Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units

Wenling Shang[1,4] WENDY.SHANG@OCULUS.COM
Kihyuk Sohn[2] KSOHN@NEC-LABS.COM
Diogo Almeida[3] DIOGO@ENLITIC.COM
Honglak Lee[1] HONGLAK@EECS.UMICH.EDU
[1]University of Michigan, Ann Arbor; [2]NEC Laboratories America; [3]Enlitic; [4]Oculus VR

Abstract

Recently, convolutional neural networks (CNNs) have been used as a powerful tool to solve many problems of machine learning and computer vision. In this paper, we aim to provide insight on the property of convolutional neural networks, as well as a generic method to improve the performance of many CNN architectures. Specifically, we first examine existing CNN models and observe an intriguing property that the filters in the lower layers form pairs (i.e., filters with opposite phase). Inspired by our observation, we propose a novel, simple yet effective activation scheme called *concatenated ReLU* (CReLU) and theoretically analyze its reconstruction property in CNNs. We integrate CReLU into several state-of-the-art CNN architectures and demonstrate improvement in their recognition performance on CIFAR-10/100 and ImageNet datasets with fewer trainable parameters. Our results suggest that better understanding of the properties of CNNs can lead to significant performance improvement with a simple modification.

1. Introduction

In recent years, convolutional neural networks (CNNs) have achieved great success in many problems of machine learning and computer vision (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy et al., 2015; Girshick et al., 2014). In addition, a wide range of techniques have been developed to enhance the performance or ease the training of CNNs (Lin et al., 2013; Zeiler & Fergus, 2013; Maas et al., 2013; Ioffe & Szegedy, 2015). Despite the great empirical success, fundamental understanding of CNNs is still lagging behind. Towards addressing this issue, this paper aims to provide insight on the intrinsic property of convolutional neural networks.

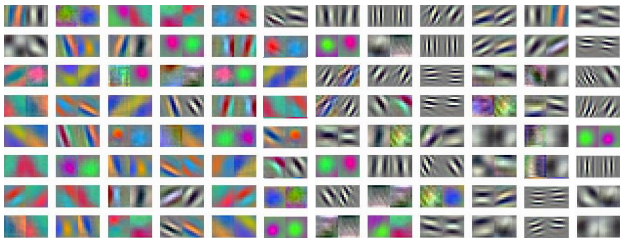
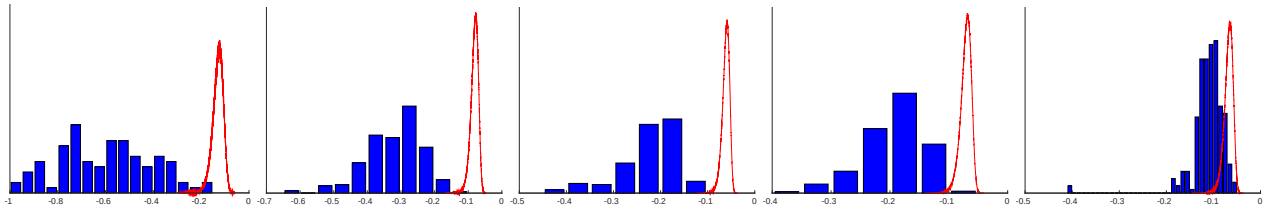


Figure 1. **Visualization of conv1 filters from AlexNet**. Each filter and its pairing filter (w_i and $\bar{w}_i$ next to each other) appear surprisingly opposite (in phase) to each other. See text for details.

To better comprehend the internal operations of CNNs, we investigate the well-known AlexNet (Krizhevsky et al., 2012) and thereafter discover that the network learns highly negatively-correlated pairs of filters for the first few convolution layers. Following our preliminary findings, we hypothesize that the lower convolution layers of AlexNet learn redundant filters to extract both positive and negative phase information of an input signal (Section 2.1). Based on the premise of our conjecture, we propose a novel, simple yet effective activation scheme called **Concatenated Rectified Linear Unit** (CReLU). The proposed activation scheme preserves both positive and negative phase information while enforcing non-saturated non-linearity. The unique nature of CReLU allows a mathematical characterization of convolution layers in terms of reconstruction property, which is an important indicator of how expressive and generalizable the corresponding CNN features are (Section 2.2).

In experiments, we evaluate the CNN models with CReLU and make a comparison to models with ReLU and Absolute Value Rectification Units (AVR) (Jarrett et al., 2009) on benchmark object recognition datasets, such as CIFAR-

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.



(a) conv1 (b) conv2 (c) conv3 (d) conv4 (e) conv5

Figure 2. **Histograms of μ^r(red) and μ^w(blue) for AlexNet**. Recall that for a set of unit length filters $\{\phi_i\}$, we define $\mu_i^{\phi} = \langle \phi_i, \bar{\phi}_i \rangle$ where $\bar{\phi}_i$ is the pairing filter of ϕ_i. For conv1 layer, the distribution of μ^w (from the AlexNet filters) is negatively centered, which significantly differs from that of μ^r (from random filters), whose center is very close to zero. The center gradually shifts towards zero when going deeper into the network.

10/100 and ImageNet (Section 3). We demonstrate that simply replacing ReLU with CReLU for the lower convolution layers of an existing state-of-the-art CNN architecture yields a substantial improvement in classification performance. In addition, CReLU allows to attain notable parameter reduction without sacrificing classification performance when applied appropriately.

We analyze our experimental results from several viewpoints, such as regularization (Section 4.1) and invariant representation learning (Section 4.2). Retrospectively, we provide empirical evaluations on the reconstruction property of CReLU models; we also confirm that by integrating CReLU, the original "pair-grouping" phenomenon vanishes as expected (Section 4.3). Overall, our results suggest that by better understanding the nature of CNNs, we are able to realize their higher potential with a simple modification of the architecture.

2. CRelu and Reconstruction Property

2.1. Conjecture on Convolution Layers

In our initial exploration of classic CNNs trained on natural images such as AlexNet (Krizhevsky et al., 2012), we noted a curious property of the first convolution layer filters: *these filters tend to form "pairs"*. More precisely, assuming unit length vector for each filter ϕ_i, we define a *pairing filter* of ϕ_i in the following way: $\bar{\phi}_i = \text{argmin}_{\phi_j} \langle \phi_i, \phi_j \rangle$. We also define their cosine similarity $\mu_i^{\phi} = \langle \phi_i, \bar{\phi}_i \rangle$.

In Figure 1, we show each normalized filter of the first convolution layer from AlexNet with its pairing filter. Interestingly, they appear surprisingly opposite to each other, i.e., for each filter, there does exist another filter that is almost on the opposite phase. Indeed, AlexNet employs the popular non-saturated activation function, *Rectified Linear Unit* (ReLU) (Nair & Hinton, 2010), which zeros out negative values and produces sparse activation. As a consequence, if both the positive phase and negative phase along a specific direction participate in representing the input space, the network then needs to learn two linearly dependent filters of both phases.

To systematically study the pairing phenomenon in higher layers, we graph the histograms of $\bar{\mu}_i^w$'s for conv1-conv5 filters from AlexNet in Figure 2. For comparison, we generate random Gaussian filters r_i's of unit norm[1] and plot the histograms of $\bar{\mu}_i^r$'s together. For conv1 layer, we observe that the distribution of $\bar{\mu}_i^w$ is negatively centered; by contrast, the mean of $\bar{\mu}_i^r$ is only slightly negative with a small standard deviation. Then the center of $\bar{\mu}_i^w$ shifts towards zero gradually when going deeper into the network. This implies that convolution filters of the lower layers tend to be paired up with one or a few others that represent their opposite phase, while the phenomenon gradually lessens as they go deeper.

Following these observations, we hypothesize that despite ReLU erasing negative linear responses, *the first few convolution layers of a deep CNN manage to capture both negative and positive phase information through learning pairs or groups of negatively correlated filters*. This conjecture implies that there exists a redundancy among the filters from the lower convolution layers.

In fact, for a very special class of deep architecture, the invariant scattering convolutional network (Bruna & Mallat, 2013), it is well-known that its set of convolution filters, which are wavelets, is overcomplete in order to be able to fully recover the original input signals. On the one hand, similar to ReLU, each individual activation within the scattering network preserves partial information of the input. On the other hand, different from ReLU but more similar to AVR, scattering network activation preserves the energy information, i.e., keeping the modulus of the responses but erasing the phase information; ReLU from a generic CNN, as a matter of fact, retains the phase information but eliminates the modulus information when the phase of a response is negative. In addition, while the wavelets for scattering networks are manually engineered, convolution filters from CNNs must be learned, which makes the rigorous theoretical analysis challenging.

[1] We sample each entry from standard normal distribution independently and normalize the vector to have unit l^2 norm.

Now suppose we can leverage the pairing prior and design a method to explicitly allow both positive and negative activation, then we will be able to alleviate the redundancy among convolution filters caused by ReLU non-linearity and make more efficient use of the trainable parameters. To this end, we propose a novel activation scheme, *Concatenated Rectified Linear Units*, or CReLU. It simply makes an identical copy of the linear responses after convolution, negate them, concatenate both parts of activation, and then apply ReLU altogether. More precisely, we denote ReLU as $[\cdot]_+ \triangleq \max(\cdot, 0)$, and define CReLU as follows:

Definition 2.1. CReLU *activation, denoted by* $\rho_c : \mathbb{R} \to \mathbb{R}^2$*, is defined as follows:* $\forall x \in \mathbb{R}, \rho_c(x) \triangleq ([x]_+, [-x]_+)$.

