LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-741553

Improvements to context based self-supervised learning.

T. Nathan Mundhenk, D. Ho, B. Chen

November 13, 2017

CVPR
Salt Lake City, UT, United States
June 18, 2018 through June 22, 2018

Disclaimer

Improvements to context based self-supervised learning.

Anonymous CVPR submission

Paper ID ****

Abstract

We develop a set of methods to improve on the results of self-supervised learning using context. We start with a baseline of patch based arrangement context learning and go from there. Our methods address some overt problems such as chromatic aberration as well as other potential problems such as spatial skew and mid-level feature neglect. We prevent problems with testing generalization on common self-supervised benchmark tests by using different datasets during our development. The results of our methods combined yield top scores on all *standard self-supervised benchmarks, including classification and detection on PASCAL VOC 2007, segmentation on PASCAL VOC 2012, and "linear tests" on the ImageNet and CSAIL Places datasets. We also show results on different standard network architectures to demonstrate generalization as well as portability.*

1. Introduction

Self-supervised learning has opened an intriguing new avenue into unsupervised learning. It is intellectually satisfying due to the way it resembles gestalt-like mechanisms of learning in visual cortical formation. It is also appealing for that fact that it can be implemented with standard off-the-shelf neural networks and toolkits.

Self-supervised learning methods create a protocol whereby the computer can learn to teach itself a supervised task. For instance, in [10] a convolutional neural network (CNN) was taught to learn the arrangement of patches in an image. By learning the relative position of these patches, the network would be forced to also learn the features and semantics that underlie the image. Although the network was trained to learn patch positions, the final goal was to generalize the learned representation to solve other tasks. For instance, the self-supervised network was trained on a transfer task (fine-tuned) to classify objects in the PASCAL VOC dataset [14], and compared with a CNN trained on a supervised task, such as learning to classify the ImageNet dataset [8]. If the self-supervised network learned good generalizations of image features and semantics, it should perform as well as a supervised network on transfer learning.

Over the last few years, several methods of self-supervised learning have been introduced. For instance, [13] trained a CNN to recognize which transformation had been performed on an image. Since then, methods have been introduced that use context arrangement of image patches [10, 31], image completion [47, 36] image colorization [46, 24] motion segmentation [35], motion frame ordering [42, 43, 25] object counting [33] and an ensemble of many models [11].

Each method has relative strengths and weaknesses. For instance, [46] uses a customized "split-brain" architecture that makes it less off-the-shelf than other solutions which frequently use a standard network such as AlexNet [23]. [42, 43, 25, 35] all use motion cues, but have the downside to being constrained to video images. Many patch based methods use a Siamese network architecture which requires extra memory demands [10, 31, 33, 25]. However, every self-supervised method suffers the drawback of still being very short of supervised methods in terms of transfer learning performance.

Our intent in this work is to improve the performance of self-supervised learning. We present a variety of techniques we hope are applicable to other methods as well. We will demonstrate generalizability of these techniques by running them on several different neural network architectures and many kinds of datasets and tasks. For instance, as we will discuss, one dataset we wish to add to the corpus of standard self-supervised tests, the Caltech-UCSD Birds 200 set (CUB) [44] is excellent for finding potential short comings of techniques designed to address the well-known *chromatic aberration* problem [10]. We will show this is due to the importance of color patterns in bird classification [7].

2. Issues and Related Work

We use a patch/context approach for the issue of self-supervised learning [10, 31]. This is a popular method, but is by no means the only active path of inquiry. Patch/context

approaches work by creating an arrangement of image patches in either space or time. Each distinct arrangement is assigned a class label, and the network then predicts the correct arrangement of these patches by solving a supervised classification problem. The network can be a typical supervised network such as AlexNet [23], VGG [38], GoogLeNet [39] or ResNet [17]. In order to view multiple patches at the same time, a Siamese network is frequently used where each patch is feed into an independent network path, and each path shares weights. [10] used a system of two patches in a finite set of eight possible spatial configurations. [31] created an extension using as many as nine patches in a *puzzle* configuration. Temporal ordering of patches can also be used. For instance, [25] shuffled four consecutive video frames to create 12 classes for prediction. Another temporal method [43] queried the networks ability to determine if a patch came from the same object later in time or a similar but different object. This can be considered a meta self-learner since it leverages [10] as a pre-self-supervised learner to determine object similarity.

Patch based methods have the advantage of being easy to understand, network architecture agnostic, and frequently straightforward to implement. They also tend to perform well on standard measures of transfer learning. For instance, [10] is a top performer on PASCAL VOC 2007 detection [15], even among a large number of new arrivals. [32] is almost tied for the top score on PASCAL VOC 2007 classification [22] and has the top score for Pascal VOC 2007 detection and the second highest score for PASCAL VOC 2012 segmentation [28, 36].

However, patch/context-based networks suffer from a typical issue of being able to "cheat" and bypass learning the desired proper semantic structure by finding incidental clues that reveal the location of a patch. An example of this is *chromatic aberration*, which occurs naturally as a result of camera lensing where different frequencies of light exit the lens at slightly different angles. This radially offsets colors such as magenta and green modestly from the center of the image. By looking at the offset relation of different color channels, a network can determine the angular location of a patch in an image. Aberration varies between lenses, so its an imperfect cue, but one that none-the-less exists. A common remedy is to withhold color data from one or more channels by using channel-dropping [10, 11], channel replication [25], or conversion to gray scale [33]. The primary difficulty with these approaches is that color becomes decorrelated (or absent) since colors are not observed together. This makes it difficult to learn color opponents for patterns that emerge in supervised training sets, such as ImageNet. Another approach is to jitter color channels [31], but this has a similar effect to blurring an image, and might affect the sharpness of learned wavelet features.

An often-cited worry in all patch/context works relates to trivial low-level boundary pattern completion [10, 31, 11, 33, 25]. The neural network may learn the alignment of patches not based on (for instance) desirable semantic information, but instead by matching the top or bottom part of simple line segments. Two common approaches are to provide a large enough gap between patches and to randomly jitter the patches. This last technique may be dubious since a convolutional neural network can align simple patterns at arbitrary offsets. This issue may also be implicitly addressed by having non-4-connected adjacent patches. Half the patches in [10] are arranged diagonally which should make them resistant to trivial low-level boundary pattern completion. Also, we should note that while we would not want a self-supervised learner to use this cheat all the time, it could be used as a cue to help form low level features. So, it is somewhat unclear how much of a problem this really is.

