Contents

A Definitions

For convenience this section gives an overview of important definitions used in the paper.

- Classifiers are referenced by their first letter where applicable:
  $L :=$ LeNet-5, $A :=$ AlexNet, $V :=$ VGG 16 BN, $I :=$ Inception v3, and $R :=$ ResNet-50.
- $\mathbb{A}_i \circ j$ indicates that the classifier j is using input samples from an AE that has been fine-tuned with gradients provided by classifier i.
- $\mathbb{A}_S$ refers to the SegNet AE pre-trained on YFCC100m and, $\mathbb{A}_i(x)$ refers to a reconstruction of input sample x using $\mathbb{A}_i$.
- $\mathcal{C} = \{L, A, V, I, R\}$ the set of all evaluated classifiers.

(a) Original

(b) Reconstruction by $\mathbb{A}_L$

(c) Original

(d) Reconstruction by $\mathbb{A}_L$

Figure 1: Two examples of images reconstructed by $\mathbb{A}_L$. (a) and (b) show an image where the border regions are not relevant to the classification, while they are for images (c) and (d).

B LeNet-5 Details

Compared to the other classifiers used through this work LeNet-5 [1] is a much earlier incarnation of DCNNs. It is originally only intended for use with the MNIST dataset for digit recognition. We adapt the original architecture definition to work with ImageNet at the same input size of 224 pixels used by the other networks.

Our version uses three convolutional layers with kernel size 5 with 6, 16, and 120 output filters respectively. The 2nd and 3rd layer use a stride of 2. Each is followed by ReLU and 2×2 max pooling. The final activation map after the convolutional part of the network has a spatial resolution of 5×5 pixels, same as the original. The architecture is finalized by two fully connected layers with 1000 neurons each to accommodate the much larger number of classes of ImageNet compared to MNIST.

We train our modified LeNet-5 for 60 epochs on ImageNet using the same set of data augmentation measures as mentioned in paper Section 3.1, an initial learning rate of $\eta = 0.01$ and a polynomial schedule $\eta' = \eta \cdot (1 - \frac{i}{n})^p$ with $p = 0.9$, i the current, and n the total number of training iterations. Final accuracies are 32.30% top-1 and 54.63% top-5.

Note that, as defined above, the final 2×2 max-pooling operation works on an activation map of 11×11 pixels. Pytorch does not perform any implicit padding, so the final output is 5×5 pixels in size. The lowest row and rightmost column are discarded, which means these regions are underrepresented in computed gradients. We see this as artifacts in the reconstructed images of $\mathbb{A}_L$ as seen in Figure 1. Note that, as seen in Figure 1d compared to Figure 1b, these artifacts are not constant and instead appear to be highly dependent on the relevance of the content. E.g., in Figure 1d parts of the tractor are clearly visible compared to the sky behind them.

Table 1: Accuracy when different parts of the combined autoencoder and classifier are trained while others remain fixed. Included is a run where only the decoder is trained starting from a random initialization.

Train			
Encoder	Decoder	Classifier	top-1
Y	N	N	16.36
Y	Y	N	71.04
N	Y	N	74.92
N	Y (random)	N	74.41

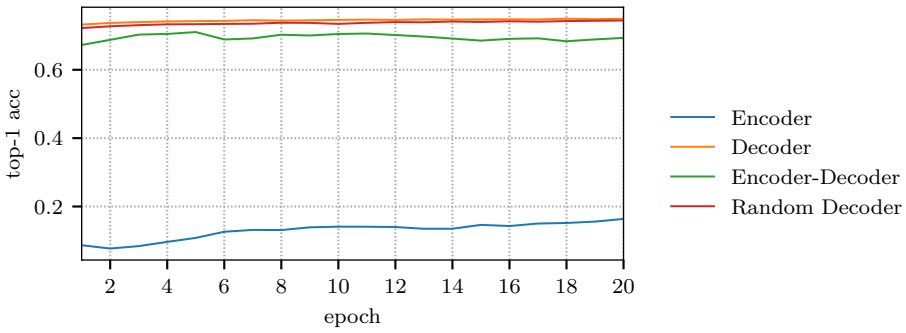


Figure 2: Validation accuracy over epochs for different training methods.

C Autoencoder Training

Issues may arise when we attempt to train combined autoencoder-classifier networks. SegNet on its own has a depth of 32 layers without residual connections to propagate stronger gradients to earlier layers. A classifier potentially adds several dozen more. Depending on the combined architecture vanishing gradients may become an issue. To find a stable training method we study behavior when parameters of different subnetworks are modified and others are fixed.

C.1 Training with Fixed Subnetworks

SegNet combined with ResNet-50 is trained for 20 epochs on ImageNet in the same fashion as previously described in paper Section 3.3. We test the following combinations: Train encoder and decoder, train encoder only, train decoder only.

The classifier remains fixed to make the fine-tuned version a drop-in add-on for existing models. This should encourage some level of interchangeability of fine-tuned autoencoders, since all our classifiers expect image-like inputs. We investigated swapping fine-tuned autoencoders to different classifiers in paper Section 3.4.

(a) Original (b) Pre-Trained (c) Random

Figure 3: Example of reconstruction result produced after fine-tuning from a (b) pre-trained or (c) randomly initialized decoder compared to (a) the original image.

Table 1 shows the validation accuracy of these fine-tuning runs. Training only the decoder proved to be most effective as it improves over standalone accuracy of ResNet-50 after 4 epochs at 74.14% and settles at 74.92% after 20. Training the whole SegNet showed unstable progress with the highest accuracy at epoch 5 of 71.04%, which actually decreases to 69.33% after 20. Encoder-only training proved not to be viable with very slow progress towards a final accuracy of 16.38%. Performance is slightly worse at 74.41% when the decoder is initialized with random weights. We observe that this decoder trained from scratch converges towards recreating outputs that are visually highly similar to the original image as seen in Figure 3. This is noteworthy since there are no hard constraints to force this behavior and previous work has shown that it is possible to create images that are visually unrelated to their class yet elicit strong responses from the classifier [2].

C.2 Autoencoder Fine-Tuning Behavior

Finally, we present validation accuracies on ImageNet during fine-tuning of autoencoder-classifier combinations. Training starts from $\mathbb{A}_S$ and only the parameters of its decoder are adapted as per the methodology outlined in Section C.1. We observe slow but steady improvement for all tested models as seen in Figure 4.

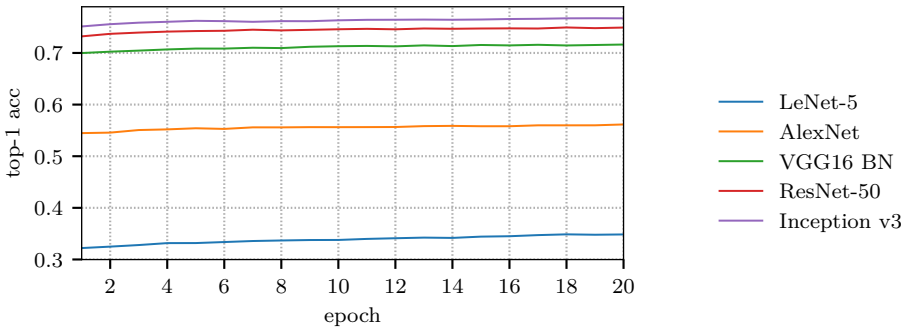


Figure 4: Top-1 accuracy on ImageNet validation set during fine-tuning of combinations of $\mathbb{A}_S$ and named classifiers.

Table 2: RRC values for accuracies reported in the cross-evaluation of autoencoders and classifiers. Values in bold are greater than 0.8. Constructing the formal context with values in bold as positives, yield the lattice where $t = 0.8$.

