance on the CIFAR-10, CIFAR-100, and SVHN vision benchmarks. So why hasn't it been reported or analyzed to date? One reason could be the fact that using a combination of corrupted and clean images appears to be important for its success. Future work will return to our original investigation of visual feature removal informed by activations.

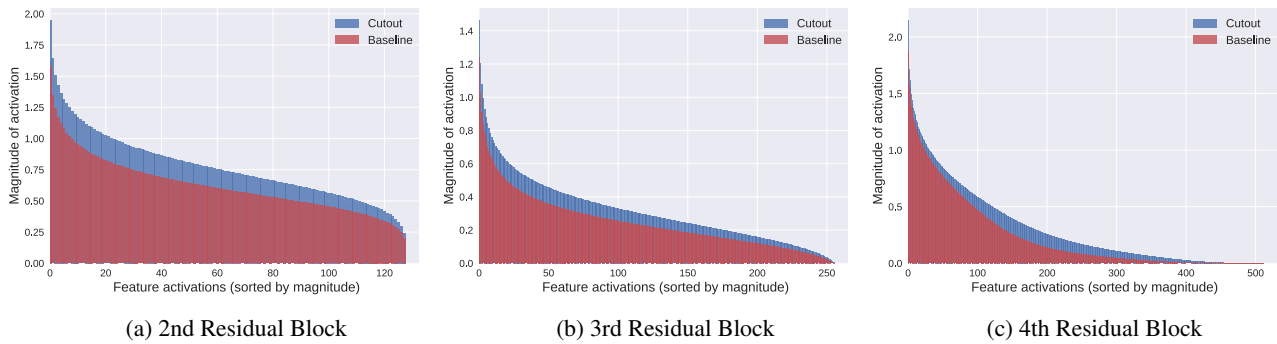|(a) 2nd Residual Block|(b) 3rd Residual Block|(c) 4th Residual Block|

Figure 4: Magnitude of feature activations, sorted by descending value, and averaged over all test samples. A standard ResNet18 is compared with a ResNet18 trained with cutout at three different depths.

Acknowledgements

References

[1] Y. Bengio, A. Bergeron, N. Boulanger-Lewandowski, T. Breuel, Y. Chherawala, et al. Deep learners benefit more from out-of-distribution examples. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 164–172, 2011.

[2] A. Canziani, A. Paszke, and E. Culurciello. An analysis of deep neural network models for practical applications. In *IEEE International Symposium on Circuits & Systems*, 2016.

[3] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[4] X. Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[7] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] J. Lemley, S. Bazrafkan, and P. Corcoran. Smart augmentation-learning an optimal data augmentation strategy. *IEEE Access*, 2017.

[11] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[12] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 5, 2011.

[13] S. Park and N. Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian Conference on Computer Vision*, pages 189–204. Springer, 2016.

[14] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544, 2016.

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

[16] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *CVPR*, pages 648–656, 2015.

[17] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, pages 1653–1660, 2014.

[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[19] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015.

[20] H. Wu and X. Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10, 2015.

[21] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun. Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, 7(8), 2015.

[22] S. Zagoruyko and N. Komodakis. Wide residual networks. *British Machine Vision Conference (BMVC)*, 2016.

A. Supplementary Materials

(a) 2nd Residual Block

(b) 3rd Residual Block

(c) 4th Residual Block

(d) 2nd Residual Block

(e) 3rd Residual Block

(f) 4th Residual Block

(g) 2nd Residual Block

(h) 3rd Residual Block

(i) 4th Residual Block

(j) 2nd Residual Block
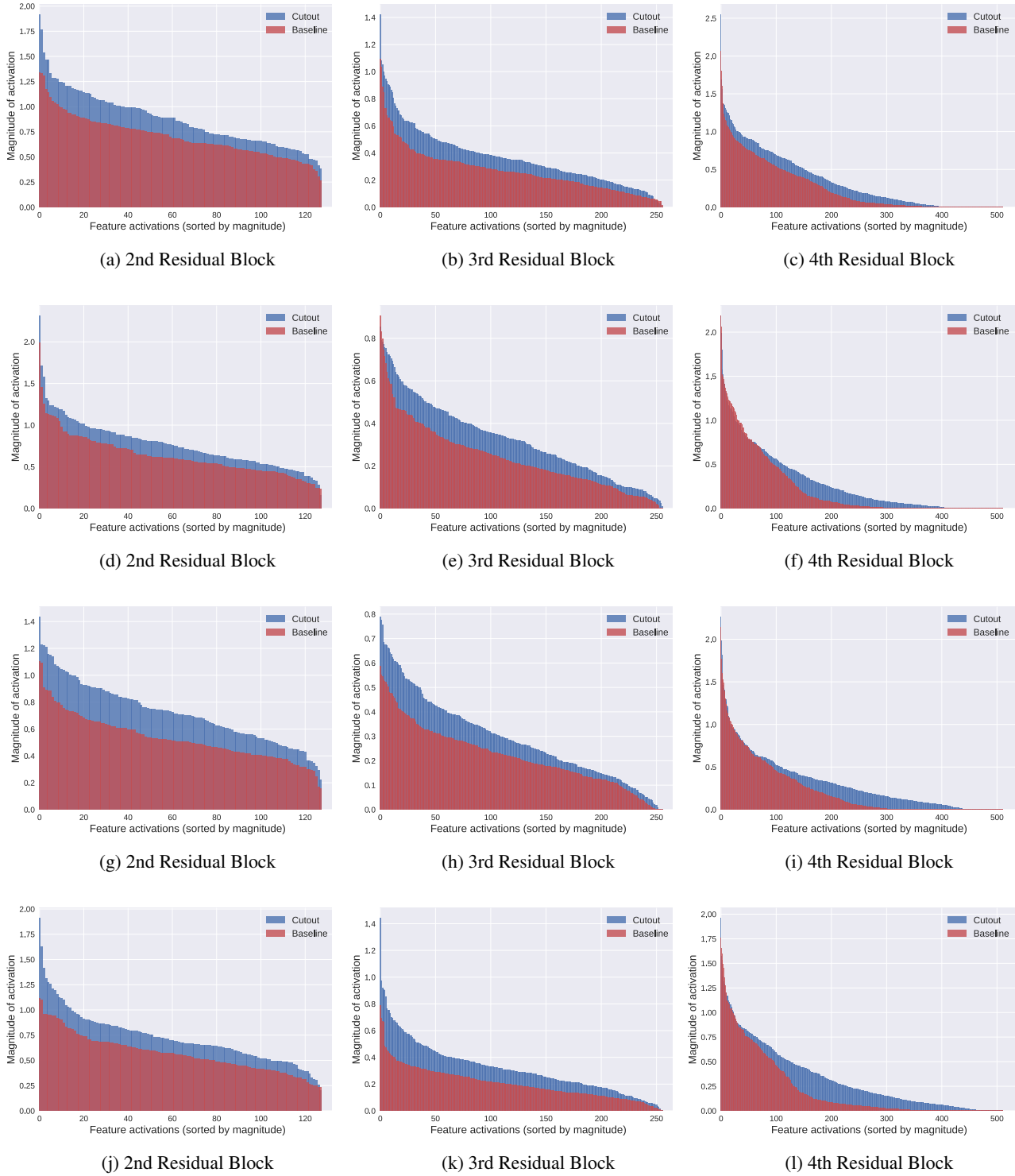
(k) 3rd Residual Block

(l) 4th Residual Block

Figure 5: Magnitude of feature activations, sorted by descending value. Each row represents a different test sample. A standard ResNet18 is compared with a ResNet18 trained with cutout at three different depths.