In another possible problem for self-supervised networks in general, mid-layers in the network may not train as well as the early and later layers. For instance, [11] created a self-supervised network using an ensemble of different methods. They then created an automated lasso to grab layers in the network most useful for their task. The lasso tended to grab layers very early or very late in the network. This suggests that for many self-supervised tasks, the information in the middle network layers is not very essential for training. Another piece of evidence comes from the CSAIL Places linear test [46], which shows how well each layer in the network performs on transfer learning. Many self-supervised networks perform as well or better than a supervised ImageNet trained network at the first and second convolutional layers in AlexNet, but struggle at deeper layers.

3. Approach

Our approach is comprised of three parts. The first is a collection of tools and enhancements which for the most part should be transferable to other self-supervised methods. In our second part, we utilize two new datasets to make our experiments more diverse and general. The third part is a demonstration on several different neural networks to verify generalization and demonstrate portability.

For our general approach, we start with the baseline of [10] using a two-patch spatial configuration paradigm and add augmentation to improve results. This approach gives good results, and is easy to both implement and understand. We then augment this approach using various techniques. For each technique, we vigorously test effects empirically to justify their usage.

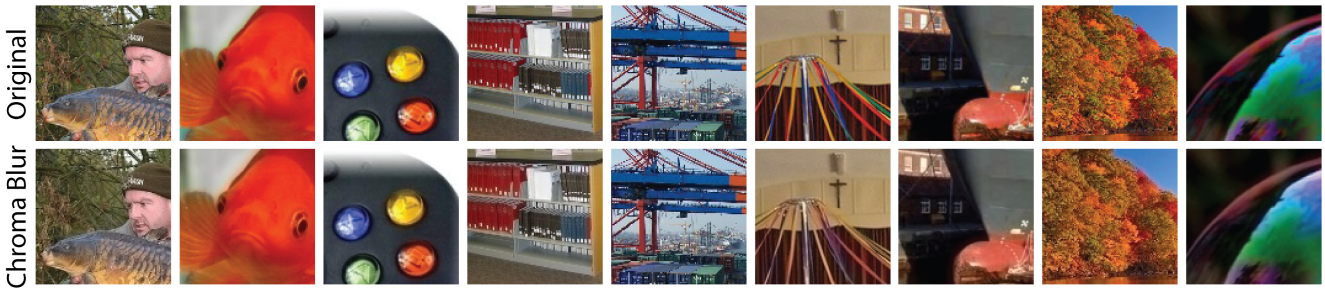

Figure 1. These are examples of patches taken from ImageNet that are used during self-supervised training. Below each original is an example with chroma blurring. It is frequently difficult to distinguish the blurred from original images, because humans are not very spatially sensitive to variation of color. Chroma blurring can sometimes result in a loss of color saturation and color bleeding of very saturated regions (such as the red ship bow). Notice the original gold fish image has signs of strong chromatic aberration (top-left of head). This is blended out effectively by chroma blurring which switches the green aberration to the fishs own red color.

3.1. Our Toolbox of Methods

3.1.1 Chroma Blurring (CB)

We address the problem of chromatic aberration by removing cues about image aberration while allowing color patterns and opponents to be at least partially preserved. We noted that the human visual system is not very sensitive to spatial frequency of chroma, but is much better at discerning detail about shifts in intensity [27]. However, even with this lack of spatial acuity for color, we can still discern meaningful color patterns. As such, we balance the tradeoff between decreasing color spatial information and removing chromatic aberration cues.

To preserve intensity information, but reduce aberration, we start by converting images into *Lab* color space. We then apply a blurring operation to the chroma *a* and *b* channels. In this case, we use a 13x13 box filter. It is two times the size of the 7x7 convolution filter of our original GoogLeNet target network. The luminance channel is left untouched, and we convert the image back to RGB. Figure 1 shows several patches which we have *chroma blurred* for comparison.

3.1.2 Yoked Jitter of Patches (YJ)

Most patch/context methods apply a random jitter between patches [10, 31, 11, 25]. The different patches are jittered in different amounts and different directions. One issue with applying a random jitter is that it might distort or skew the spatial understanding in the network. As an example, if the head of an animal is observed in one patch and the feet in the other, the true spatial extent between these items would be difficult to discern given a random jitter: the patches might be 30 pixels apart, or they might be 60. If on the other hand, the patches maintained a fixed spacing, reasoning about the extent of an object between patches would be easier. Thus, the network might make better inferences about the larger shape of an object beyond each patch itself.

We do this by yoking the patch jitter. Each patch is randomly jittered to create a random crop effect, but they are jittered by the same amount in the same direction. This might make us prone to trivial low-level boundary pattern completion, but as mentioned, we suspect this can be partially addressed by having non-4-connect patches. Also, it is unclear how well a random jitter will address such a problem since features do not need to be aligned in order to be recognized in a CNN. Additionally, a certain amount of low-level boundary pattern completion may not be a problem since it may enhance learning of simple features.

3.1.3 Additional Patches and Configurations

We use the same 96x96 sized patches as [10], since it fits the receptive field (Our use of the term is somewhat liberal here) of a 3x3 convolution at the top of many CNNs. That is, most of the popular networks have a five-layer topology with each layer being half the dimension of the preceding one [39, 17, 18, 23, 30] (AlexNet technically has four layers, but the first layer has half-scale of most other networks). There is some dimensional variation caused by the omission of padding in some layers, but one pixel on the top layer maps to an extent of about 32 pixels in the input image. Since most networks tend to only use 3x3 convolutions at the top layer, a patch size of 96x96 is justified to cover its full field.

We use three patches **(TP)**. In the two-patch configuration of [10], one of the two patches may not cover an area with useful information. For instance, half of the image may be covered by ocean. We can address this problem by using more patches. [31] uses a "puzzle" system of nine 80x80 patches. However, if we want to use 96x96 sized patches and be able to train larger, more contemporary networks, this may not be feasible to train properly. If, on the other hand, we just add one more patch, we create a triple Siamese network which is smaller than a single network over the traditional 224x224 image size. This makes

it easy to move to a larger network. We did no test four patches.

We add extra patches configurations **(EPC)**. That is, we added several new configurations of patches seen in Figure 2. Adding new patterns (1) creates more orthogonal and unique patterns (2) covers more of the image at once (3) mixes scales to prevent simple pattern matching between scales (4) creates a natural way to cover the image, but use multiple scales for training.