	L	A	V	I	R
$\mathbb{A}_L$	**1.0000**	**0.8830**	0.1193	**1.2490**	**1.3575**
$\mathbb{A}_A$	0.0606	**1.0000**	0.0174	**0.9575**	0.1647
$\mathbb{A}_V$	**0.8405**	**0.9553**	**1.0000**	**1.0423**	**1.0282**
$\mathbb{A}_I$	0.5250	0.5387	0.6416	**1.0000**	0.6059
$\mathbb{A}_R$	0.0467	**0.8857**	0.0999	**0.9674**	**1.0000**

D Lattices

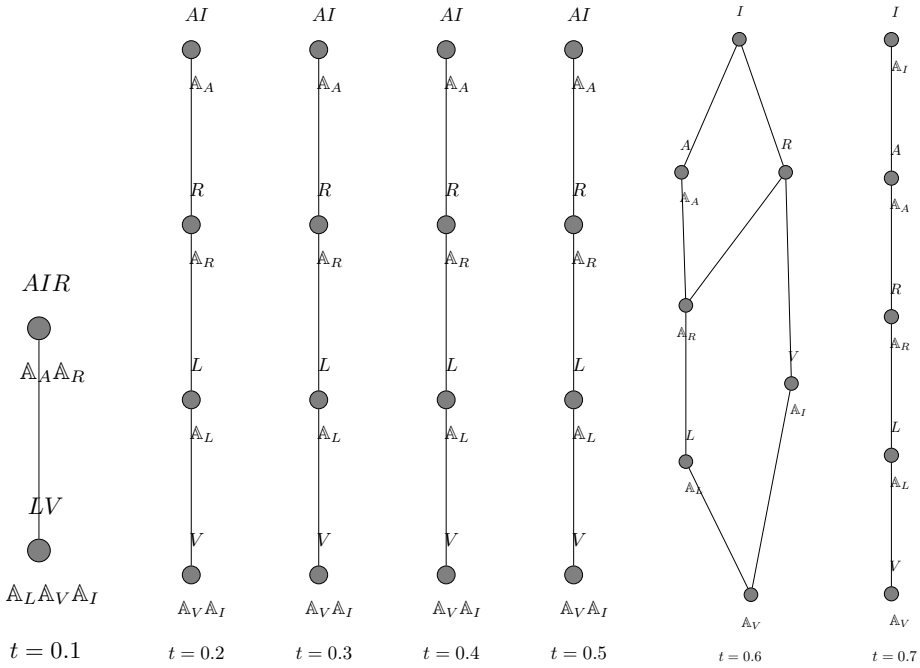
This section presents more lattices corresponding to the formal contexts obtained by thresholding the RRC tables of the cross-evaluation between fine-tuned autoencoders and classifiers. As mentioned in the main paper, the RRC metric is defined as

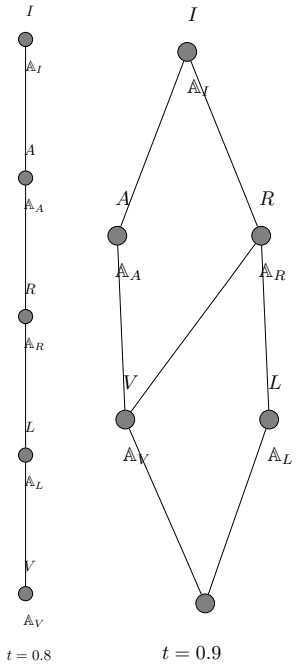
$$RRC(\mathbb{A}_i \circ j) = \frac{acc(\mathbb{A}_i \circ j)}{m(i,j)} \tag{1}$$

where $m(i,j) = min(acc(i), acc(j))$. Given all combinations of $acc(\mathbb{A}_i \circ j)$, the corresponding RRC values for each combination of autoencoder and classifier can be computed (Table 2). To construct the formal concepts of autoencoders and classifiers, we must binarize RRC values by applying a threshold to it. In other words, given all RRC values for any combination of autoencoder-classifier, the binary value for each entry in the table given a threshold is: $binaryRRC(\mathbb{A}_i \circ j, t) = 1$ `if` $t \cdot acc(\mathbb{A}_i \circ j) \geq min(acc(i), acc(j))$. Said threshold t can be interpreted as the minimum relative change in accuracy that is preserved by classifier j when using the signal of $\mathbb{A}_i$. For example, we say that $\mathbb{A}_I$ preserves information that is used by V when $t = 0.6$ but not when $t = 0.8$. FC lattices are a useful tool to discover new relationships in terms of these thresholds as lower and upper bounds for the usefulness of signals (represented by the different fine-tuned autoencoders) for each classifier.

Notice that other normalizations like $acc(i), acc(j), max(acc(i), acc(j))$, may yield different RRC values and thus, produce additional lattices. However, these are prone to misinterpretation since the capacity of the model plays a role on those. We say that the characterization of a model's signal ($\mathbb{A}_i$) is useful to a second classifier (j) if j reaches an portion (t) of the accuracy reached by the lowest performing between i and j. There are two possible reasons for this condition to be false. On one hand, the network may be unfit to process the signal from another model. On the other hand it may be just because the capacity of the model is lower altogether. For example, LeNet cannot reach more than 66% of the performance of AlexNet with original input signals. Hence, it is interesting to look at lower bound cases, where variations in accuracy are measured with respect to the lowest performing model of the pair i, j in $\mathbb{A}_i \circ j$.

D.1 Lattices Normalized by Minimum





E Reconstruction Examples

This section contains more examples of images reconstructed by fine-tuned autoencoders. Compared to the version of this plot in paper Section 3.3.1 these figures are extended with difference images of reconstructions minus original for individual channels in LAB color space, marked as ΔL, ΔA, and ΔB. Positive values mean increased intensity in that channel compared to the original and vice-versa. Leftmost is the difference of original image to itself, i.e., zero for reference.

One interesting property revealed by pixel-wise differences is that the contours of foreground objects present in the image are easily recognizable in the luminance values of $\mathbb{A}_R$ and $\mathbb{A}_I$, and to lesser extent with $\mathbb{A}_A$. $\mathbb{A}_V$ shows very subtle differences if any and remains very neutral overall, which lines up with our previous observation that $\mathbb{A}_V$ can serve as input for all classifiers.

At this point we would like to stress that the presented examples are simply the first 25 images in our random shuffle of the ImageNet validation set. There was no cherry-picking.

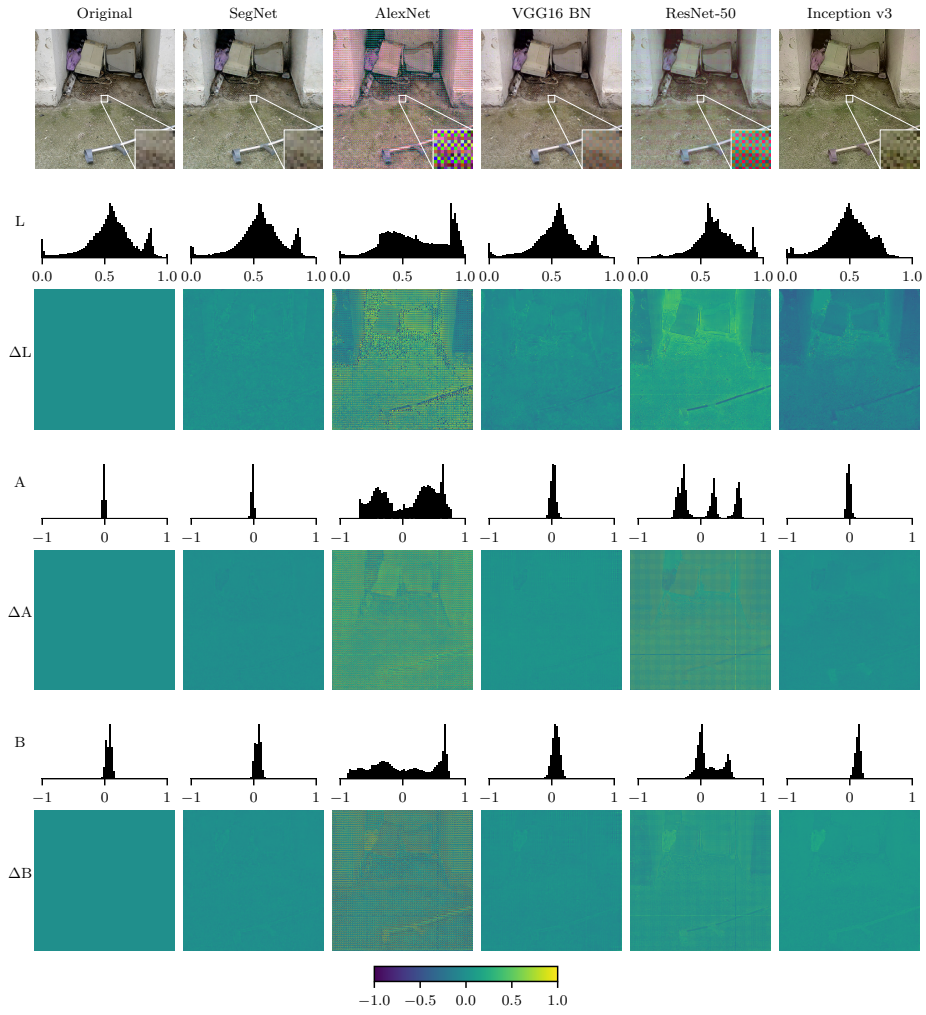


Figure 5: An example of images reconstructed by fine-tined autoencoders.