The rationale of our activation scheme is to allow a filter to be activated in both positive and negative direction while maintaining the same degree of non-saturated non-linearity.

An alternative way to allow negative activation is to employ the broader class of non-saturated activation functions including Leaky ReLU and its variants (Maas et al., 2013; Xu et al., 2015). Leaky ReLU assigns a small slope to the negative part instead of completely dropping it. These activation functions share similar motivation with CReLU in the sense that they both tackle the two potential problems caused by the hard zero thresholding: (1) the weights of a filter will not be adjusted if it is never activated, and (2) truncating all negative information can potentially hamper the learning. However, CReLU is based on an activation *scheme* rather than a *function*, which fundamentally differentiates itself from Leaky ReLU or other variants. In our version, we apply ReLU after separating the negative and positive part to compose CReLU, but it is not the only feasible non-linearity. For example, CReLU can be combined with other activation functions, such as Leaky ReLU, to add more diversity to the architecture.

Another natural analogy to draw is between CReLU and AVR, where the latter one only preserves the modulus information but discard the phase information, similar to the scattering network. AVR has not been widely used recently for the CNN models due to its suboptimal empirical performance. We confirm this common belief in the matter of large-scale image recognition task (Section 3) and conclude that modulus information alone does not suffice to produce state-of-the-art deep CNN features.

2.2. Reconstruction Property

A notable property of CReLU is its *information preservation nature*: CReLU conserves both negative and positive linear responses after convolution. A direct consequence of information preserving is the reconstruction power of the convolution layers equipped with CReLU.

Reconstruction property of a CNN implies that its features are representative of the input data. This aspect of CNNs has gained interest recently: Mahendran & Vedaldi (2015) invert CNN features back to the input under simple natural image priors; Zhao et al. (2015) stack autoencoders with reconstruction objective to build better classifiers. Bruna et al. (2013) theoretically investigate general conditions under which the max-pooling layer followed by ReLU is injective and measure stability of the inverting process by computing the Lipschitz lower bound. However, their bounds are non-trivial only when the number of filters significantly outnumbers the input dimension, which is not realistic.

In our case, it becomes more straightforward to analyze the reconstruction property since CReLU preserves all the information after convolution. The rest of this section mathematically characterizes the reconstruction property of a single convolution layer followed by CReLU with or without max-pooling layer.

We first analyze the reconstruction property of convolution followed by CReLU without max-pooling. This case is directly pertinent as deep networks replacing max-pooling with stride has become more prominent in recent studies (Springenberg et al., 2014). The following proposition states that the part of an input signal spanned by the shifts of the filters is well preserved.

Proposition 2.1. *Let* $x \in \mathbb{R}^D$ *be an input vector*[2] *and* W *be the* D*-by-*K *matrix whose columns vectors are composed of* $w_i \in \mathbb{R}^l, i = 1, \ldots, K$ *convolution filters. Furthermore, let* $x = x' + (x - x')$*, where* $x' \in \text{range}(W)$ *and* $x - x' \in \ker(W)$*. Then we can reconstruct* x' *with* $f_{\text{cnn}}(x)$*, where* $f_{\text{cnn}}(x) \triangleq \text{CReLU}\left(W^T x\right)$.

See Section A.1 in the supplementary materials for proof.

Next, we add max-pooling into the picture. To reach a non-trivial bound, we need additional constraints on the input space. Due to space limit, we carefully explain the constraints and the theoretical consequence in Section A.2 of the supplementary materials. We will revisit this subject after the experiment section (Section 4.3).

3. Benchmark Results

We evaluate the effectiveness of the CReLU activation scheme on three benchmark datasets: CIFAR-10, CIFAR-100 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). To directly assess the impact of CReLU, we employ existing CNN architectures with ReLU that have already shown a good recognition baseline and demonstrate improved performance on top by replacing ReLU into CReLU. Note that the models with CReLU activation don't need sig-

[2]For clarity, we assume the input signals are vectors (1D) rather than images (2D); however, similar analysis can be done for 2D case.

Table 1. **Test set recognition error rates on CIFAR-10/100.** We compare the performance of ReLU models (baseline) and CReLU models with different model capacities: "double" refers to the models that double the number of filters and "half" refers to the models that halve the number of filters. The error rates are provided in multiple ways, such as "Single", "Average" (with standard error), or "Vote", based on cross-validation methods. We also report the corresponding train error rates for the Single model. The number of model parameters are given in million. Please see the main text for more details about model evaluation.

Model	CIFAR-10				CIFAR-100				params.
	Single		Average	Vote	Single		Average	Vote	
	train	test			train	test			
Baseline	1.09	9.17	10.20±0.09	7.55	13.68	36.30	38.52±0.12	31.26	1.4M
+ (double)	0.47	8.65	9.87±0.09	7.28	6.03	34.77	36.73±0.15	28.34	5.6M
AVR	4.10	**8.32**	10.26±0.10	7.76	19.35	35.00	37.24±0.20	29.77	1.4M
CReLU	4.23	8.43	**9.39**±0.11	**7.09**	14.25	**31.48**	**33.76**±0.12	**27.60**	2.8M
+ (half)	4.73	**8.37**	**9.44**±0.09	**7.09**	21.01	33.68	36.20±0.18	29.93	0.7M

nificant hyperparameter tuning from the baseline ReLU model, and in most of our experiments, we only tune dropout rate while other hyperparameters (e.g., learning rate, mini-batch size) remain the same. We also replace ReLU with AVR for comparison with CReLU. The details of network architecture are in Section F of the supplementary materials.

3.1. CIFAR-10 and CIFAR-100

The CIFAR-10 and 100 datasets (Krizhevsky, 2009) each consist of $50,000$ training and $10,000$ testing examples of 32×32 images evenly drawn from 10 and 100 classes, respectively. We subtract the mean and divide by the standard deviation for preprocessing and use random horizontal flip for data augmentation.

We use the ConvPool-CNN-C model (Springenberg et al., 2014) as our baseline model, which is composed of convolution and pooling followed by ReLU without fully-connected layers. This baseline model serves our purpose well since it has clearly outlined network architecture only with convolution, pooling, and ReLU. It has also shown competitive recognition performance using a fairly small number of model parameters.

First, we integrate CReLU into the baseline model by simply replacing ReLU while keeping the number of convolution filters the same. This doubles the number of output channels at each convolution layer and the total number of model parameters is doubled. To see whether the performance gain comes from the increased model capacity, we conduct additional experiments with the baseline model while doubling the number of filters and the CReLU model while halving the number of filters. We also evaluate the performance of the AVR model while keeping the number of convolution filters the same as the baseline model.

Since the datasets don't provide pre-defined validation set, we conduct two different cross-validation schemes:

1. "Single": we hold out a subset of training set for initial training and retrain the network from scratch using the whole training set until we reach at the same loss on a hold out set (Goodfellow et al., 2013). For this case, we also report the corresponding train error rates.
2. 10-folds: we divide training set into 10 folds and do validation on each of 10 folds while training the networks on the rest of 9 folds. The mean error rate of single network ("Average") and the error rate with model averaging of 10 networks ("Vote") are reported.

The recognition results are summarized in Table 1. On CIFAR-10, we observe significant improvement with the CReLU activation over ReLU. Especially, CReLU models consistently improve over ReLU models with the same number of neurons (or activations) while reducing the number of model parameters by half (e.g., CReLU + half model and the baseline model have the same number of neurons while the number of model parameters are 0.7M and 1.4M, respectively). On CIFAR-100, the models with larger capacity generally improve the performance for both activation schemes. Nevertheless, we still find a clear benefit of using CReLU activation that shows significant performance gain when it is compared to the model with the same number of neurons, i.e., half the number of model parameters. One possible explanation for the benefit of using CReLU is its regularization effect, as can be confirmed in Table 1 that the CReLU models showed significantly lower gap between train and test set error rates than those of the baseline ReLU models.

To our slight surprise, AVR outperforms the baseline ReLU model on CIFAR-100 with respect to all evaluation metrics and on CIFAR-10 with respect to single-model evaluation. It also reaches promising single-model recognition accuracy compared to CReLU on CIFAR-10; however, when averaging or voting across 10-folds validation models, AVR becomes clearly inferior to CReLU.

Experiments on Deeper Networks. We conduct experiments with very deep CNN that has a similar network architecture to the VGG network (Simonyan & Zisserman, 2014). Specifically, we follow the model architecture and

Table 2. **Test set recognition error rates on CIFAR-10/100 using deeper networks.** We gradually apply CReLU to replace ReLU after conv1, conv3, and conv5 layers of the baseline VGG network while halving the number of convolution filters.