We draw three different patterns of patches. We start by extracting all patches at a 110x110 resolution. 3x3 patches are taken from a 384x384 image. This is taken from the center of an aspect preserved image by reducing the smallest dimension to that size. The patches are evenly spaced and aligned with the image corners to cover the image (there is no edge margin). 2x2 patches are taken from a 256x256 image and overlap patches are taken from a 196x196 image. We then use eight 3x3 patterns similar to [10], 2x2 patterns are L-shaped and we use four of them. We combine patches from 3x3 and overlap to create the hybrid patches. These are hybrid scale patches which allow more semantic reasoning by preventing easy matching of simple features between patches since the patches are at different scales. The processed patches are fed into the network in batches. The final patches fed into the network are sized 96x96. Note that Figure 2 shows all 20 typical combinations from this sample image. These are the exact same patterns extracted from all images in the ImageNet training set. In all, we obtain 25,623,340 training patch sets from ImageNet.

3.1.4 Random Aperture of Patches (RA)

We mentioned some evidence that middle layers in a self-supervised network are being neglected. One approach would be to try and create a bias towards these neurons. In the general five-layer topology, the fourth layer has a receptive field of 48x48 given a 3x3 filter. In AlexNet, this would include layers conv3, conv4 and conv5. In GoogLeNet, this would include all 4th layers (4a, 4b etc). If we create an aperture, we could create a patch that doesnt cover the extent of the 5th topological layer, but does cover the extent of the 4th topological layers filters. This could bias against the 5th layer from learning since it cannot see the whole patch. Ideally, this would put emphasis on learning in the 4th layer. See Figure 3 for an example of this.

A random aperture on two of the three patches is created. The idea behind leaving one patch un-apertured is so that we dont completely bias against topological layer 5, we still want it to learn. The aperture is square and for each sample is randomly sized between 64x64 and 96x96. The minimum size is 64 since this is the smallest size we can use and guarantee that at least one 3x3 convolution is unobstructed in the fourth layer. The position of the aperture is also randomized but must fit inside the patch so we can never have a viewable area less than 64x64. The area outside the aperture is filled with ImageNet mean RGB. The size and position of the aperture in two patches is yoked. Which two patches are apertured is randomized for each sample.

3.1.5 Rotation with Classification (RWC)

Each patch in a patch/context model may simply contain a part of a much larger object. In general, this is the intent of the patch/context approach. Might it help if parts can be understood at different orientations? For instance, if one has seen in upside-down roof top, one may better understand a triangular yield sign or a funnel. Additionally, humans have the ability to *conditionally* recognize upside-down parts embedded in a whole image. This is illustrated by the famous Thatcher illusion [40] (see Figure 4). We reason that self-supervised learning might benefit from exposure to upside patches, and it would help to make the network identify if patches are right-side-up or upside-down. We do this by flipping the whole image so that all patches are flipped. Then, we double the number of classes by giving each upside-down image its own class. For instance, if we have 20 classes of patch arrangements, when we add upside-down images, we have 40 classes. We also explore 90 and 270 degree rotations. This yields a total of 80 classes.

Forcing the network to classify patches as upside-down also reduces the strength of clues generated by chromatic aberration. Aberration radiates from the center of the image. Without rotating the image, a downward sloping arch of green/magenta to the left indicates the patch comes from the upper left-hand corner. However, in a flipped image, the same pattern indicates the lower right corner instead. By just trying to guess upper left-hand corner from the chromatic aberration pattern, it will be wrong 50% of the time. With four rotations, it will be wrong 75% of the time.

3.1.6 Miscellany

We present experiments with a few other tricks which we found helpful in varying degrees. One method is a typical mixture of label preserving transformations [37, 6, 5] we are calling the *usual bag of tricks* **(UBT)**. This involves augmentation by randomly mirroring, zooming, and cropping images. The mirroring is simple horizontal flipping and has no special classification (Like RWC) since this would most likely prove confusing to the network. For random zooming, we randomly scaled each input 110x110 patch to between 96x96 to 128x128, and then extract a random 96x96 patch from this. The zoom and crop location is random for each sample, but is yoked between the three input patches.

Figure 2. The left column shows the location patches are extracted from in the image. The next column over shows some example configurations obtained from those patches. In the middle are all 20 patches extracted from this (and every) image. The right column shows how these patches are fed into the process to then create the final patches fed into the neural network.

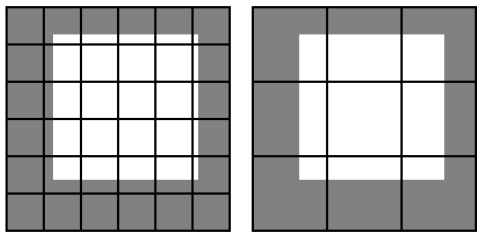

Figure 3. The grayed-out area has been apertured on a 96x96 size patch. The aperture is 64x64, the smallest size we use. The left image shows the pixel arrangement on topological layer four. At least one 3x3 region is not directly interfered with by the aperture. However, on the right, layer 5, only one pixel is fully uncovered. All spatial interactions at this layer will involve at least one occluded region. Ideally, this would create some inhibition to layer five forming meaningful spatial associations and perhaps bias towards layer 4 which can. Note that this description is simplified and somewhat imprecise since image information can propagate laterally through consecutive layers.

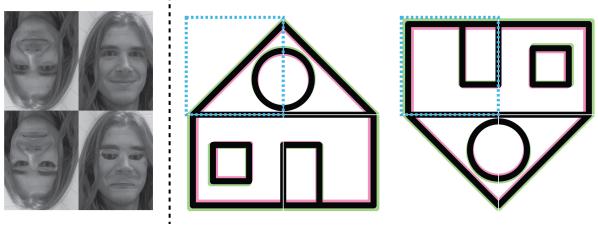

Figure 4. On the left is an example of the famous thatcher illusion [40, 9]. It demonstrates conditional sensitivity to upside-down features in an image against the background. We used this mostly as inspiration. On the left house image [41], the network can tell that the blue bordered area comes from the upper left corner based on chromatic aberration alone. However, on the right image, rotation with classification makes it tell us if the patch is inverted and comes from the lower right corner. If it uses chromatic aberration as the only cue, it would be wrong 50% of the time. (Enlarged in appendix [[[LABEL]]])

Borrowed from [33], we take the idea of mixing the method of rescaling. Each of the three patches is rescaled by one of four randomly chosen rescale techniques (Bilinear, Area, Bicubic or Lanczos). The random selection is *not* yoked between patches. The idea is yet again to make it harder to match low level statistics between patches. We call this randomization of rescaling methods **(RRM)**.