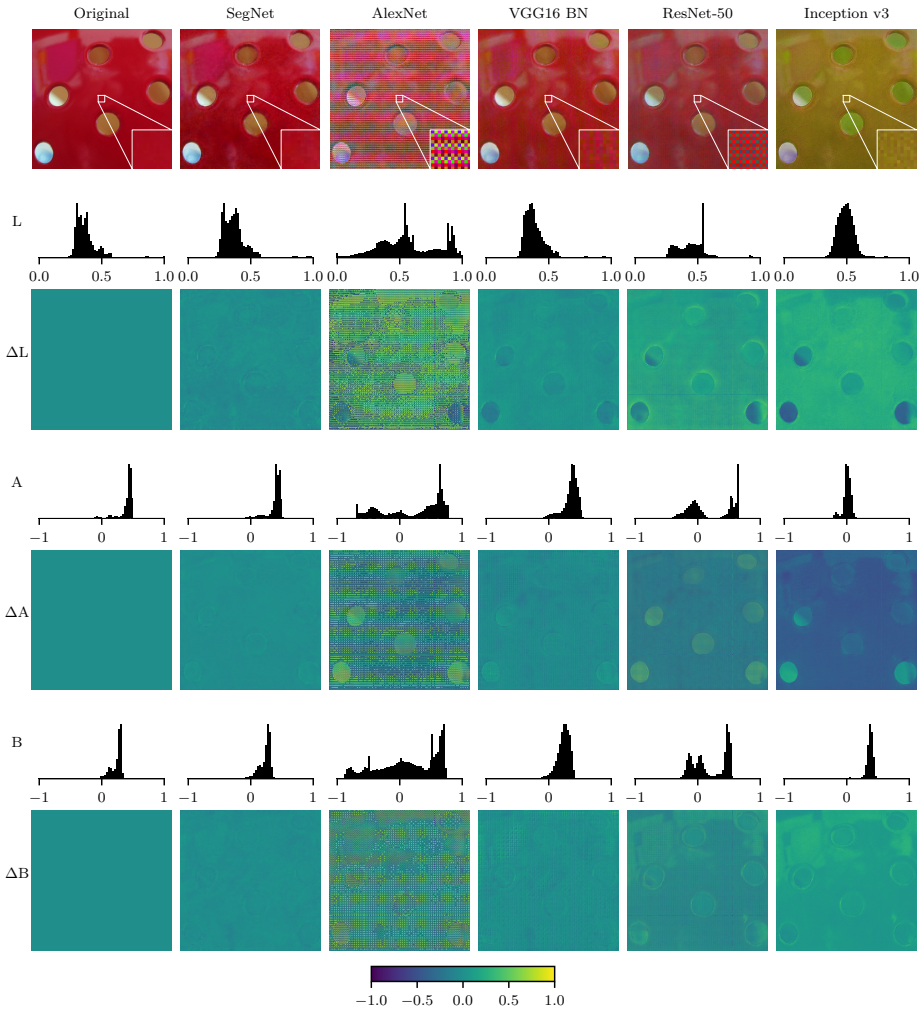


Figure 6: An example of images reconstructed by fine-tined autoencoders.

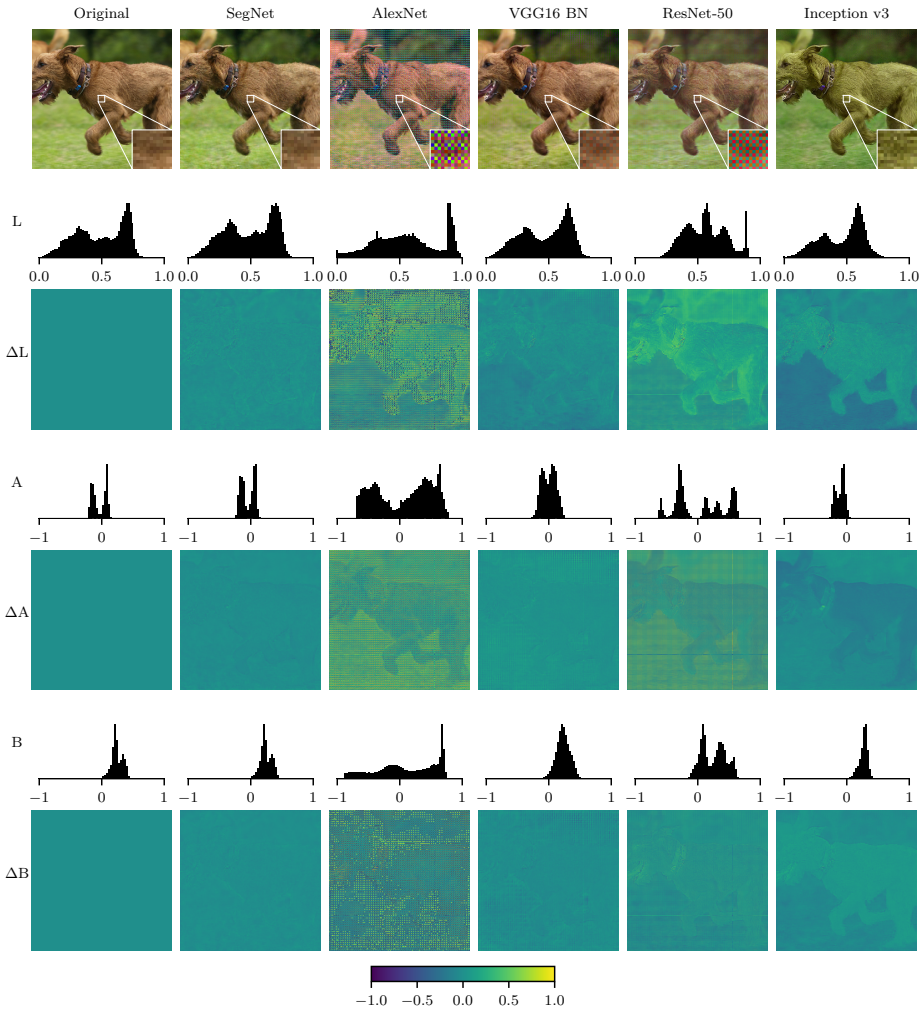


Figure 7: An example of images reconstructed by fine-tined autoencoders.