CIFAR-10			
Model	Single	Average	Vote
VGG	6.35	6.90±0.03	5.43
(conv1)	6.18	**6.45**±0.05	5.22
(conv1,3)	**5.94**	**6.45**±0.02	**5.09**
(conv1,3,5)	6.06	**6.45**±0.07	5.16
CIFAR-100			
Model	Single	Average	Vote
VGG	28.99	30.27±0.09	26.85
(conv1)	27.29	28.43±0.11	24.67
(conv1,3)	26.52	27.79±0.08	23.93
(conv1,3,5)	**26.16**	**27.67**±0.07	**23.66**

training procedure in Zagoruyko (2015). Besides the convolution and pooling layers, this network contains batch normalization (Ioffe & Szegedy, 2015) and fully connected layers. Due to the sophistication of the network composition which may introduce complicated interaction with CReLU, we only integrate CReLU into the first few layers. Similarly, we subtract the mean and divide by the standard deviation for preprocessing and use horizontal flip and random shifts for data augmentation.

In this experiment[3], we gradually replace ReLU after the first, third, and the fifth convolution layers[4] with CReLU while halving the number of filters, resulting in a reduced number of model parameters. We report the test set error rates using the same cross-validation schemes as in the previous experiments. As shown in Table 2, there is substantial performance gain in both datasets by replacing ReLU with CReLU. Overall, the proposed CReLU activation improves the performance of the state-of-the-art VGG network significantly, achieving highly competitive error rates to other state-of-the-art methods, as summarized in Table 3.

3.2. ImageNet

To assess the impact of CReLU on large scale dataset, we perform experiments on ImageNet dataset (Deng et al., 2009)[5], which contains about 1.3M images for training and $50,000$ for validation from $1,000$ object categories. For preprocessing, we subtract the mean and divide by the standard deviation for each input channel, and follow the data augmentation as described in (Krizhevsky et al., 2012).

We take the All-CNN-B model (Springenberg et al., 2014) as our baseline model. The network architecture of All-CNN-B is similar to that of AlexNet (Krizhevsky et al., 2012), where the max-pooling layer is replaced by convolution with the same kernel size and stride, the fully connected layer is replaced by 1×1 convolution layers followed by average pooling, and the local response normalization layers are discarded. In sum, the layers other than convolution layers are replaced or discarded and finally the network consists of convolution layers only. We choose this model since it reduces the potential complication introduced by CReLU interacting with other types of layers, such as batch normalization or fully connected layers.

Table 3. **Comparison to other methods on CIFAR-10/100.**

Model	CIFAR-10	CIFAR-100
(Rippel et al., 2015)	8.60	31.60
(Snoek et al., 2015)	6.37	27.40
(Liang & Hu, 2015)	7.09	31.75
(Lee et al., 2016)	6.05	32.37
(Srivastava et al., 2015)	7.60	32.24
VGG	5.43	26.85
VGG + CReLU	**5.09**	**23.66**

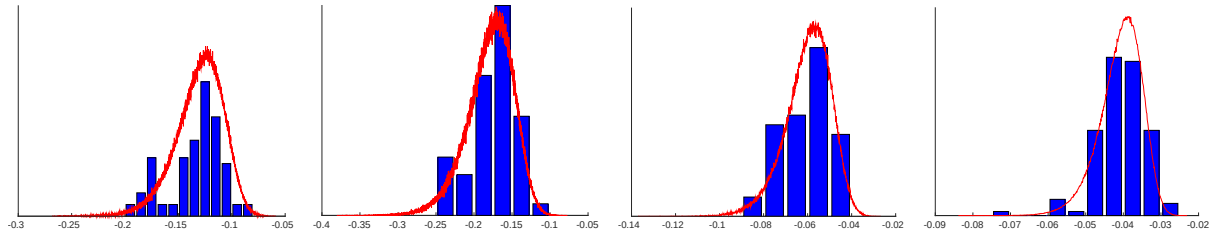
We gradually integrate more convolution layers with CReLU (e.g., conv1–4, conv1–7, conv1–9), while keeping the same number of filters. These models contain more parameters than the baseline model. We also evaluate two models where one replaces all ReLU layers into CReLU and the other conv1,conv4 and conv7 only, where both models reduce the number of convolution layers before CReLU by half. Hence, these models contain fewer parameters than the baseline model. For comparison, AVR models are also constructed by gradually replacing ReLU in the same manner as the CReLU experiments (conv1–4, conv1–7, conv1–9). The network architectures and the training details are in Section F and Section E of the supplementary materials.

The results are provided in Table 4. We report the top-1 and top-5 error rates with center crop only and by averaging scores over 10 patches from the center crop and four corners and with horizontal flip (Krizhevsky et al., 2012). Interestingly, integrating CReLU to conv1-4 achieves the best results, whereas going deeper with higher model capacity does not further benefit the classification performance. In fact, this parallels with our initial observation on AlexNet (Figure 2 in Section 2.1)—there exists less "pairing" in the deeper convolution layers and thus there is not much gain by decomposing the phase in the deeper layers. AVR networks exhibit the same trend but do not noticeably improve upon the baseline performance, which implies that AVR is not the most suitable candidate for large-scale deep representation learning. Another interesting observation, which we will discuss further in Section 4.2, is that the model integrating CReLU into conv1, conv4 and

[3]We attempted to replace ReLU with AVR on various layers but we observed significant performance drop with AVR nonlinearity when used for deeper networks.

[4]Integrating CReLU into the second or fourth layer before max-pooling layers did not improve the performance.

[5]We used a version of ImageNet dataset for ILSVRC 2012.



(a) conv1 (b) conv2 (c) conv3 (d) conv4

Figure 3. **Histograms of μ^r(red) and μ^w(blue) for CReLU model on ImageNet.** The two distributions align with each other for all conv1-conv4 layers–as we expected, the pairing phenomenon is not present any more after applying the CReLU activation scheme.

Table 4. **Validation error rates on ImageNet.** We compare the performance of baseline model with the proposed CReLU models at different levels of activation scheme replacement. Error rates with † are obtained by averaging scores from 10 patches.

Model	top-1	top-5	top-1†	top-5†
Baseline	41.81	19.74	38.03	17.17
AVR (conv1–4)	41.12	19.25	37.32	16.49
AVR (conv1–7)	42.36	20.05	38.21	17.42
AVR (conv1–9)	43.33	21.05	39.70	18.39
CReLU (conv1,4,7)	40.45	18.58	**35.70**	**15.32**
CReLU (conv1–4)	**39.82**	**18.28**	36.20	15.72
CReLU (conv1–7)	39.97	18.33	36.53	16.01
CReLU (conv1–9)	40.15	18.58	36.50	16.14
CReLU (all)	40.93	19.39	37.28	16.72

Table 5. **Comparison to other methods on ImageNet.** We compare with AlexNet and other variants, such as FastFood-32-AD (FriedNet) (Yang et al., 2015) and pruned AlexNet (PrunedNet) (Han et al., 2015), which are modifications of AlexNet aiming at reducing the number of parameters, as well as All-CNN-B, the baseline model (Springenberg et al., 2014). Error rates with † are obtained by averaging scores from 10 patches.

Model	top-1	top-5	top-1†	top-5†	params.
AlexNet	42.6	19.6	40.7	18.2	61M
FriedNet	41.93	–	–	–	32.8M
PrunedNet	42.77	19.67	–	–	6.7M
AllConvB	41.81	19.74	38.03	17.17	9.4M
CReLU (all)	40.93	19.39	37.28	16.72	**4.7M**
(conv1,4,7)	40.45	18.58	**35.70**	**15.32**	8.6 M
(conv1–4)	**39.82**	**18.28**	36.20	15.72	10.1M

conv7 layers also achieve highly competitive recognition results with even fewer parameters than the baseline model. In sum, we believe that such a significant improvement over the baseline model by simply modifying the activation scheme is a pleasantly surprising result.[6]

We also compare our best models with AlexNet and other variants in Table 5. Even though reducing the number of parameters is not our primary goal, it is worth noting that our model with only 4.6M parameters (CReLU + all) outperforms FastFood-32-AD (FriedNet) (Yang et al., 2015) and Pruned AlexNet (PrunedNet) (Han et al., 2015), whose designs directly aim at parameter reduction. Therefore, besides the performance boost, another significance of CReLU activation scheme is in designing more parameter-efficient deep neural networks.

4. Discussion

In this section, we discuss qualitative properties of CReLU activation scheme in several viewpoints, such as regularization of the network and learning invariant representation.

4.1. A View from Regularization

In general, a model with more trainable parameters is more prone to overfitting. However, somewhat counter-intuitively, for the all-conv CIFAR experiments, the models with CReLU display much less overfitting issue compared to the baseline models with ReLU, even though it has twice as many parameters (Table 1). We contemplate that keeping both positive and negative phase information makes the training more challenging, and such effect has been leveraged to better regularize deep networks, especially when working on small datasets.