We also tried varying the learning rate and decay rate of network layers to increase learning of middle layer weights. For instance, one can adjust the first layer to have 70% the learning rate as the center most layer. We try to linearly increase the learning rate towards a middle layer and then reduce the rate back down. Given the nine layers of a Siamese AlexNet, we would have learning rates $\{0.7, 0.8, 0.9, 1.0, 0.9, 0.8, 0.7, 0.6, 0.5\}$. Here, conv4 layer has learning rate multiplier 1.0 and the last fully connected layer has 0.5. We call this weight varying **(WV)**.

3.2. Verification Datasets

The development and testing of new techniques generally requires fishing for results. As such, one should avoid using the target dataset for testing each new idea. Fishing

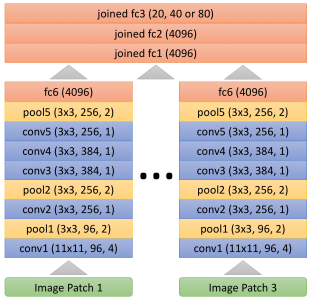

Figure 5. This is our custom batch normalized triple Siamese AlexNet. It is very similar to [10]. Each layer has a batch norm layer after it. Notice we have removed LRN [23] layers.

leads to solutions specialized towards the specific dataset rather than a general solution. For self-supervised learning, test metrics based on PASCAL VOC [14], ImageNet [8] and CSAIL Places [48] are commonly used. Therefore, we test techniques on a few new datasets with a certain amount of overlap, but which possess differences so that we can be more confident in generalization.

For validation, we use a combination of CUB Birds [44] (a fine-grained bird species dataset) and CompCars [45] (a fine-grained car model dataset). We call this combination CUB/CCARS. We use these sets by training a network in a self-supervised manner and then apply transfer learning by fine-tuning them for classification. Both data sets are fine grained for their respective class (birds and cars). However, there are major differences between the kinds of features cars and birds have. Additionally, the CUB Birds dataset provides an ideal test set for dealing with chromatic aberration. The four keys to identifying birds are size/shape, habitat, behavior and color pattern [7]. When trying to control chromatic aberration, one may alter the image in a way that negatively affects color processing and thus classification for birds.

3.3. Alternative Networks

We are interested in generalization of our solutions and portability to other architectures. If we constrain self-supervised learning to mostly being a training protocol, it would be easier to train on different networks. As with using many datasets, using many networks also helps to assure that a technique is not network specific, but works well on other designs. We demonstrate results on four different networks. These are (a) standard CaffeNet type AlexNet [23, 21] (b) AlexNet with *batch normalization* **(BN)** [19] (figure 5) (c) a ResCeption network [30] (d) an Inception network with BN [4, 19].

4. Experiments

We perform a variety of experiments. We show the ablation gain of each tool on our CUB/CCARS dataset combination, and also *post hoc* on VOC classification.

4.1. Self-supervised Training

We use triple Siamese networks which share weights between branches, and are then concatenated together and run through a few fully connected layers. The input is a set of three 96x96 RGB image patches processed from 110x110 patches, taken from the ImageNet training dataset. Recall that we apply the chroma blur operation offline before we train to remove the expense of repeated Lab conversion and blurring. The output is a softmax classification for the patch arrangement pattern class with 20, 40 or 80 classes. All networks load in a list of shuffled training and testing patches. The list is reshuffled after each epoch.

We use a slightly different protocol for training the batch normalized networks than for training the non-normalized CaffeNet. The batch normalized networks train with stochastic gradient descent (SGD) for 750k iterations with a batch size of 128 and an initial learning rate of 0.01. A step protocol is used with a size of 300k and gamma 0.1. Momentum is 0.9 and weight decay is 0.0002. For our CaffeNet, we use a Google exponential style training [4]. We train for 1.5 million iterations with a batch size of 128 and initial learning rate of 0.00666 (the fastest rate seemingly stable). We train SGD with a step size of 10k and a gamma of 0.96806. Momentum is 0.9 and weight decay is 0.0002.

4.2. Validation and Ablation on CUB Birds and CompCars

The bulk of our testing and validation was carried out by fine-tuning a self-supervise trained network to the CUB/CCARS datasets. Both sets were split *a priori* into training and testing sets by the authors. We use provided bounding boxes from both sets to pre-crop the images. Further details can be seen in the appendix 5.

We perform most ablation and validation experiments on our custom batch normalized AlexNet which can be seen in figure 5. The target network is similar to [10] in that we use the same conv6 and conv6b layers, but we do not try to transfer the fully connected layers from the self-supervise trained network. We kept these layers mostly for diversity, so that our batch normalized AlexNet is somewhat different from the very standard CaffeNet/AlexNet we perform benchmark tests on. Again, generalization is important to us. We self-supervise train, then transfer the weights of the five convolution and batch norm layers to the non-Siamese network and initialize new fully connected layers. Both CUB and CCars are trained the same way. The methods for training both had been established *a priori* to avoid overtuning of hyperparameters. For fine-tuning, we use a polynomial learning policy with an initial learning rate of 0.01 with SGD for 100k iterations with a batch size of 64. Polynomial power is 0.5. Momentum is 0.9 and weight decay

	Accuracy					Improvement				
Method	**CUB**	**CCars**	**Mean**	**VOC**	**All**	**CUB**	**CCars**	**Mean**	**VOC**	**All**
No Pretrain	56.20	59.13	57.67	55.12	56.82	–	–	–	–	–
ImageNet Supervised	74.44	85.40	79.92	72.67†	77.50	–	–	–	–	–
Baseline 2 Patch Protocol	62.33	79.86	71.09	63.61	68.60	–	–	–	–	–
add Chroma Blurring **(CB)**	64.29	80.80	72.55	64.98	70.02	1.97	0.94	1.45	1.37	1.42
add Yoked Jitter **(YJ)**	65.17	80.95	73.06	65.15	70.42	0.87	0.15	0.51	0.16	0.40
add 3rd Patch **(TP)**	65.19	81.54	73.36	65.27	70.66	0.02	0.59	0.30	0.12	0.24
add Extra Patch Cfgs. **(EPC)**	67.07	80.50	73.79	65.67	71.08	1.89	-1.04	0.43	0.41	0.42
add Usual Tricks **(UBT)**	67.91	80.83	74.37	65.58	71.44	0.84	0.33	0.58	-0.10	0.35
add Rand. Aperture **(RA)**	68.01	82.07	75.04	66.79	72.29	0.10	1.24	0.67	1.21	0.85
add Rotation 180 **(RWC)**	68.89	84.23	76.56	68.39	73.83	0.88	2.16	1.52	1.60	1.55
add Rotation 90, 270 and WV	69.39	84.25	76.82	68.31	73.99	0.50	0.02	0.26	-0.07	0.15