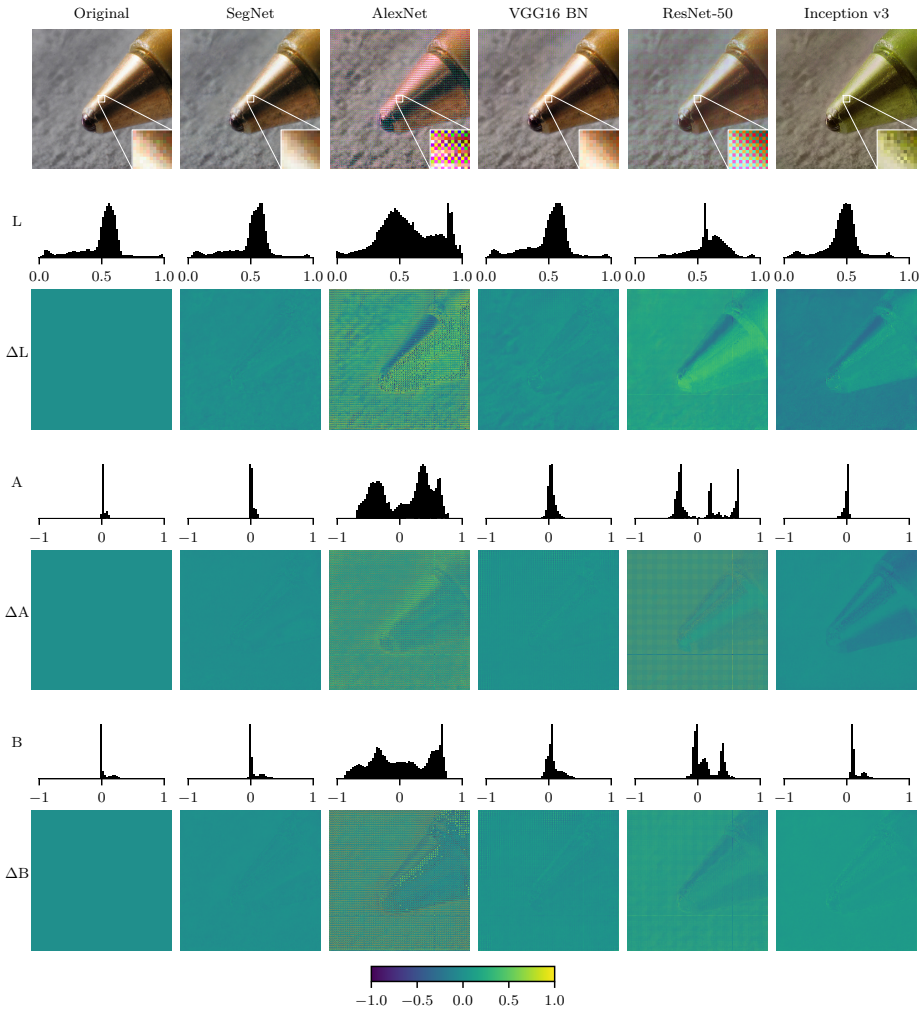


Figure 8: An example of images reconstructed by fine-tined autoencoders.

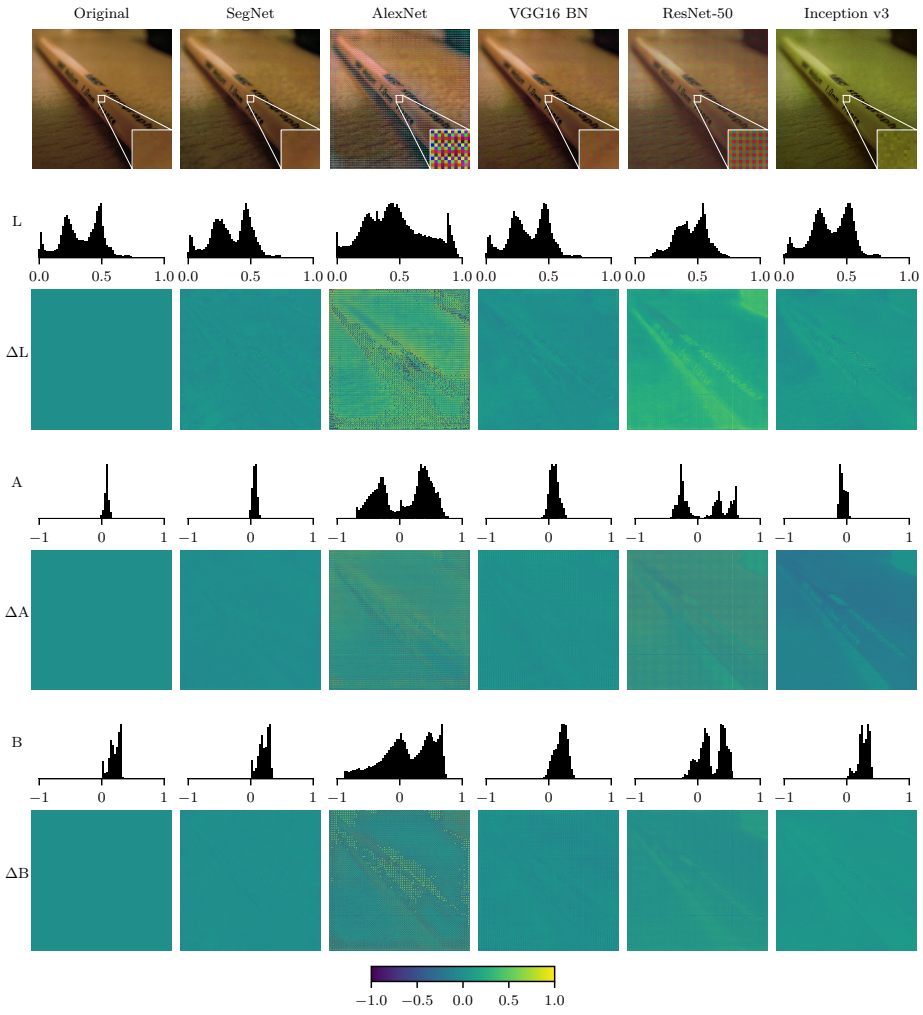


Figure 9: An example of images reconstructed by fine-tined autoencoders.

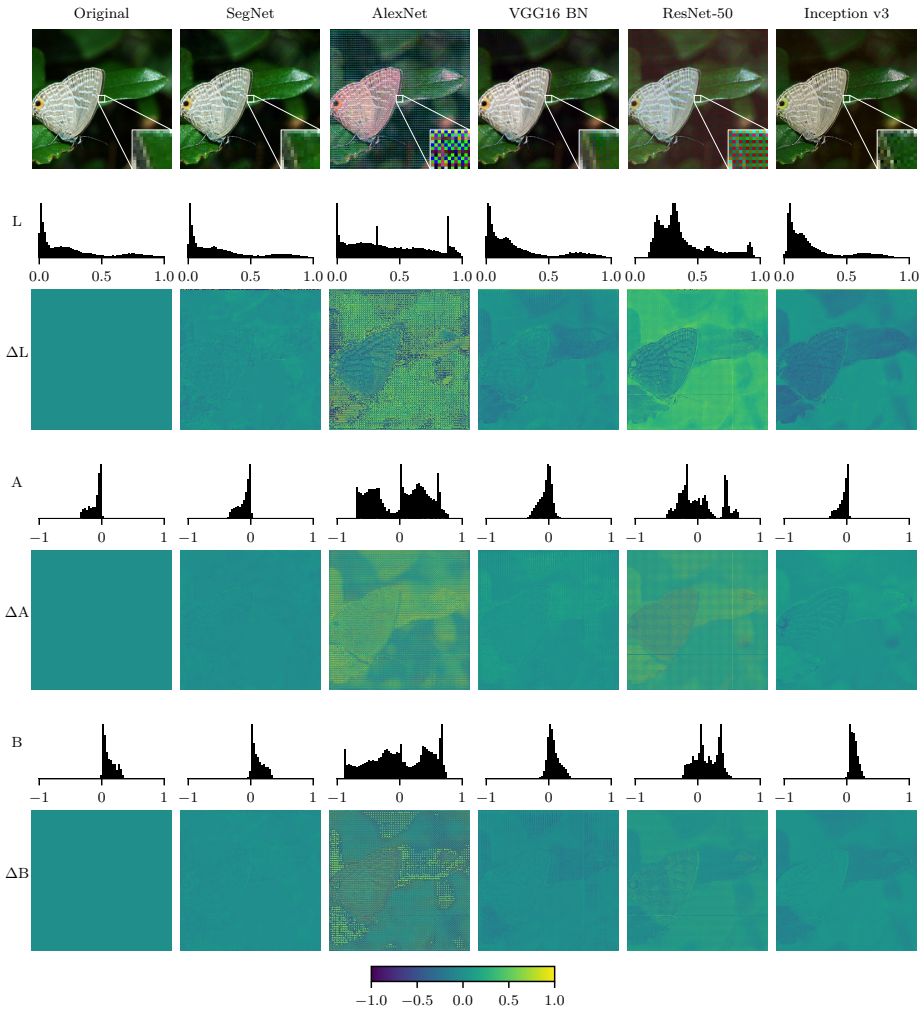


Figure 10: An example of images reconstructed by fine-tined autoencoders.