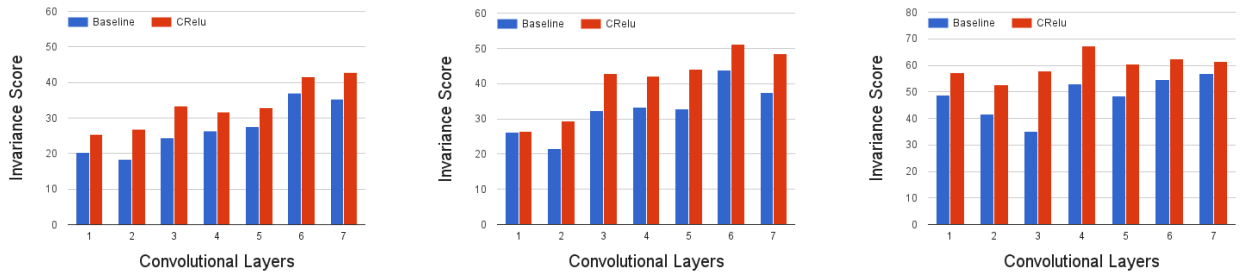
Besides the empirical evidence, we can also describe the regularization effect by deriving a Rademacher complexity bound for the CReLU layer followed by linear transformation as follows:

Theorem 4.1. *Let $\mathcal{G}$ be the class of real functions $\mathbb{R}^{d_{in}} \to \mathbb{R}$ with input dimension $\mathcal{F}$, that is, $\mathcal{G} = [\mathcal{F}]_{j=1}^{d_{in}}$. Let $\mathcal{H}$ be a linear transformation function from $\mathbb{R}^{2d_{in}}$ to $\mathbb{R}$, parametrized by W, where $\|W\|_2 \leq B$. Then, we have*

$$\hat{R}_L(\mathcal{H} \circ \rho_c \circ \mathcal{G}) \leq \sqrt{d_{in}} B \hat{R}_L(\mathcal{F}).$$

The proof is in Section B of the supplementary materials. Theorem 4.1 says that the complexity bound of CReLU + linear transformation is the same as that of ReLU + linear transformation, which is proved by Wan et al. (2013). In other words, although the number of model parameters are doubled by CReLU, the model complexity does not necessarily increase.

[6] We note that Springenberg et al. (2014) reported slightly better result (41.2% top-1 error rate with center crop only) than our replication result, but still the improvement is significant.



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet

Figure 4. **Invariance Scores for** ReLU **Models vs** CReLU **Models**. The invariance scores for CReLU models are consistently higher than ReLU models. The invariance scores jump after max-pooling layers. Moreover, even though the invariance scores tend to increase along with the depth of the networks, the progression is not monotonic.

Table 6. **Correlation Comparison.** The averaged correlation between the normalized positive-negative-pair (pair) outgoing weights and the normalized unmatched-pair (non-pair) outgoing weights are both well below 1 for all layers, indicating that the pair outgoing weights are capable of imposing diverse non-linear manipulation separately on the positive and negative components.

ImageNet Conv1–7 CReLU Model		
layer	pair	non-pair
conv1	0.372 $\pm$0.372	0.165 $\pm$0.154
conv2	0.180 $\pm$0.149	0.157 $\pm$0.137
conv3	0.462 $\pm$0.249	0.120 $\pm$0.120
conv4	0.175 $\pm$0.146	0.119 $\pm$0.100
conv5	0.206 $\pm$0.136	0.105 $\pm$0.093
conv6	0.256 $\pm$0.124	0.086 $\pm$0.080
conv7	0.131 $\pm$0.122	0.080 $\pm$0.070

4.2. Towards Learning Invariant Features

We measure the invariance scores using the evaluation metrics from (Goodfellow et al., 2009) and draw another comparison between the CReLU models and the ReLU models. For a fair evaluation, we compare all 7 conv layers from all-conv ReLU model with those from all-conv CReLU model trained on CIFAR-10/100. In the case of ImageNet experiments, we choose the model where CReLU replaces ReLU for the first 7 conv layers and compare the invariance scores with the first 7 conv layers from the baseline ReLU model. Section D in the supplementary materials details how the invariance scores are measured.

Figure 4 plots the invariance scores for networks trained on CIFAR-10, CIFAR-100, and ImageNet respectively. The invariance scores of CReLU models are consistently higher than those of ReLU models. For CIFAR-10 and CIFAR-100, there is a big increase between conv2 and conv3 then again between conv4 and conv6, which are due to max-pooling layer extracting shift invariance features. We also observe that although as a general trend, the invariance scores increase while going deeper into the networks– consistent with the observations from (Goodfellow et al., 2009), the progression is not monotonic. This interesting observation suggests the potentially diverse functionality of different layers in the CNN, which would be worthwhile for future investigation.

In particular, the scores of ImageNet ReLU model attain local maximum at conv1, conv4 and conv7 layers. It inspires us to design the architecture where CReLU are placed after conv1, 4, and 7 layers to encourage invariance representations while halving the number of filters to limit model capacity. Interestingly, this architecture achieves the best top1 and top5 recognition results when averaging scores from 10 patches.

4.3. Revisiting the Reconstruction Property

In Section 2.1, we observe that lower layer convolution filters from ReLU models form negatively-correlated pairs. Does the pairing phenomenon still exist for CReLU models? We take our best CReLU model trained on ImageNet (where the first 4 conv layers are integrated with CReLU) and repeat the histogram experiments to generate Figure 3. In clear contrast to Figure 2, the distributions of $\bar{\mu}_i^w$ from CReLU model well align with the distributions of $\bar{\mu}_i^r$ from random Gaussian filters. In other words, each lower layer convolution filter now uniquely spans its own direction without a negatively correlated pairing filter, while CReLU implicitly plays the role of "pair-grouping".

The empirical gap between CReLU and AVR justifies that both modulus and phase information are essential in learning deep CNN features. In addition, to ensure that the outgoing weights for the positive and negative phase are not merely negations of each other, we measure their correlations for the conv1-7 CReLU model trained on ImageNet. Table 6 compares the averaged correlation between the (normalized) positive-negative-pair (pair) outgoing weights and the (normalized) unmatched-pair (non-pair) outgoing weights. The pair correlations are

(a) Original image (b) conv1 (c) conv2 (d) conv3 (e) conv4

Figure 5. CReLU **Model Reconstructions.** We use a simple linear reconstruction algorithm (see Algorithm 1 in the supplementary materials) to reconstruct the original image from conv1-conv4 features (left to right). The image is best viewed in color/screen.

marginally higher than the non-pair ones but both are on average far below 1 for all layers. This suggests that, in contrast to AVR, the CReLU network does not simply focus on the modulus information but imposes different manipulation over the opposite phases.

In Section 2.2, we mathematically characterize the reconstruction property of convolution layers with CReLU. Proposition 2.1 claims that the part of an input spanned by the shifts of the filters can be fully recovered. ImageNet contains a large number of training images from a wide variety of categories; the convolution filters learned from ImageNet are thus expected to be diverse enough to describe the domain of natural images. Hence, to qualitatively verify the result from Proposition 2.1, we can directly invert features from our best CReLU model trained on ImageNet via the simple reconstruction algorithm described in the proof of Proposition 2.1 (Algorithm 1 in the supplementary materials). Figure 5 shows an image from the validation set along with its reconstructions using conv1-conv4 features (see Section G in the supplementary materials for more reconstruction examples). Unlike other reconstruction methods (Dosovitskiy & Brox, 2015; Mahendran & Vedaldi, 2015), our algorithm does not involve any additional learning. Nevertheless, it still produces reasonable reconstructions, which supports our theoretical claim in Proposition 2.1.

For the convolution layers involving max-pooling operation, it is less straightforward to perform direct reconstruction. Yet we evaluate the conv+CReLU+max-pooling reconstruction power via measuring properties of the convolution filters and the details are elaborated in Section C of the supplementary materials.

5. Conclusion

We propose a new activation scheme, CReLU, which conserves both positive and negative linear responses after convolution so that each filter can efficiently represent its unique direction. Our work demonstrates that CReLU improves deep networks with classification objective. Since CReLU preserves the available information from input while maintaining the non-saturated nonlinearity, it can potentially benefit more complicated machine learning tasks such as structured output prediction and image generation. Another direction for future research involves engaging CReLU to the abundant set of existing deep neural network techniques and frameworks. We hope to investigate along these directions in the near future.

ACKNOWLEDGMENTS

We are grateful to Erik Brinkman, Harry Altman and Mark Rudelson for their helpful comments and support. We acknowledge Yuting Zhang and Anna Gilbert for discussions during the preliminary stage of this work. This work was supported in part by ONR N00014-13-1-0762 and NSF CAREER IIS-1453651. We thank Technicolor Research for providing resources and NVIDIA for the donation of GPUs.

References

Bruna, J. and Mallat, S. Invariant scattering convolution networks. *PAMI*, 2013.

Bruna, J., Szlam, A., and LeCun, Y. Signal recovery from pooling representations. In *ICML*, 2013.

Christensen, O. *An introduction to frames and Riesz bases*. Birkhuser Basel, 2003.

Deng, J., Dong, W., Socher, R., Li, L.-j., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Dosovitskiy, A. and Brox, T. Inverting convolutional networks with convolutional networks. In *CVPR*, 2015.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., and Ng, A. Measuring invariances in deep networks. In *NIPS*, 2009.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. In *ICML*, 2013.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. What is the best multi-stage architecture for object recognition? In *CVPR*, 2009.

Krizhevsky, A. Learning multiple layers of features from tiny images, 2009.

Krizhevsky, A., Sutskever, l., and Hinton, G. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Lee, C.-y., Gallagher, P. W., and Tu, Z. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016.

Liang, M. and Hu, X. Recurrent convolutional neural network for object recognition. In *CVPR*, 2015.

Lin, M., Chen, Q., and Yan, S. Network in network. In *ICLR*, 2013.

Maas, A., Hannun, A. Y., and Ng, A. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *CVPR*, 2015.