Table 1. This is a basic ablation showing the effect of adding each method one at a time. The scores for CUB Birds (CUB) and CompCars (CCARS) are the single class classification accuracy. PASCAL VOC uses mean average precision (mAP). The mean column is for CUB and CCARS, but we show a mean of CUB, CCARS and VOC as "All". VOC is the *Post Hoc* classification results run after the fact to see how well our CUB/CCARS surrogate set matches a core self-supervised benchmark test. The baseline two patch protocol uses color dropping and matches the protocol of [10]. Gains in CUB/CCARS appear to correlate with gains in VOC (but not perfectly). The largest gains for both CUB/CCARS and VOC are from rotation with classification, chroma blurring and random aperture. Also notice that the results for CompCars is only one percentage point less than the ImageNet pretrained network. †The ImageNet pretrain for VOC uses conv1 through conv5. All fully connected (fc) layers are initialized new.

is 0.0002. For each condition we wished to test, we trained three times and took the average testing accuracy to reduce minor variation within condition results.

Ablation results for each method can be seen in Table 4.1. We show *post hoc* results from PASCAL VOC 2007 classification on the same network and condition to see how well our validation set results map to one of our target data sets. VOC was trained by the standard classification method described in [22] and results are in mean average precision (mAP). Finer details on ablation can be seen in the appendix 6.

4.3. Standard Transfer Learning Testing Battery

We demonstrate how the results we have obtained compared with self-supervised methods using a suite of standard benchmark tests. These include classification [22] and detection [15] on PASCAL VOC 2007, and segmentation [28, 36] on PASCAL VOC 2012. They also include the "linear classifier" tests on CSAIL places and ImageNet [46]. We note two possible differences from the standard benchmark methodology here. For detection, we use multiscale training and testing. This is common and used by [35, 31, 32, 10, 11], but not all authors use it. For segmentation, we take our self-supervised network and train out it on PASCAL VOC 2007 classification. This is the network we obtain classification results on. We then transfer that network over using the standard network surgery to train segmentation per usual [36]. This allows us to copy fc6 and fc7 to the segmentation network. There is some overlap between VOC 2007 training and the VOC 2012 derived validation set used to test segmentation from [28]. However, removing the overlap actually *improves* results slightly. As such, we use the standard VOC 2012 segmentation validation set used in [28] and take the lesser of the two results.

For these tests, we self-supervise train a triple Siamese CaffeNet type AlexNet using the non-batch normalized protocol previously described. After pooling layer 5, we use the same Siamese structure as our custom batch normalized AlexNet. We leave out batch normalization in these layers, but insert dropout layers after joined_fc1 and joined_fc2 with a dropout ratio of 0.5. Convolution weights from layers one through five are transferred to a completely off the shelf CaffeNet. Training and testing are performed in the standard way defined by the authors of each test (with the two noted differences). Results can be seen in Table 4.3, 4.3 and 4.3. Our improvements yield results that out-perform all other methods on all of the standard benchmark tests.

4.4. Portability to Other Networks

We trained on two more networks to demonstrate portability and generalization. The first new network ResCeption [30] is a GoogLeNet [39] like network with batch normalization (BN) [19] and residual short-cutting [17]. It has 5x5 convolutions in layer 5 which extend beyond the self-supervised receptive layer. So we self-supervise trained by replacing these with 1x1 surrogate filters that cannot train. Then we put freshly initialized 5x5 convolutions back in place for this layer when we fine-tuned.

We also used a standard inception network with BN. The ImageNet pre-train was performed by [3] on the full set of 21k ImageNet labels. The network required no augmentation. All weights are copied from self-supervised training

Method	Class.	Det.	C + D	Seg.	All
ImageNet Labels [8, 23]	79.9	56.8	68.4	48.0	61.6
Jayaraman [20]	–	41.7	–	–	–
Li [26]	56.6	–	–	–	–
Misra [29]	–	42.4	–	–	–
Owens [34]†	61.3	–	–	–	–
Larsson [24]	65.9	–	–	38.4	–
Agrawal [1]†	54.2	43.9	49.1	–	–
Gomez [16]	55.7	43.0	49.4	–	–
Pathak (Inpainting) [36]	56.5	44.5	50.5	29.7	43.6
Donahue [12]†	60.1	46.9	53.5	35.2	47.4
Wang (Video) [42]†	63.1	47.4	55.3	–	–
Lee [25]	63.8	46.9	55.4	–	–
Zhang (Colorizing) [46]†	65.9	46.9	56.4	35.6	49.5
Pathak (Move) [35]	61.0	52.2	56.6	–	–
Zhang (Split-Brain) [47]	67.1	46.7	56.9	36.0	49.9
Bojanowski [2]	65.3	49.4	57.4	–	–
Doersch (Patches) [10]†	65.3	51.1	58.2	–	–
Noroozi (Counting) [33]	67.7	51.4	59.6	36.6	51.9
Noroozi (Puzzle) [32, 31]	67.6	53.2	60.4	37.6	52.8
Doersch (Ensamble)* [11]	–	54.9	–	–	–
Wang (Invariance)* [43]	–	53.1	–	–	–
RWC 180	69.0	54.6	61.8	**40.2**	54.6
add rotations 90, 270	68.7	**55.2**	61.9	**40.2**	54.7
add RRM	**69.3**	55.0	**62.1**	40.0	**54.8**