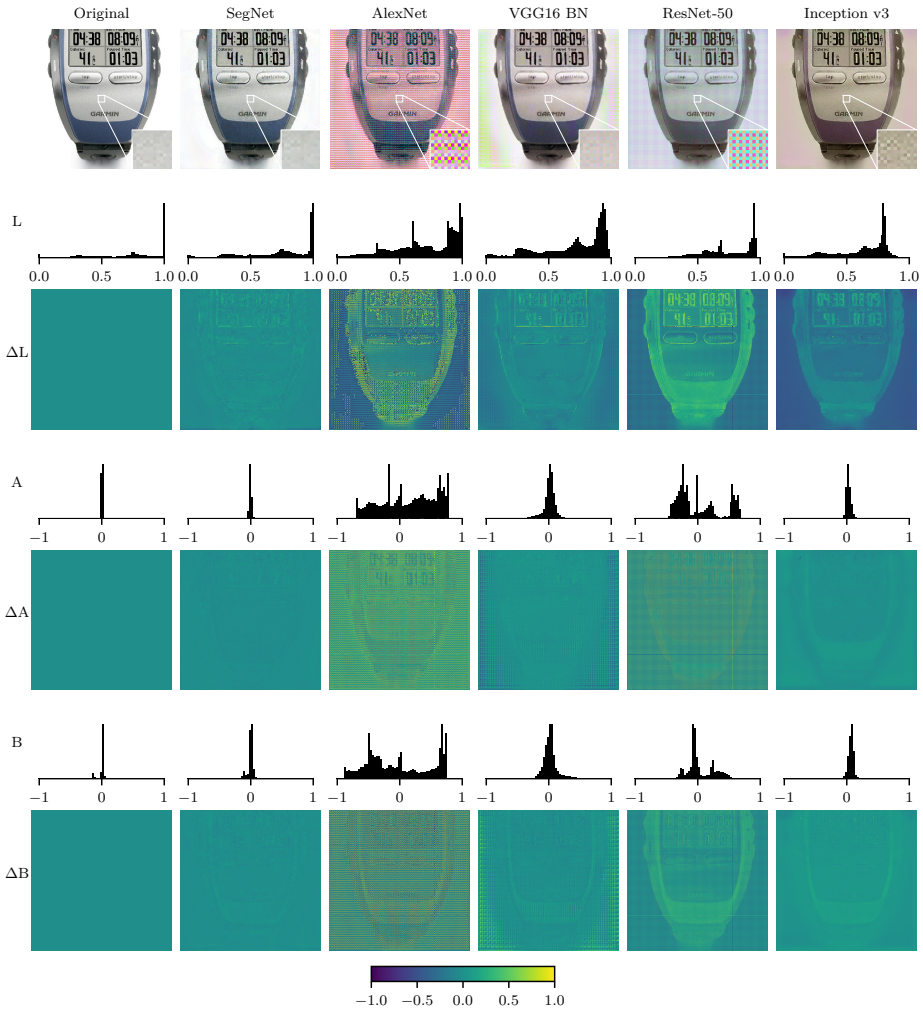


Figure 11: An example of images reconstructed by fine-tined autoencoders.

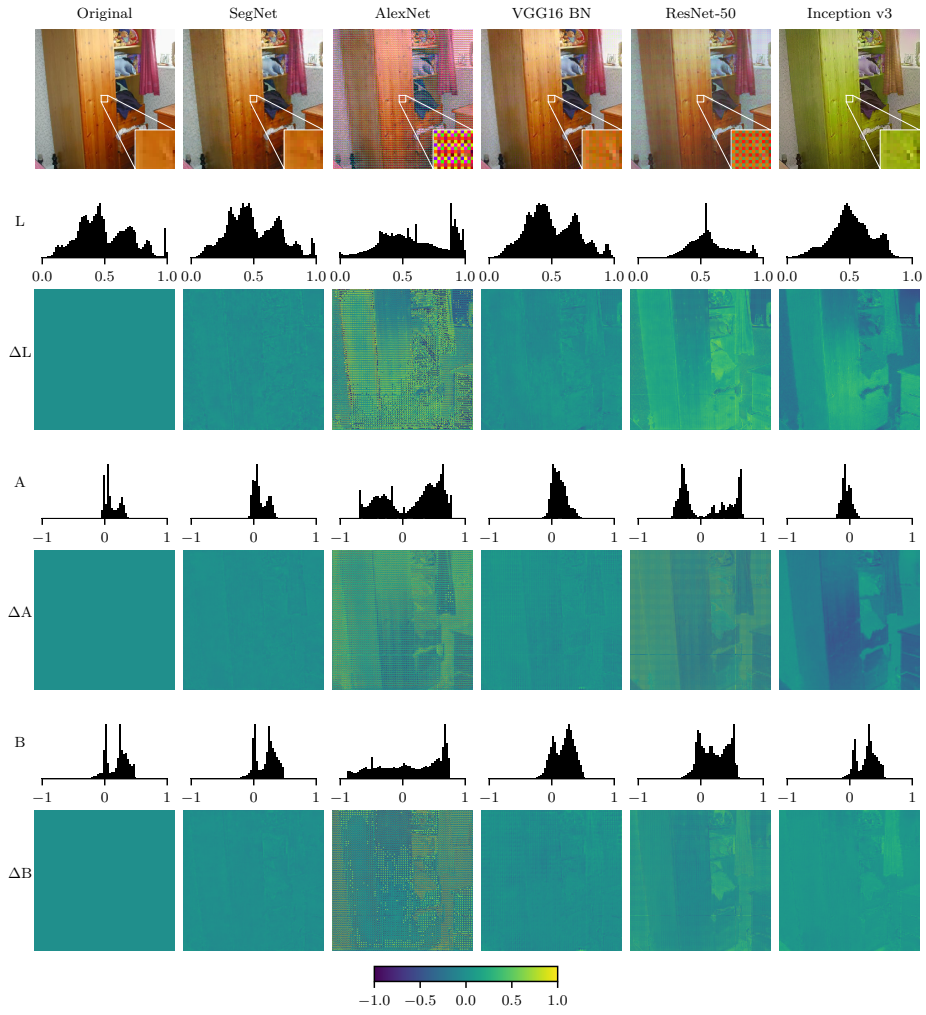


Figure 12: An example of images reconstructed by fine-tined autoencoders.