Nair, V. and Hinton, G. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Rippel, O., Snoek, J., and Adams, R. Spectral representations for convolutional neural networks. In *NIPS*, 2015.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M. M. A., and Adams, R. Scalable bayesian optimization using deep neural networks. In *ICML*, 2015.

Springenberg, J., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. In *ICLR Workshop*, 2014.

Srivastava, R., Greff, K., and Schmidhuber, J. Training very deep networks. In *NIPS*, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *ICML*, 2013.

Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. In *ICML Workshop*, 2015.

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., and Wang, Z. Deep fried convnets. In *ICCV*, 2015.

Zagoruyko, S. Torch blog. http://torch.ch/blog/2015/07/30/cifar.html, 2015.

Zeiler, M. D. and Fergus, R. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

Zhao, J., Mathieu, M., Goroshin, R., and Lecun, Y. Stacked what-where auto-encoders. In *ICLR*, 2015.

Algorithm 1 Reconstruction over a single convolution region without max-pooling

1: $f_{\text{cnn}}(x) \leftarrow$ conv features.
2: $W \leftarrow$ weight matrix.
3: Obtain the linear responses after convolution by reverting CReLU: $z = \rho_c^{-1}(f_{\text{cnn}}(x))$.
4: Compute the Moore Penrose pseudoinverse of W^T, $(W^T)^+$.
5: Obtain the final reconstruction: $x' = (W^T)^+ z$.

Algorithm 2 Reconstruction over a single max-pooling region

1: $f_{\text{cnn}}(x) \leftarrow$ conv features after max-pooling.
2: $\widehat{W}_x \leftarrow$ weight matrix consisting of shifted conv filters that are activated by x.
3: Obtain the linear responses after convolution by reverting CReLU: $z = \rho_c^{-1}(f_{\text{cnn}}(x))$.
4: Compute the Moore Penrose pseudoinverse of $\widehat{W}_x^T$, $(\widehat{W}_x^T)^+$.
5: Obtain the final reconstruction: $x' = (\widehat{W}_x^T)^+ z$.

Appendix

A. Reconstruction Property Proofs

A.1. Non-Max-Pooling Case

Proposition A.1. *Let $x \in \mathbb{R}^D$ be an input vector and W be the D-by-K matrix whose columns vectors are composed of $w_i \in \mathbb{R}^l, i = 1, \ldots, K$ convolution filters. Furthermore, let $x = x' + (x - x')$, where $x' \in \text{range}(W)$ and $x - x' \in \ker(W)$. Then we can reconstruct x' with $f_{\text{cnn}}(x)$, where $f_{\text{cnn}}(x) \triangleq \text{CReLU}\left(W^T x\right)$.*

Proof. We show x' can be reconstructed from $f_{\text{cnn}}(x)$ by providing a simple reconstruction algorithm described by Algorithm 1. First, apply the inverse function of CReLU on $f_{\text{cnn}}(x)$: $z = \rho_c^{-1}(f_{\text{cnn}}(x))$. Then, compute the Moore Penrose pseudoinverse of W^T, denote by $(W^T)^+$. By definition $Q = (W^T)^+ W^T$ is the orthogonal projector onto $\text{range}(W)$, therefore we can obtain $x' = (W^T)^+ z$. □

A.2. Max-Pooling Case

Problem Setup. Again, let $x \in \mathbb{R}^D$ be an input vector and $w_i \in \mathbb{R}^\ell$, $i = 1, \ldots, K$ be convolution filters. We denote $w_i^j \in \mathbb{R}^D$ the j^{th} coordinate shift of the convolution filter w_i with a fixed stride length of s, i.e., $w_i^j[(j-1)s + k] = w_i[k]$ for $k = 1, \ldots, \ell$, and 0's for the rest of entries in the vector. Here, we assume $D - \ell$ is divisible by s and thus there are $n = \frac{D-\ell}{s} + 1$ shifts for each w_i. We define W to be the $D \times nK$ matrix whose columns are the shifts w_i^j, $j = 1, \ldots, n$, for w_i; the columns of W are divided into K blocks, with each block consisting of n shifts of a single filter. The conv + CReLU + max-pooling layer can be defined by first multiplying an input signal x by the matrix W^T (conv), separating positive and negative phases then applying the ReLU non-linearity (CReLU), and selecting the maximum value in each of the K block (max-pooling). The operation is denoted as $f_{\text{cnn}} : \mathbb{R}^D \to \mathbb{R}^{2K}$ such that $f_{\text{cnn}}(x) \triangleq g\left(W^T x\right)$, where $g \triangleq \text{pool} \circ \text{CReLU}$. Figure S1 illustrates an example of the problem setting.

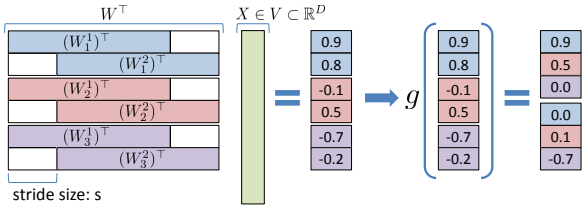


Figure S1. **An illustration of convolution,** CReLU**, and max-pooling operation.** For simplicity, we describe with 3 convolution filters (W_1, W_2, W_3) with stride of s, and with 2×2 pooling. In Figure (a), g denotes CReLU followed by the max-pooling operation.

Assumption. To reach a non-trivial bound when max-pooling is present, we put a constraint on the input space $\mathcal{V}$: $\forall x \in \mathcal{V}$, there exists $\{c_i^j\}_{i=1,\cdots,K}^{j=1,\cdots,n}$ such that

$$x = \sum_{i=1}^{K} \sum_{j=1}^{n} c_i^j w_i^j, \text{ where } \sum_{j=1}^{n} \mathbf{1}\{c_i^j > 0\} \leq 1, \quad \forall i. \quad \text{(S1)}$$

In other words, we assume that an input x is a linear combination of the shifted convolution filters $\{w_i^j\}_{i=1,\cdots,K}^{j=1,\cdots,n}$ such that over a single max-pooling region, only one of the shifts participates: $\sum_{j=1}^{n} \mathbf{1}\{c_i^j > 0\} \leq 1$: a slight translation of an object or viewpoint change does not alter the nature of a natural image, which is how max-pooling generates shift invariant features by taking away some fine-scaled locality information.

Next, we denote the matrix consisting of the shifts whose corresponding c_i^j's are non-zero by W_x , and the vector consisting of the non-zero c_i^j's by $\mathbf{c}_x$, i.e. $W_x \mathbf{c}_x = x$. Also, we denote the matrix consisting of the shifts whose activation is positive and selected after max-pooling operation by $\widehat{W}_x^+$, negative by $\widehat{W}_x^-$. Let $\widehat{W}_x \triangleq \left[\widehat{W}_x^+, \widehat{W}_x^-\right]$. Finally, we give notation, $\widetilde{W}_x$, to the matrix consisting of a subset of $\widehat{W}_x$, such that the ith column comes from $\widehat{W}_x^+$ if $c_i^j \geq 0$ or from $\widehat{W}_x^-$ if otherwise.

Frame Theory. Before proceeding to the main theorem and its proof, we would like to introduce more tools from Frame Theory.

Definition A.1. *A frame is a set of elements of a vector space V, $\{\phi_k\}_{k=1,\cdots,K}$, which satisfies the frame condition: there exist two real numbers C_1 and C_2, the* frame bounds*, such that $0 < C_1 \leq C_2 < \infty$, and $\forall v \in V$*

$$C_1\|v\|_2^2 \leq \sum_{k=1}^{K} |\langle v, \phi_i\rangle|^2 \leq C_2\|v\|_2^2.$$

(Christensen, 2003)

Proposition A.2. *Let $\{\phi_k\}_{k=1,\ldots,K}$ be a sequence in V, then $\{\phi_k\}$ is a frame for* $\mathrm{span}\{\phi_k\}$*. Hence, $\{\phi_k\}$ is a frame for V if and only if $V = \mathrm{span}\{\phi_k\}$*[7]*. (Christensen, 2003)*

Definition A.2. *Consider now V equipped with a frame $\{\phi_k\}_{k=1,\ldots,K}$. The* Analysis Operator*, $\mathcal{T} : V \to \mathbb{R}^K$, is defined by $\mathcal{T}v = \{\langle v, \phi_k\rangle\}_{k=1,\ldots,K}$. The* Synthesis Operator*, $\mathcal{T}^* : \mathbb{R}^K \to V$, is defined by $\mathcal{T}^*\{c_k\}_{k=1,\ldots,K} = \sum_{k=1}^{K} c_k\phi_k$, which is the adjoint of the Analysis Operator. The* Frame Operator*, $\mathcal{S} : V \to V$, is defined to be the composition of $\mathcal{T}$ with its adjoint:*

$$\mathcal{S}v = \mathcal{T}^*\mathcal{T}v.$$

The Frame Operator is always invertible. (Christensen, 2003)

Theorem A.3. *The optimal lower frame bound C_1 is the smallest eigenvalue of $\mathcal{S}$; the optimal upper frame bound C_2 is the largest eigenvalue of $\mathcal{S}$. (Christensen, 2003)*

We would also like to investigate the matrix representation of the operators $\mathcal{T}, \mathcal{T}^*$ and $\mathcal{S}$. Consider V, a subspace of $\mathbb{R}^D$, equipped with a frame $\{\phi_k\}_{k=1,\cdots,K}$. Let $U \in R^{D\times d}$ be a matrix whose column vectors form an orthonormal basis for V (here d is the dimension of V). Choosing U as the basis for V and choosing the standard basis $\{e_k\}_{k=1,\cdots,K}$ as the basis for R^K, the matrix representation of $\mathcal{T}$ is $\tilde{\mathcal{T}} = W^TU$, where W is the matrix whose column vectors are $\{\phi_k^T\}_{k=1,\cdots,K}$. Its transpose, $\tilde{\mathcal{T}}^*$, is the matrix representation for $\mathcal{T}^*$; the matrix representation for $\mathcal{S}$ is $\tilde{\mathcal{S}} = \tilde{\mathcal{T}}^*\tilde{\mathcal{T}}$.