Table 2. These are classification mAP [22], detection mAP [15] and segmentation mIU [28] test results over PASCAL VOC [14]. Mean scores are shown for classification + detection *(C + D)* as well as for all three if the segmentation score is available. CB + YJ + TP + EPC + UBT + RA + RWC (four rotations) + RRM gives the highest score when all three tests are averaged, but the conditions without RRM give the best scores on detection and segmentation. †To conserve space, we have taken the largest of two scores when network weights have been rescaled [22]. *Denotes that this is an estimate for the score based on a very recent result with a different network other than AlexNet. The estimate is computed by adding the gain reported in the work to a mutual baseline method that has an AlexNet result and also appears in our table (namely [10]).

except for the very top full connected layer which would be discarded anyway. Table 4.4 shows the results from these new networks with our BN AlexNet. CompCars results tend to be within about one to two percentage point to ImageNet supervised training. However, CUB runs from three to six. The results from self-supervision seem enticingly close to ImageNet supervised, but are not yet there. Further efforts will be needed to bridge that gap.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015. 8

[2] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. In *ICML*, 2017. 8

[3] P. Campr and M. Li. Imagenet-21k-inception. https://github.com/dmlc/ mxnet-model-gallery/blob/ master/imagenet-21k-inception.md. 7

[4] V. V. Christian Szegedy, Sergey Ioffe. Inception-v4, inception-resnet and the impact of residual connections on learning. In *arXiv:1602.07261*, 2016. 6

[5] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *CoRR*, 2012. 4

Method	C1	C2	C3	C4	C5	Best
ImageNet [8, 23, 46]	19.3	36.3	44.2	48.3	50.5	50.5
Random [33]	11.6	17.1	16.9	16.3	14.1	17.1
Pathak [36]	14.1	20.7	21.0	19.8	15.5	21.0
Donahue [12]	17.7	24.5	31.0	29.9	28.0	31.0
Doersch [10]	16.2	23.3	30.2	31.7	29.6	31.7
Zhang (Colorizing) [46]	13.1	24.8	31.0	32.6	32.6	32.6
Noroozi (Puzzle) [32, 31]	18.2	28.8	34.0	33.9	27.1	34.0
Noroozi (Counting) [33]	18.0	30.6	34.3	32.5	25.7	34.3
Zhang (Split-Brain) [47]	17.7	29.3	35.4	35.2	32.8	35.4
RWC 180	**19.6**	31.4	36.1	36.2	31.5	36.2
add rotations 90, 270	19.5	31.4	37.0	**37.8**	**33.3**	**37.8**
add RRM	19.3	**31.4**	36.7	37.0	32.3	37.0

Table 3. This is the linear test for ImageNet data [8]. The network is fine-tuned up to the convolution layer shown on ImageNet. Out results are the bottom three rows. These use all the methods we have presented except for WV. The maximum score is shown in bold with the previous best result underlined. Rotation with Classification using 90, 180 and 270 degree rotations is generally the best performer. For conv2, RRM edges out the other two by a small margin of 0.05.

Method	C1	C2	C3	C4	C5	Best
Places [48, 46]	22.1	35.1	40.2	43.3	44.6	44.6
ImageNet [8, 23, 46]	22.7	34.8	38.4	39.4	38.7	39.4
Random [33]	15.7	20.3	19.8	19.1	17.5	20.3
Pathak [36]	18.2	23.2	23.4	21.9	18.4	23.4
Wang (Video) [42]	20.1	28.5	29.9	29.7	27.9	29.9
Zhang (Colorizing) [46]	16.0	25.7	29.6	30.3	29.7	30.3
Donahue [12]	22.0	28.7	31.8	31.3	29.7	31.8
Owens [34]	19.9	29.3	32.1	28.8	29.8	32.1
Doersch [10]	19.7	26.7	31.9	32.7	30.9	32.7
Zhang (Split-Brain) [47]	21.3	30.7	34.0	34.1	32.5	34.1
Noroozi (Puzzle) [32, 31]	23.0	31.9	35.0	34.2	29.3	35.0
Noroozi (Counting) [33]	23.3	33.9	36.3	34.7	29.6	36.3
RWC 180	**23.4**	33.7	36.6	36.2	33.3	36.6
add rotations 90, 270	23.0	33.9	37.2	**37.3**	**34.7**	**37.3**
add RRM	**23.4**	**34.0**	**37.3**	37.1	34.4	**37.3**

Table 4. This is the linear test for CSAIL Places [48] data. The network is fine-tuned up to the convolution layer shown. Our results are the bottom three rows. These all use the methods we have presented except for WV. The maximum score is shown in bold with the previous best underlined. Adding randomization of the rescaling method improves lower layer results but falls slightly behind the condition which does not use it in higher layers.

Method	AlexNet BN	ResCeption	Inception 21k
ImageNet Pretrain	79.92	88.62	89.01
add Rotations 180 (RWC)	76.56	86.37	85.20
add rotations 90, 270 (RWC)	76.82	86.52	85.27
Diff from ImageNet	3.10	2.10	3.74

Table 5. These are the mean results for CUB and CompCars on the different networks. More detail in appendix [[[LABEL]]]

[6] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification. In *CoRR*, 2011. 4

[7] Four keys to identifying birds. *Bird Scope*, 23(2), 2009. 1, 6

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 6, 8

[9] A. Dodge. Iain mteffect: Wikimedia public domain image. https://commons.wikimedia.org/wiki/File:Iain_MTeffect.jpg. 5

[10] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1, 2, 3, 4, 6, 7, 8

[11] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. In *ICCV*, 2017. 1, 2, 3, 7, 8

[12] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017. 8

[13] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, 2015. 1

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 1, 6, 8

[15] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2, 7, 8

[16] L. Gomez, Y. Patel, M. R. nol, D. Karatzas, and C. V. Jawahar. Self-supervised learning of visual features through embedding images into text topic spaces. In *CVPR*, 2017. 8

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *arXiv preprint arXiv:1512.03385*, 2015. 2, 3, 7

[18] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 3

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6, 7

[20] D. Jayaraman and K. Grauman. Learning image representation tied to ego-motion. In *ICCV*, 2015. 8

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6

[22] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016. 2, 7, 8

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2013. 1, 2, 3, 6, 8

[24] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017. 1, 8

[25] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *ICCV*, 2017. 1, 2, 3, 8

[26] D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang. Unsupervised visual representation learning by graph-based consistent constraints supplementary material. In *ECCV*, 2016. 8

[27] M. Livingstone. *The First Stages of Processing Color and Luminance: Where and What.*, page 4667. Harry N. Abrams, New York, 2002. 3