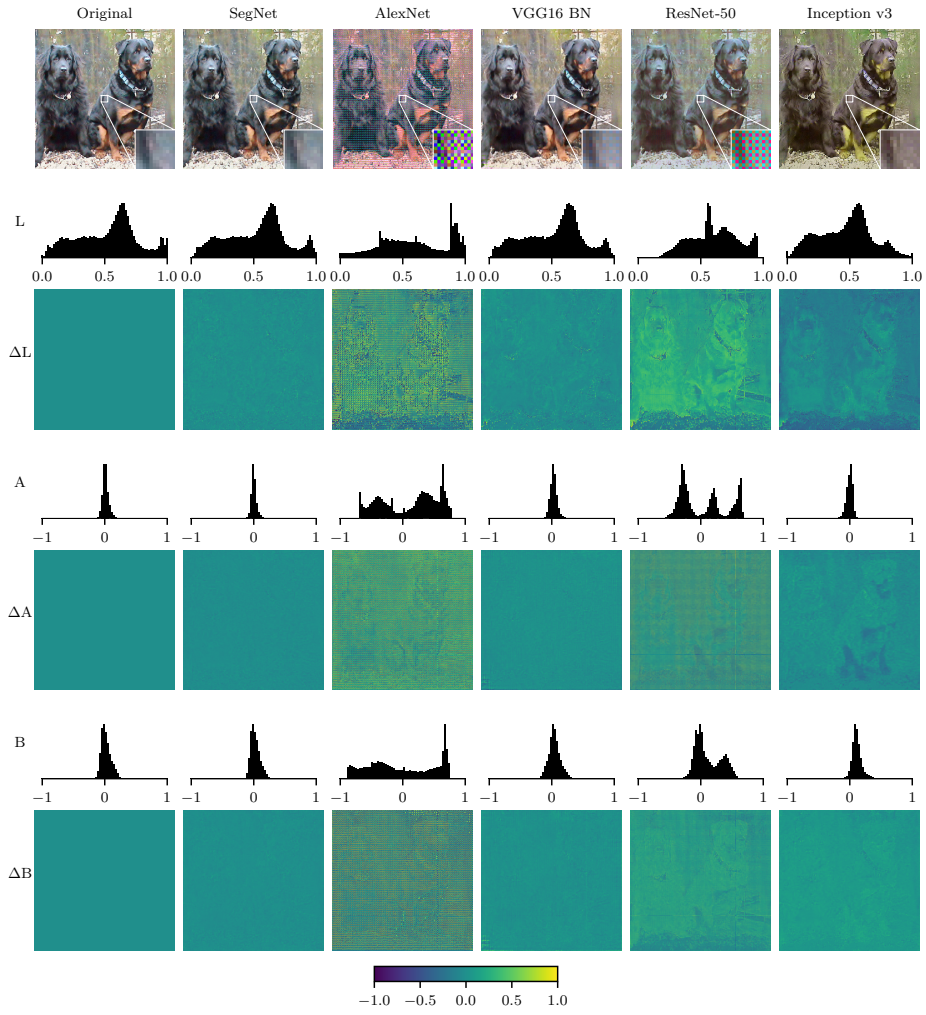


Figure 13: An example of images reconstructed by fine-tined autoencoders.

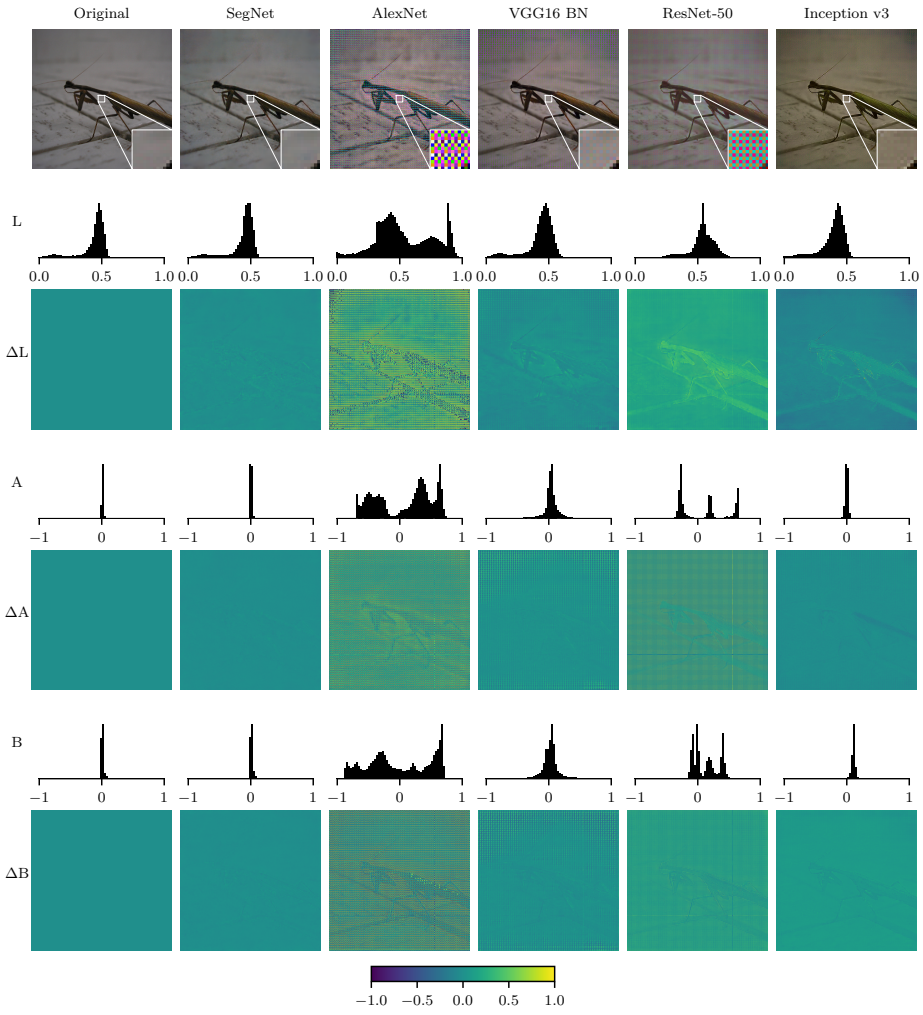


Figure 14: An example of images reconstructed by fine-tined autoencoders.

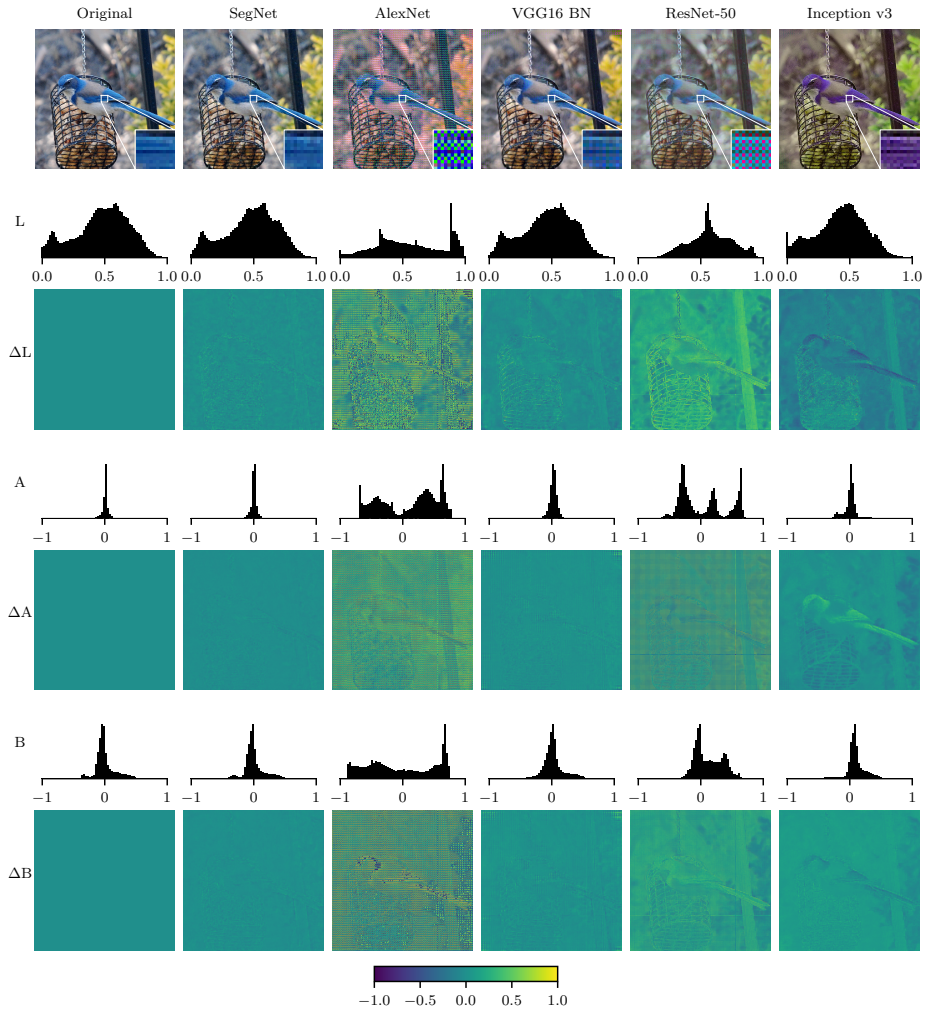


Figure 15: An example of images reconstructed by fine-tined autoencoders.