Lemma A.4. *Let $x \in \mathbb{R}^D$ and W an D-by-K matrix. If $x \in \mathrm{range}(W)$, then $\sigma_{\min}\|x\|_2 \leq \|W^Tx\|_2 \leq \sigma_{\max}\|x\|_2$, where $\sigma_{\min}$ and $\sigma_{\max}$ are the least and largest singular value of V respectively.*

Proof. By Proposition A.2, the columns in W form a frame for $\mathrm{range}(W)$. Let U be an orthonormal basis for $\mathrm{range}(W)$. Then the matrix representation under U for the Analysis Operator, $\mathcal{T}$, is $\tilde{\mathcal{T}} = W^TU$, and the corresponding representation for x under U is U^Tx. Now, by Theorem A.3, we have:

$$\lambda_{\min}\|x\|_2^2 \leq \|\tilde{\mathcal{T}}x\|_2^2 = \|W^TUU^Tx\|_2^2 = \|W^Tx\|_2^2,$$

where λ_1 is the least eigenvalue of $\tilde{\mathcal{S}}$. Therefore, we have $\sigma_{\min}\|x\|_2 \leq \|W^Tx\|_2$, where $\sigma_{\min}$ is the least singular value of W. Lastly, by the definition of operator-induced matrix norm, we have the upper bound $\|W^Tx\|_2 \leq \sigma_{\max}\|x\|_2$ □

Reconstruction Property. Now we are ready to present the theorem that characterizes the reconstruction property of the conv+CReLU+max-pooling operation.

Theorem A.5. *Let $x \in \mathcal{V}$ and satisfy the assumption from Equation (S1). Then we can obtain x', the reconstruction of x using $f_{\mathrm{cnn}}(x)$ such that*

$$\frac{\|x - x'\|_2}{\|x\|_2} \leq \sqrt{\frac{\widetilde{\lambda}_{\max} - \lambda_{\min}}{\lambda_{\min}}},$$

where $\lambda_{\min}$ and $\widetilde{\lambda}_{\max}$ are square of the minimum and maximum singular values of W_x and $\widetilde{W}_x$ respectively.

Proof. We use similar method to reconstruct as described by Algorithm 2: first reverse the CReLU activation and obtain $z = \rho_c^{-1}(f_{\mathrm{cnn}}(x))$; then compute the Moore Penrose pseudoinverse of $\widehat{W}_x^T$, denote by $(\widehat{W}_x^T)^+$; finally, obtain $x' = (\widehat{W}_x^T)^+z$, since by definition, $Q = (\widehat{W}_x^T)^+\widehat{W}_x^T$ is the orthogonal projector onto $\mathrm{range}(\widehat{W}_x)$. To proceed the proof, we denote the subset of z which matches the corresponding activation of the filters from $\widetilde{W}_x$ by $\tilde{z}$, compute the Morre Penrose pseudoinverse of $\widetilde{W}_x$ and obtain $\tilde{x} = (\widetilde{W}_x^T)^+\tilde{z}$. Note that since $\mathrm{range}(\widetilde{W}_x)$ is a subspace of $\mathrm{range}(\widehat{W}_x)$, therefore, the reconstruction x' will always be equal or better than $\tilde{x}$, i.e. $\|x - x'\|_2 \leq \|x - \tilde{x}\|_2$. From Lemma A.4, the nature of max-pooling and the assumption on x (Equation S1), we derive the following inequality

$$\begin{aligned}\lambda_{\min}\|x\|_2^2 \leq \|W_x^Tx\|_2 &\leq \|\widetilde{W}_x^Tx\|_2^2 \\ &= \|\widetilde{W}_x^T\tilde{x}\|_2^2 \leq \tilde{\lambda}_{\max}\|\tilde{x}\|_2^2,\end{aligned}$$

where $\lambda_{\min}$ and $\tilde{\lambda}_{\max}$ are square of the minimum and maximum singular values of W_x and $\widetilde{W}_x$ respectively.

Because $\tilde{x}$ is the orthogonal projection of x on to $\mathrm{range}(\widetilde{W}_x)$, thus $\|x\|_2^2 = \|\tilde{x}\|_2^2 + \|x - \tilde{x}\|_2^2$. Now substitute

[7] There exist infinite spanning sets that are not frames, but we will not be concerned with those here since we only deal with finite dimensional vector spaces.

$\|x\|_2^2$ with $\|\tilde{x}\|_2^2 + \|x - \tilde{x}\|_2^2$, we have:

$$\lambda_{\min}(\|\tilde{x}\|_2^2 + \|x - \tilde{x}\|_2^2) \leq \tilde{\lambda}_{\max}\|\tilde{x}\|_2^2$$
$$\|x - \tilde{x}\|_2^2 \leq \frac{\tilde{\lambda}_{\max} - \lambda_{\min}}{\tilde{\lambda}_{\min}}\|\tilde{x}\|_2^2$$
$$\|x - x'\|_2^2 \leq \frac{\tilde{\lambda}_{\max} - \lambda_{\min}}{\lambda_{\min}}\|x\|_2^2$$
$$\|x - x'\|_2 \leq \sqrt{\frac{\tilde{\lambda}_{\max} - \lambda_{\min}}{\lambda_{\min}}}\|x\|_2$$
$$\frac{\|x - x'\|_2}{\|x\|_2} \leq \sqrt{\frac{\tilde{\lambda}_{\max} - \lambda_{\min}}{\lambda_{\min}}}.$$

□

We refer to the term $\frac{\|x-x'\|_2}{\|x\|_2}$ as the *reconstruction ratio* in later discussions.

B. Proof of Model Complexity Bound

Definition B.1. (Rademacher Complexity) *For a sample $S = \{x_1, \cdots, x_L\}$ generated by a distribution D on set X and a real-valued function class $\mathcal{F}$ in domain X, the empirical Rademacher complexity of $\mathcal{F}$ is the random variable:*

$$\hat{R}_L(\mathcal{F}) = \mathbf{E}_\sigma\left[\sup_{f\in\mathcal{F}}|\frac{2}{L}\sigma_i f(x_i)|\Big| x_1, \cdots, x_L\right],$$

where σ_i's are independent uniform $\{\pm 1\}$-valued (Rademacher) random variables. The Rademacher complexity of $\mathcal{F}$ is $R_L(\mathcal{F}) = \mathbf{E}_S\left[\hat{R}_L(\mathcal{F})\right]$

Lemma B.1. (Composition Lemma) *Assume $\rho : \mathbb{R} \to \mathbb{R}$ is a L_ρ-Lipschitz continuous function, i.e. , $|\rho(x) - \rho(y)| \leq L_\rho|x - y|$. Then $\hat{R}_L(\rho \circ \mathcal{F}) = L_\rho\hat{R}_L(\mathcal{F})$.*

Proposition B.2. (Network Layer Bound) *Let $\mathcal{G}$ be the class of real functions $\mathbb{R}^{d_{in}} \to \mathbb{R}$ with input dimension $\mathcal{F}$, that is, $\mathcal{G} = [\mathcal{F}]_{j=1}^{d_{in}}$ and $\mathcal{H}$ is a linear transform function parametrized by W with $\|W\|_2 \leq B$, then $\hat{R}_L(\mathcal{H} \circ \mathcal{G}) \leq \sqrt{d_{in}}B\hat{R}_L(\mathcal{F})$. (Wan et al., 2013)*

Corollary B.3. *By Lemma B.1, Proposition B.2, and the fact that* ReLU *is 1-Lipschitz, we know that $\hat{R}_L(\text{ReLU} \circ \mathcal{G}) = \hat{R}_L(\mathcal{G})$ and that $\hat{R}_L(\mathcal{H}\circ\text{ReLU}\circ\mathcal{G}) \leq \sqrt{d_{in}}B\hat{R}_L(\mathcal{F})$.*

Theorem B.4. *4.1 Let $\mathcal{G}$ be the class of real functions $\mathbb{R}^{d_{in}} \to \mathbb{R}$ with input dimension $\mathcal{F}$, that is, $\mathcal{G} = [\mathcal{F}]_{j=1}^{d_{in}}$. Let $\mathcal{H}$ be a linear transform function from $\mathbb{R}^{2d_{in}}$ to $\mathbb{R}$, parametrized by W, where $\|W\|_2 \leq B$. Then $\hat{R}_L(\mathcal{H} \circ \rho_c \circ \mathcal{G}) \leq \sqrt{d_{in}}B\hat{R}_L(\mathcal{F})$.*

Recall from Definition 2.1, ρ_c is the CReLU formulation.