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 7, 8

[29] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 8

[30] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *14th European Conference on Computer Vision (ECCV)*, Amsterdam, October 2016. 3, 6, 7

[31] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 1, 2, 3, 7, 8

[32] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246, 2016. 2, 7, 8

[33] M. Noroozi, H. Pirsiavash, and P. Favaro. Representation learning by learning to count. In *ICCV*, 2017. 1, 2, 5, 8, 12

[34] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual-learning. In *ECCV*, 2016. 8

[35] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017. 1, 7, 8

[36] D. Pathak, P. K. hl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 1, 2, 7, 8

[37] P. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003. 4

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scaleimage recognition. In *CoRR*, 2014. 2

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2, 3, 7

[40] P. Thompson. Margaret thatcher: a new illusion. *Perception*, 9(4):483–484, 1980. 4, 5

[41] Umbert. Kolora aberacio: Wikimedia public domain image. https://commons.wikimedia.org/wiki/File:Kolora_aberacio,_transversa_%C5%9Dovo,_1.svg. 5

[42] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 1, 8

[43] X. Wang, K. He, and A. Gupta. Transitive invariance for self-supervised visual representation learning. In *ICCV*, 2017. 1, 2, 8

[44] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical

Report CNS-TR-2010-001, California Institute of Technology, 2010. 1, 6

[45] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015. 6

[46] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016. 1, 2, 7, 8

[47] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017. 1, 8

[48] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 6, 8

Method	CUB	CCars	Mean	Improvement
CB	64.29	80.80	72.55	–
CB + YJ	65.17	80.95	73.06	0.51
CB + TP + EPC	65.21	80.17	72.69	–
CB + TP + EPC + YJ	67.07	80.50	73.79	1.10

Table 6. We get a general improvement from using a yoked jitter over a random jitter. When we then include the extra patch configurations, the improvement grows. The hybrid patches intrinsically may prevent low-level trivial boundary completion since they have mixed scales.

5. Appendix A: CompCars Dataset Augmentation

We further augment the CompCars dataset by creating rotating hue and minor perspective jitter. In prior unpublished experiments, these changes seemed to improve accuracy. We rotate the hue by simply swapping color channels. We do this because we hypothesized that car models have varying color, but they seem to have color styles. For instance, family sedans seem to have conservative low saturated colors while sports cars tend to have hot intense and highly saturated colors. We obtain perspective jitter by randomly perturbing three Euler angles by +/- 0.00286 degrees, then we create a perspective transformation matrix from it. We create 24 augmentations per image, from which 20 have random perspective jitter and 12 have hue rotation. Four images are just a repeat of the un-augmented original. We create these permutations before training since perspective transformation is modestly expensive, but still can be a bottleneck. [[[We will release this dataset variation for use]]] [[[ADD EXAMPLE IMAGES]]]

6. Appendix B: Further Ablation Details

6.1. Yoked Jitter Gets Better with Extra Patches

As mentioned, one of the goals of using hybrid patches was to reduce the ability of the network to learn trivial pattern completion between adjacent patches. As a test, we tried the usage of extra patch configurations with yoked jitter and with random jitter. The results can be seen in Table 6.1. By getting a larger gain for yoked jitter, there is some evidence that extra patch configurations (EPC) may have the effect of reducing trivial pattern completions.

6.2. Do Rotations Need Classification?

We tested to see if just rotating a patch without classification for that rotation was sufficient to improve performance. Table 6.2 shows that if we just rotate the patch, there is almost no difference than without rotation. The classification component might help to sharpen the features of objects by forcing the network to recognize the rotated objects uniquely. Also, as we have discussed, classification may

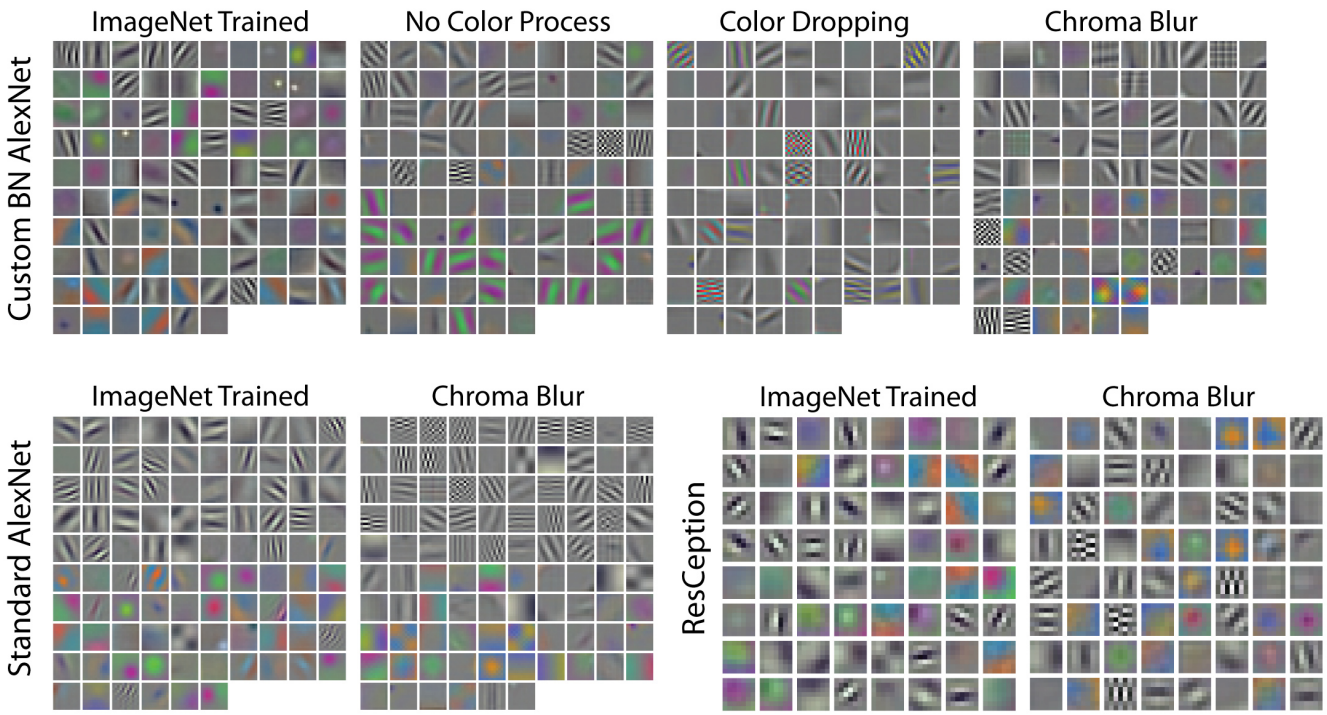

Figure 6. These are the first layer filters from several networks which have either been supervise trained on ImageNet or self-supervise trained. All self-supervised networks are using the full set of methods (but not RRM or WV). Even though rotation with classification may mitigate chromatic aberration, the network still forms filters sensitive to it. Thus, we believe it is only a partial solution. The chroma blur networks are all free of chromatic aberration effects and show formation of healthy color filters (especially when compared to color dropping). As an observation, we can zoom to an effective size of 171x171 during self-supervised training; as a result, we see the presence of finer wavelets compared with ImageNet training.