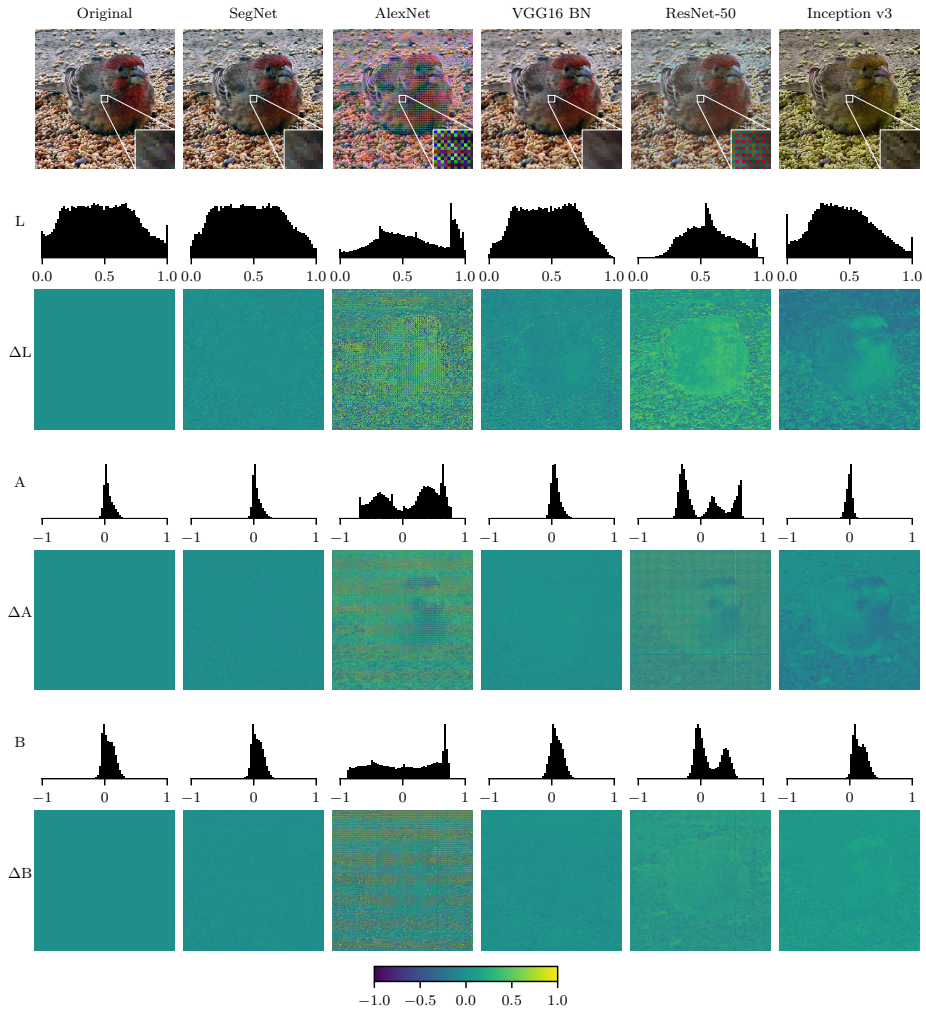


Figure 16: An example of images reconstructed by fine-tined autoencoders.

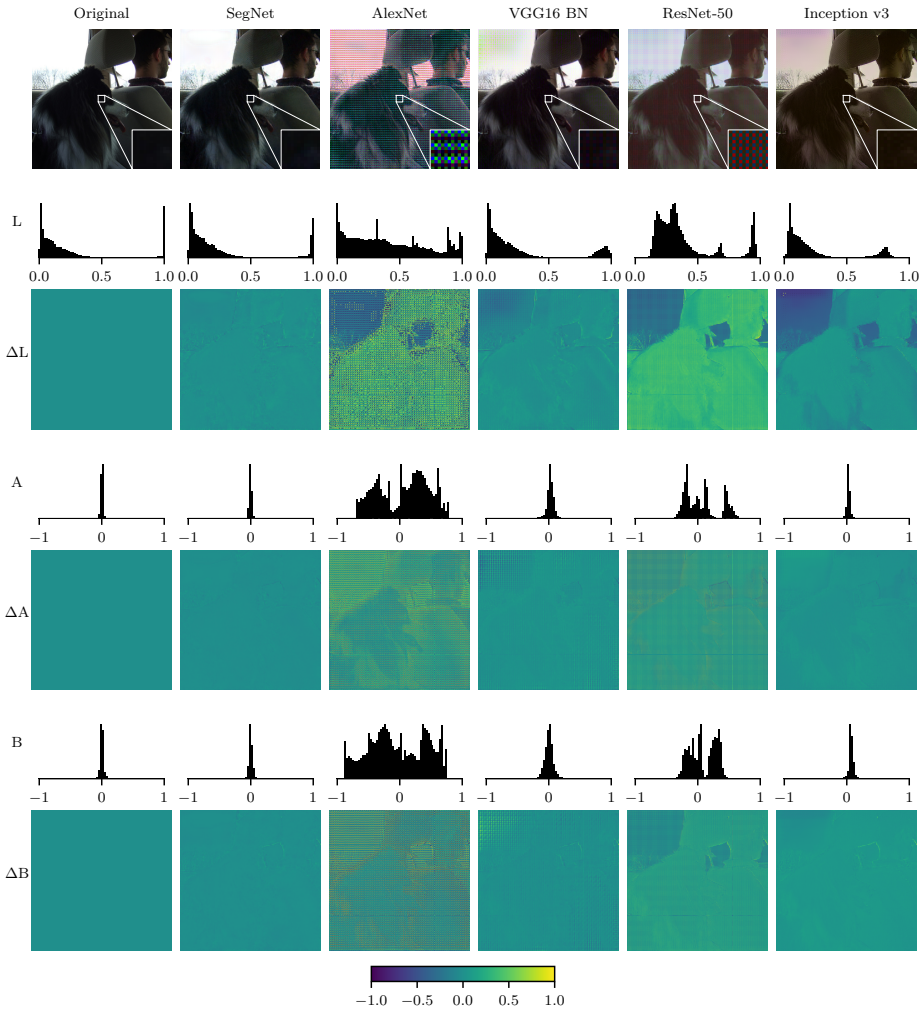


Figure 17: An example of images reconstructed by fine-tined autoencoders.