Table S1. **Empirical mean of the reconstruction ratios.** Reconstruct the sampled images from test set using the features after CReLU and max-pooling; then calculate the reconstruction ratio, $\|x - x'\|_2/\|x\|_2$.

CIFAR-10		
layer	learned	random
conv2	0.92 $\pm$0.0002	0.99 $\pm$0.00005
conv5	0.96 $\pm$0.0003	0.99 $\pm$0.00005
CIFAR-100		
layer	learned	random
conv2	0.93 $\pm$0.0002	0.99 $\pm$0.00005
conv5	0.96 $\pm$0.0001	0.99 $\pm$0.00005

Proof.

$$\hat{R}_L(\mathcal{H} \circ \rho_c \circ \mathcal{G}) = \mathbf{E}_\sigma\left[\sup_{h\in\mathcal{H},g\in\mathcal{G}}|\frac{2}{L}\sum_{i=1}^{L}\sigma_i h \circ \rho_c \circ g(x_i)|\right] \tag{S2}$$
$$= \mathbf{E}_\sigma\left[\sup_{\|W\|\leq B,g\in\mathcal{G}}|\langle W, \frac{2}{L}\sum_{i=1}^{L}\sigma_i\rho_c \circ g(x_i)\rangle|\right] \tag{S3}$$
$$\leq B\mathbf{E}_\sigma\left[\sup_{f\in\mathcal{F}}\|\left[\frac{2}{L}\sum_{i=1}^{L}\sigma_i^j\rho_c \circ f^j(x_i)\right]_{j=1}^{d_{\text{in}}}\|_2\right] \tag{S4}$$
$$= B\mathbf{E}_\sigma\left[\sup_{f\in\mathcal{F}}\|\left[\frac{2}{L}\sum_{i=1}^{L}\sigma_i^j f^j(x_i)\right]_{j=1}^{d_{\text{in}}}\|_2\right] \tag{S5}$$
$$= B\sqrt{d_{\text{in}}}\mathbf{E}_\sigma\left[\sup_{f\in\mathcal{F}}|\frac{2}{L}\sum_{i=1}^{L}\sigma_i f(x_i)|\right] \tag{S6}$$
$$= \sqrt{d_{\text{in}}}B\hat{R}_L(\mathcal{F}). \tag{S7}$$

From (S1) to (S2), use the definition of linear transformation and inner product. From (S2) to (S3), use Cauchy-Schwarz inequality and the assumption that $\|W\|_2 \leq B$. From (S3) to (S4), use the definition of CReLU and l^2 norm. From (S4) to (S5), use the definition of l^2 norm and sup operator. From (S5) to (S6), use the definition of $\hat{R}_L$ □

We see that CReLU followed by linear transformation reaches the same Rademacher complexity bound as ReLU followed by linear transformation with the same input dimension.

C. Reconstruction Ratio

Recall that Theorem A.5 characterizes the reconstruction property when max-pooling is added after CReLU. As an example, we study the all-conv CReLU (half) models used for CIFAR-10/100 experiments. In this model, conv2 and conv5 layers are followed by max-pooling. CIFAR images are much less diverse than those from ImageNet. Instead

of directly inverting features all the way back to the original images, we empirically calculate the reconstruction ratio, $\|x - x'\|_2/\|x\|_2$. We sample testing examples, extract pooled features after conv2(conv5) layer and reconstruct features from the previous layer via Algorithm 2. To compare, we perform the same procedures on random convolution filters[8]. Essentially, convolution imposes structured zeros to the random $\widetilde{W}_x$; there has not been published results on random subspace projection with such structured zeros. In a simplified setting without structured zeros, i.e. no convolution, it is straightforward to show that the expected reconstruction ratio is $\sqrt{\frac{D-K}{D}}$ (Theorem C.1), where, in our case, $D = 48(96) \times 5 \times 5$ and $K = 48(96)$ for conv2(conv5) layer. Table S1 compares between the empirical mean of reconstruction ratios using learned filters and random filters: random filters only recover 1% of the original input, whereas the learned filters span more of the input domain.

Theorem C.1. *Let $x \in \mathbb{R}^D$, and let $x_s \in \mathbb{R}^D$ be its projection onto a random subspace of dimension D_2, then*

$$\mathbf{E}\left[\frac{\|x_s\|_2}{\|x\|_2}\right] = \sqrt{\frac{D_s}{D}}$$

Proof. Without loss of generality, let $\|x\|_2 = 1$. Projecting a fixed x onto a random subspace of dimension D_s is equivalent of projecting a random unit-norm vector $z = (z_1, z_2, \cdots, z_D)^T$ onto a fixed subspace of dimension D_s thanks to the rotational invariance of inner product. Without loss of generality, assume the fixed subspace here is spanned by the first D_s standard basis covering the first D_2 coordinates of z. Then the resulting projection is $z_s = (z_1, z_2, \cdots, z_{D_s}, 0, \cdots, 0)$.

Because z is unit norm, we have

$$\mathbf{E}\left[\|z\|_2^2\right] = \mathbf{E}\left[\sum_{i=1}^{D} z_i^2\right] = 1.$$

Because each entry of z, z_i, is identically distributed, we have

$$\mathbf{E}\left[\|z_s\|_2^2\right] = \mathbf{E}\left[\sum_{i=1}^{D_s} z_i^2\right] = \frac{D_s}{D}.$$

Together we have

$$\mathbf{E}\left[\frac{\|x_s\|_2}{\|x\|_2}\right] = \mathbf{E}\left[\frac{\|z_s\|_2}{\|z\|_2}\right] = \sqrt{\frac{D_s}{D}}.$$

□

[8]Each entry is sampled from standard normal distribution.

D. Invariance Score

We use consistent terminology employed by Goodfellow et al. (2009) to illustrate the calculation of the invariance scores.

For CIFAR-10/100, we utilize all 50k testing images to calculate the invariance scores; for ImageNet, we take the center crop from 5k randomly sampled validation images

For each individual filter, we calculate its own *firing* threshold, such that it is fired one percent of the time, i.e. the *global firing rate* is 0.01. For ReLU models, we zero out all the negative negative responses when calculating the threshold; for CReLU models, we take the absolute value.

To build the set of semantically similar stimuli for each testing image x, we apply horizontal flip, 15 degree rotation and translation. For CIFAR-10/100, translation is composed of horizontal/vertical shifts by 3 pixels; for ImageNet, translation is composed of cropping from the 4 corners.

Because our setup is convolutional, we consider a filter to be fired only if both the transformed stimulus and the original testing example fire the same convolution filter at the *same* spatial location.

At the end, for each convolution layer, we average the invariance scores of all the filters at this layer to form the final score.

E. Implementation Details on ImageNet Models

The networks from Table S8, S9, S10,and S11, where the number of convolution filters after CReLU are kept the same, are optimized using SGD with mini-batch size of 64 examples and fixed momentum 0.9. The learning rate and weight decay is adapted using the following schedule: epoch 1-10, $1\mathrm{e}{-2}$ and $5\mathrm{e}{-4}$; epoch 11-20, $1\mathrm{e}{-3}$ and $5\mathrm{e}{-4}$; epoch 21-25, $1\mathrm{e}{-4}$ and $5\mathrm{e}{-4}$; epoch 26-30, $5\mathrm{e}{-5}$ and 0; epoch 31-35, $1\mathrm{e}{-5}$ and 0; epoch 36-40, $5\mathrm{e}{-6}$ and 0; epoch 41-45, $1\mathrm{e}{-6}$ and 0.

The networks from Table S12 and S13, where the number of convolution filters after CReLU are reduced by half, are optimized using Adam with an initial learning rate 0.0002 and mini-batch size of 64 examples for 100 epochs.

F. Details of Network Architecture

Table S2. (Left) Baseline or AVR and (right) baseline (double) models used for CIFAR-10/100 experiment. "avg" refers average pooling.

	Baseline/AVR		Baseline (double)	
Layer	kernel, stride, padding	activation	kernel, stride, padding	activation
conv1	3×3×3×96, 1, 1	ReLU/AVR	3×3×3×192, 1, 1	ReLU
conv2	3×3×96×96, 1, 1	ReLU/AVR	3×3×192×192, 1, 1	ReLU
pool1	3×3, 2, 0	max	3×3, 2, 0	max
conv3	3×3×96×192, 1, 1	ReLU/AVR	3×3×192×384, 1, 1	ReLU
conv4	3×3×192×192, 1, 1	ReLU/AVR	3×3×384×384, 1, 1	ReLU
conv5	3×3×192×192, 1, 1	ReLU/AVR	3×3×384×384, 1, 1	ReLU
pool2	3×3, 2, 0	max	3×3, 2, 0	max
conv6	3×3×192×192, 1, 1	ReLU/AVR	3×3×384×384, 1, 1	ReLU
conv7	1×1×192×192, 1, 1	ReLU/AVR	1×1×384×384, 1, 1	ReLU
conv8	1×1×192×10/100, 1, 0	ReLU/AVR	1×1×384×10/100, 1, 0	ReLU
pool3	10×10 (100 for CIFAR-100)	avg	10×10 (100 for CIFAR-100)	avg

Table S3. (Left) CReLU and (right) CReLU (half) models used for CIFAR-10/100 experiment.