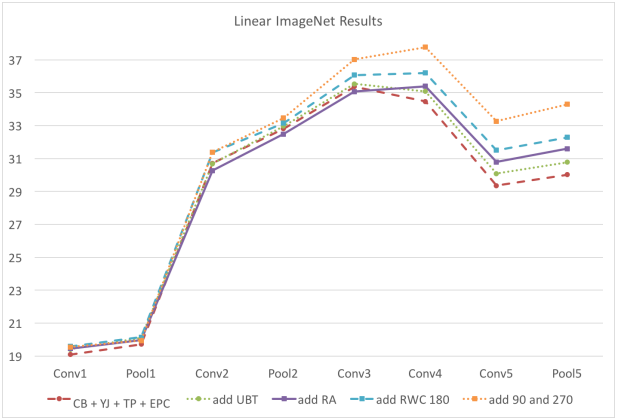

Figure 7. These are linear results for CaffeNet/AlextNet on ImageNet. We can see that when we add Random Aperture (RA), the results seem to switch over to improving layers four and five at the expense of layers one, two and three.

help to mitigate chromatic aberration.

6.3. How much does Chroma Blurring Help?

As figure 6 shows, chroma burring definitely seems to remove any chromatic aberration effects while preserving at least some color feature processing. Looking at Table 6.3, we do get a moderate boost in classification by using chroma blurring compared with no color processing. However, improvement in classification from chroma blurring subsides when all tools are used. We believe that rotation with classification is probably responsible for this.

Method	CUB	CCars	Mean
CB + YJ + TP + EPC + UBT + RA	68.01	82.07	75.04
... + RWC 180 without classification	68.26	81.82	75.04
... + RWC 180 with classification	68.89	84.23	76.56

Table 7. By just rotating the patches but not classifying them, we obtain almost no gain. It appears critical that rotations should have their own class. Note we are using the full toolset except for RRM or WV.

6.4. The Benefit of Adding Different Kinds of Patches

Extra patch configurations definitely seem to help, but their interaction with each other and the other tools is not deterministic. Table 6.4 parses out the contribution of the two types of new configurations we use.

Method	CUB	CCars	Mean	Impr.
YJ	65.04	80.21	72.62	–
... + CD	62.36	79.70	71.03	–
... + CB	65.17	80.95	73.06	0.44
YJ + TP + EPC + UBT + RA + RWC	68.23	83.70	75.96	–
... + CD	65.73	82.94	74.33	–
... + CB	68.42	83.58	76.00	0.04

Table 8. Here we compare color dropping (CD) and chroma blurring (CB) to no color processing. CUB Birds is a pathological case for color dropping since it is very dependent on color patterns for classification. However, and somewhat perplexingly, color dropping does not appear to help CompCars either. Chroma blurring ceases to help once we add in the full set of tools (not including RRM or WV). We suspect this is because rotation with classification at least partially mitigates the effects of chromatic aberration.

Method	CUB	CCars	Mean	Impr.
CB + YJ + TP	65.19	81.54	73.36	–
... + EPC (2x2 Patches)	66.80	81.80	74.30	0.93
... + EPC (Hybrid Patches)	67.32	81.69	74.50	1.14
... + EPC (2x2 *and* Hybrid Patches)	67.07	80.50	73.79	0.43
CB + YJ + TP + UBT + RA + RWC	68.63	82.87	75.75	–
... + EPC (2x2 Patches)	68.29	83.67	75.98	0.23
... + EPC (Hybrid Patches)	67.09	82.58	74.83	-0.92
... + EPC (2x2 *and* Hybrid Patches)	68.89	84.23	76.56	0.81

Table 9. Here we show the effect of the two types of extra patch configurations on their own. Improvement is not straight forward from the addition of the two types. Using both is always better than using just the 3x3 patches. However, when using the full tool set (not including RRM or WV), the hybrid patches by themselves are actually worse. Our hypothesis is that the 2x2 patches help with rotation classification (RWC) since they always include the top and bottom of the image. These are good locations for cues an image is upside-down (sky v. ground). Without that help, the hybrid patches somehow inhibit performance. It is not entirely clear why.

Method	CUB	CCars	Mean
With three apertures	67.76	83.22	75.49
With two apertures	69.04	83.46	76.25

Table 10. If we aperture all three patches, we see a noticeable drop particularly in CUB Birds. We did not test the aperture of one patch.

6.5. Two v. Three Apertures

We chose to only apply the patch aperture to two patches and not all three. The idea was to inhibit the highest levels of the network and instead focus learning on mid-levels. If we applied the aperture to all three patches, we reasoned that we would *always* inhibit the higher levels when we only want to inhibit them *some of the time*. As table 6.5 shows, two apertures are definitely better than three.

We ran some further testing to see if applying the aperture has the effect of improving middle layers. Figure 7 shows that it has a boost on layers four and five which are middle layers in AlexNet. Interesting, it has somewhat degrading effects on layers one and two. This is a kind of behavior we would expect if mid-layers are being biased for.

6.6. RRM has a Similar Pattern of Gain to Counting

RRM has a Similar Pattern of Gain to Self-Supervised Learning from Counting [33], from Which it is Derived. RRM gave us somewhat mixed results. However, it gave us a pick up on tests which the work it is derived from also did best on. RRM seemed to boost our scores on (1) ImageNet Linear Test layer conv2 (2) The VOC classification test (3) conv layers 1, 2, and 3 on the Places Linear Test (but not conv4, the correspondence is not perfect). This may suggest that we got the same kind benefit from using it even though our usage was somewhat different.