	CReLU		CReLU (half)	
Layer	kernel, stride, padding	activation	kernel, stride, padding	activation
conv1	3×3×3×96, 1, 1	CReLU	3×3×3×48, 1, 1	CReLU
conv2	3×3×192×96, 1, 1	CReLU	3×3×96×48, 1, 1	CReLU
pool1	3×3, 2, 0	max	3×3, 2, 0	max
conv3	3×3×192×192, 1, 1	CReLU	3×3×96×48, 1, 1	CReLU
conv4	3×3×384×192, 1, 1	CReLU	3×3×96×96, 1, 1	CReLU
conv5	3×3×384×192, 1, 1	CReLU	3×3×192×96, 1, 1	CReLU
pool2	3×3, 2, 0	max	3×3, 2, 0	max
conv6	3×3×384×192, 1, 1	CReLU	3×3×192×96, 1, 1	CReLU
conv7	1×1×384×192, 1, 1	CReLU	1×1×192×96, 1, 1	CReLU
conv8	1×1×384×10/100, 1, 0	ReLU	1×1×192×10/100, 1, 0	ReLU
pool3	10×10 (100 for CIFAR-100)	avg	10×10 (100 for CIFAR-100)	avg

Table S4. VGG for CIFAR-10/100

Layer	kernel, stride, padding	activation
conv1	3×3×3×64, 1, 1	BN+ReLU
	dropout with ratio 0.3	
conv2	3×3×64×64, 1, 1	BN+ReLU
pool1	2×2, 2, 0	
conv3	3×3×64×128, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv4	3×3×128×128, 1, 1	BN+ReLU
pool2	2×2, 2, 0	
conv5	3×3×128×256, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv6	3×3×256×256, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv7	3×3×256×256, 1, 1	BN+ReLU
pool3	2×2, 2, 0	
conv8	3×3×256×512, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv9	3×3×512×512, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv10	3×3×512×512, 1, 1	BN+ReLU
pool4	2×2, 2, 0	
conv11	3×3×512×512, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv12	3×3×512×512, 1, 1	BN+ReLU
	dropout with ratio 0.4	
conv13	3×3×512×512, 1, 1	BN+ReLU
pool5	2×2, 2, 0	
	dropout with ratio 0.5	
fc14	512×512	BN+ReLU
	dropout with ratio 0.5	
fc15	512×10/100	

Table S5. VGG + (conv1) for CIFAR-10/100

Layer	kernel, stride, padding	activation
conv1	3×3×3×32, 1, 1	CReLU
	dropout with ratio 0.1	
conv2	···	

Table S6. VGG + (conv1, 3) for CIFAR-10/100

Layer	kernel, stride, padding	activation
conv1	3×3×3×32, 1, 1	CReLU
	dropout with ratio 0.1	
conv2	3×3×64×64, 1, 1	BN+ReLU
pool1	2×2, 2, 0	
conv3	3×3×64×64, 1, 1	CReLU
	dropout with ratio 0.2	
conv4	···	

Table S7. VGG + (conv1, 3, 5) for CIFAR-10/100

Layer	kernel, stride, padding	activation
conv1	3×3×3×32, 1, 1	CReLU
	dropout with ratio 0.1	
conv2	3×3×64×64, 1, 1	BN+ReLU
pool1	2×2, 2, 0	
conv3	3×3×64×64, 1, 1	CReLU
	dropout with ratio 0.2	
conv4	3×3×128×128, 1, 1	BN+ReLU
pool2	2×2, 2, 0	
conv5	3×3×128×128, 1, 1	CReLU
	dropout with ratio 0.2	
conv6	3×3×256×256, 1, 1	BN+ReLU
	dropout with ratio 0.2	
conv7	···	

Table S8. Baseline for ImageNet

Layer	kernel, stride, padding	activation
conv1	11×11×3×96, 4,0	ReLU
conv2	1×1×96×96, 1,0	ReLU
conv3	3×3×96×96, 2,0	ReLU
conv4	5×5×96×256, 1, 2	ReLU
conv5	1×1×256×256, 1,0	ReLU
conv6	3×3×256×256, 2,0	ReLU
conv7	3×3×256×384, 1 ,1	ReLU
conv8	1×1×384×384, 1,0	ReLU
conv9	3×3×384×384, 2,1	ReLU
	no dropout	
conv10	3×3×384×1024, 1,1	ReLU
conv11	1×1×1024×1024, 1,0	ReLU
conv12	1×1×1024×1000, 1	ReLU
pool	6×6 average-pooling	

Table S9. CReLU/AVR (conv1-4) for ImageNet

Layer	kernel, stride, padding	activation
conv1	11×11×3×96, 4,0	CReLU/AVR
conv2	1×1×192/96×96, 1,0	CReLU/AVR
conv3	3×3×192/96×96, 2,0	CReLU/AVR
conv4	5×5×192/96×256, 1, 2	CReLU/AVR
conv5	1×1×512/256×256, 1,0	ReLU
conv6	3×3×256×256, 2,0	ReLU
conv7	3×3×256×384, 1 ,1	ReLU
conv8	1×1×384×384, 1,0	ReLU
conv9	3×3×384×384, 2,1	ReLU
	no dropout	
conv10	3×3×384×1024, 1,1	ReLU
conv11	1×1×1024×1024, 1,0	ReLU
conv12	1×1×1024×1000, 1	ReLU
pool	6×6 average-pooling	

Table S10. CReLU/AVR (conv1-7) for ImageNet

Layer	kernel, stride, padding	activation
conv1	11×11×3×96, 4,0	CReLU/AVR
conv2	1×1×192/96×96, 1,0	CReLU/AVR
conv3	3×3×192/96×96, 2,0	CReLU/AVR
conv4	5×5×192/96×256, 1, 2	CReLU/AVR
conv5	1×1×512/256×256, 1,0	CReLU/AVR
conv6	3×3×512/256×256, 2,0	CReLU/AVR
conv7	3×3×512/256×384, 1 ,1	CReLU/AVR
conv8	1×1×768/384×384, 1,0	ReLU
conv9	3×3×384×384, 2,1	ReLU
	dropout with ratio 0.25	
conv10	3×3×384×1024, 1,1	ReLU
conv11	1×1×1024×1024, 1,0	ReLU
conv12	1×1×1024×1000, 1	ReLU
pool	6×6 average-pooling	

Table S11. CReLU/AVR (conv1-9) for ImageNet

Layer	kernel, stride, padding	activation
conv1	11×11×3×96, 4,0	CReLU/AVR
conv2	1×1×192/96×96, 1,0	CReLU/AVR
conv3	3×3×192/96×96, 2,0	CReLU/AVR
conv4	5×5×192/96×256, 1, 2	CReLU/AVR
conv5	1×1×512/256×256, 1,0	CReLU/AVR
conv6	3×3×512/256×256, 2,0	CReLU/AVR
conv7	3×3×512/256×384, 1 ,1	CReLU/AVR
conv8	1×1×768/384×384, 1,0	CReLU/AVR
conv9	3×3×768/384×384, 2,1	CReLU/AVR
	dropout with ratio 0.25	
conv10	3×3×768/384×1024, 1,1	ReLU
conv11	1×1×1024×1024, 1,0	ReLU
conv12	1×1×1024×1000, 1	ReLU
pool	6×6 average-pooling	

Table S12. CReLU (all) for ImageNet

Layer	kernel, stride, padding	activation
conv1	11×11×3×48, 4,0	CReLU
conv2	1×1×96×48, 1,0	CReLU
conv3	3×3×96×48, 2,0	CReLU
conv4	5×5×96×128, 1, 2	CReLU
conv5	1×1×256×128, 1,0	CReLU
conv6	3×3×256×128, 2,0	CReLU
conv7	3×3×256×192, 1 ,1	CReLU
conv8	1×1×384×192, 1,0	CReLU
conv9	3×3×384×192, 2,1	CReLU
	dropout with ratio 0.25	
conv10	3×3×384×512, 1,1	CReLU
conv11	1×1×512×512, 1,0	CReLU
conv12	1×1×512×1000, 1	CReLU
pool	6×6 average-pooling	

Table S13. CReLU (conv1,4,7) for ImageNet

Layer	kernel, stride, padding	activation
conv1	11×11×3×48, 4,0	CReLU
conv2	1×1×96×96, 1,0	ReLU
conv3	3×3×96×96, 2,0	ReLU
conv4	5×5×96×128, 1, 2	CReLU
conv5	1×1×256×256, 1,0	ReLU
conv6	3×3×256×256, 2,0	ReLU
conv7	3×3×256×192, 1 ,1	CReLU
conv8	1×1×384×384, 1,0	ReLU
conv9	3×3×384×384, 2,1	ReLU
	dropout with ratio 0.25	
conv10	3×3×384×1024, 1,1	ReLU
conv11	1×1×1024×1024, 1,0	ReLU
conv12	1×1×1024×1000, 1	ReLU
pool	6×6 average-pooling	

G. Image Reconstruction

In this section, we provide more image reconstruction examples.

(a) Original image (b) conv1 (c) conv2 (d) conv3 (e) conv4

(f) Original image (g) conv1 (h) conv2 (i) conv3 (j) conv4

(k) Original image (l) conv1 (m) conv2 (n) conv3 (o) conv4

(p) Original image (q) conv1 (r) conv2 (s) conv3 (t) conv4