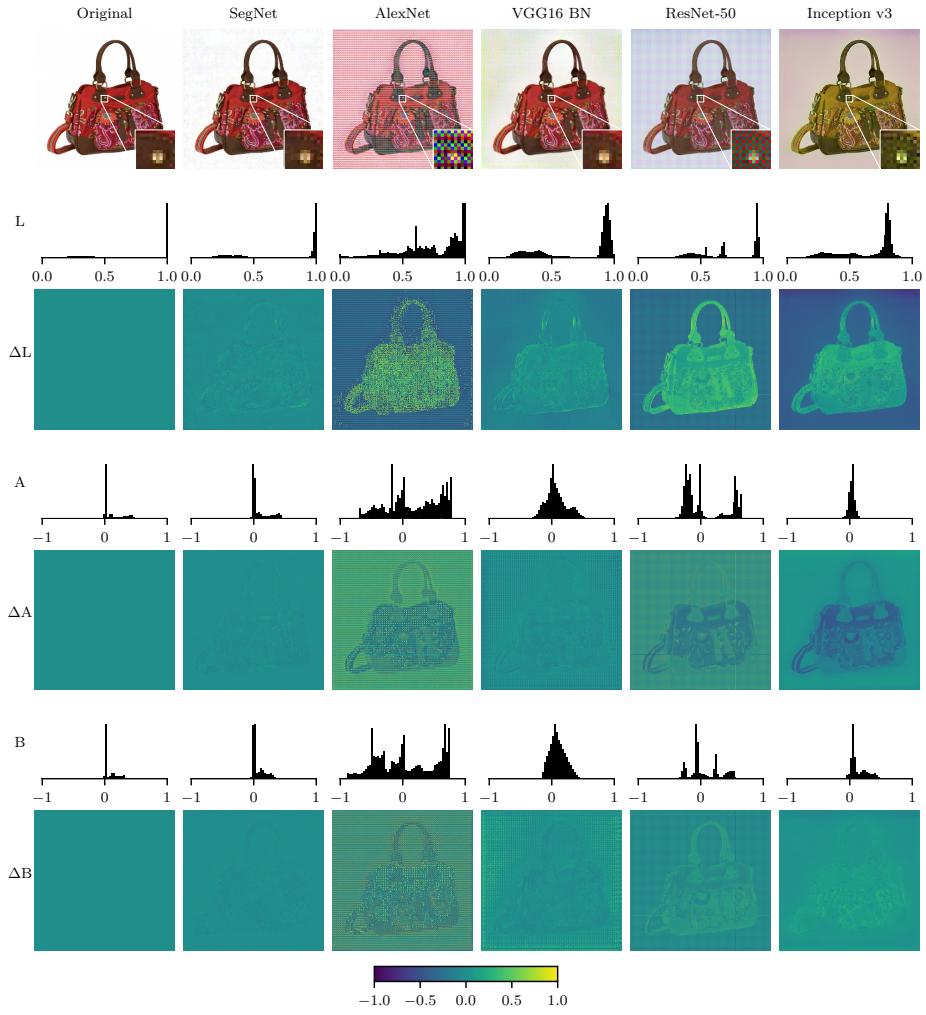


Figure 18: An example of images reconstructed by fine-tined autoencoders.

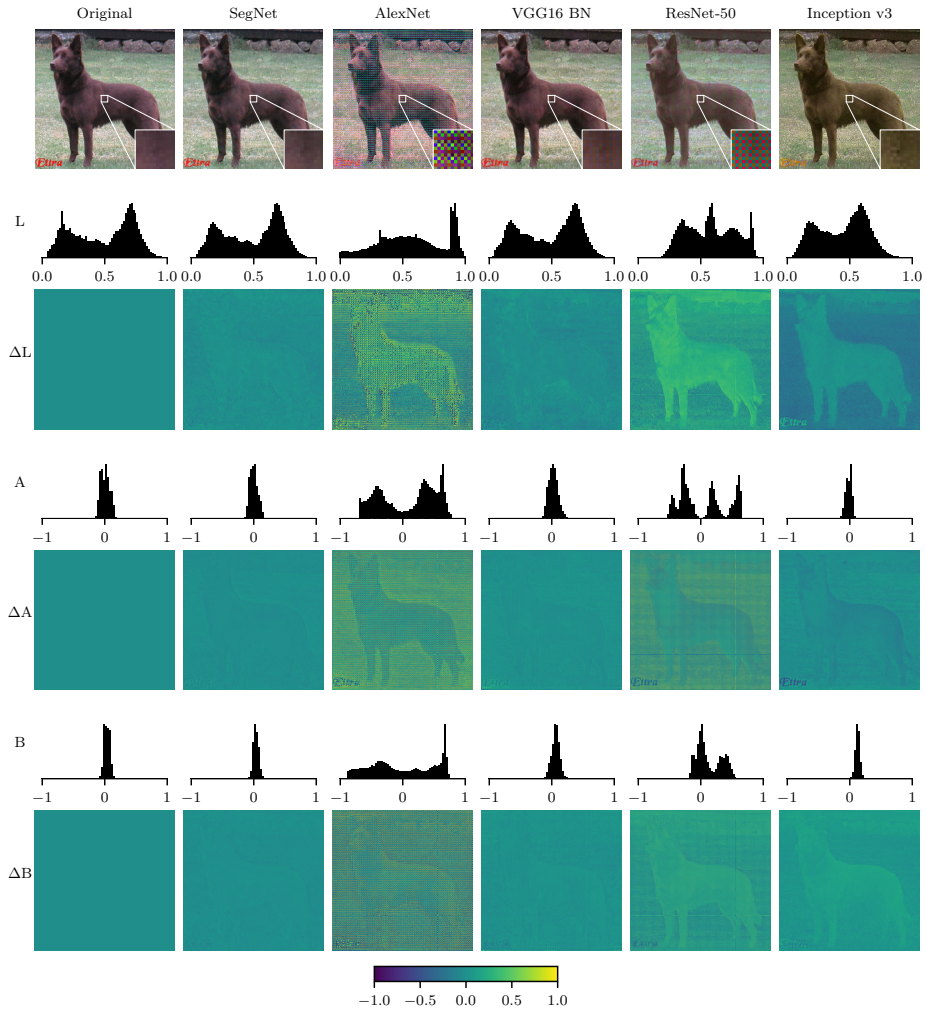


Figure 19: An example of images reconstructed by fine-tined autoencoders.

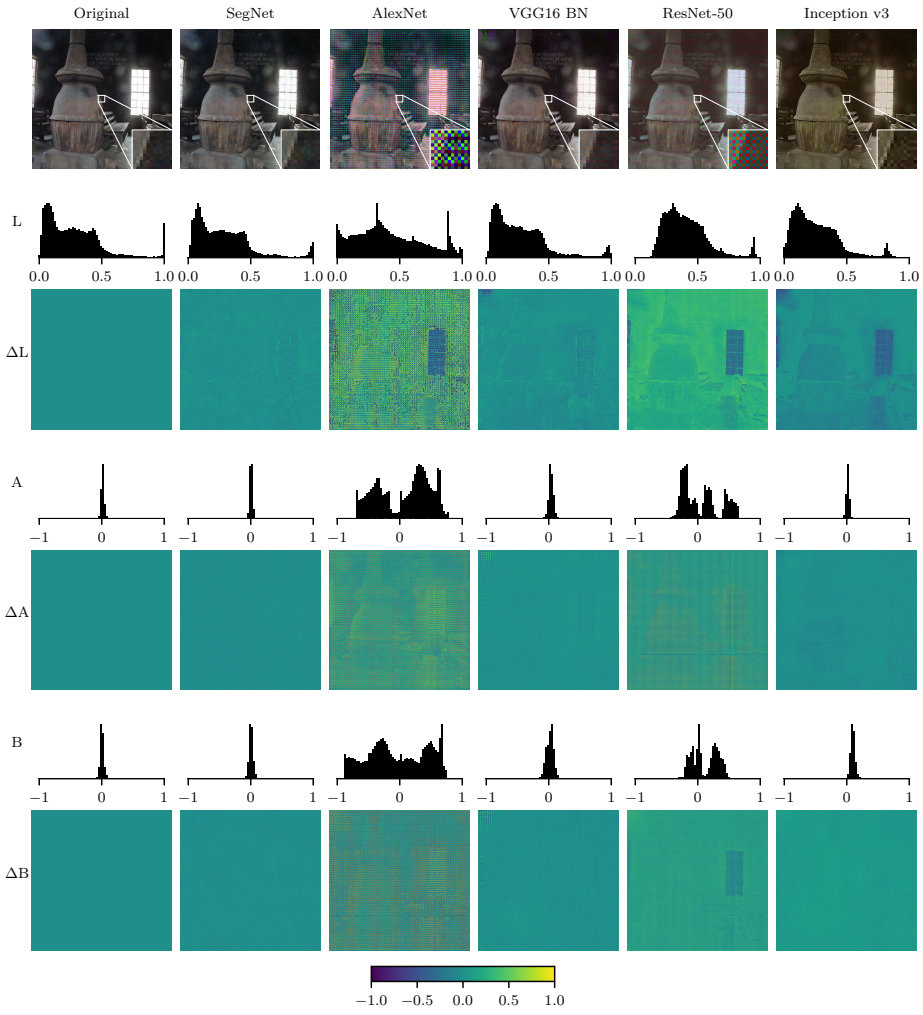


Figure 20: An example of images reconstructed by fine-tined autoencoders.

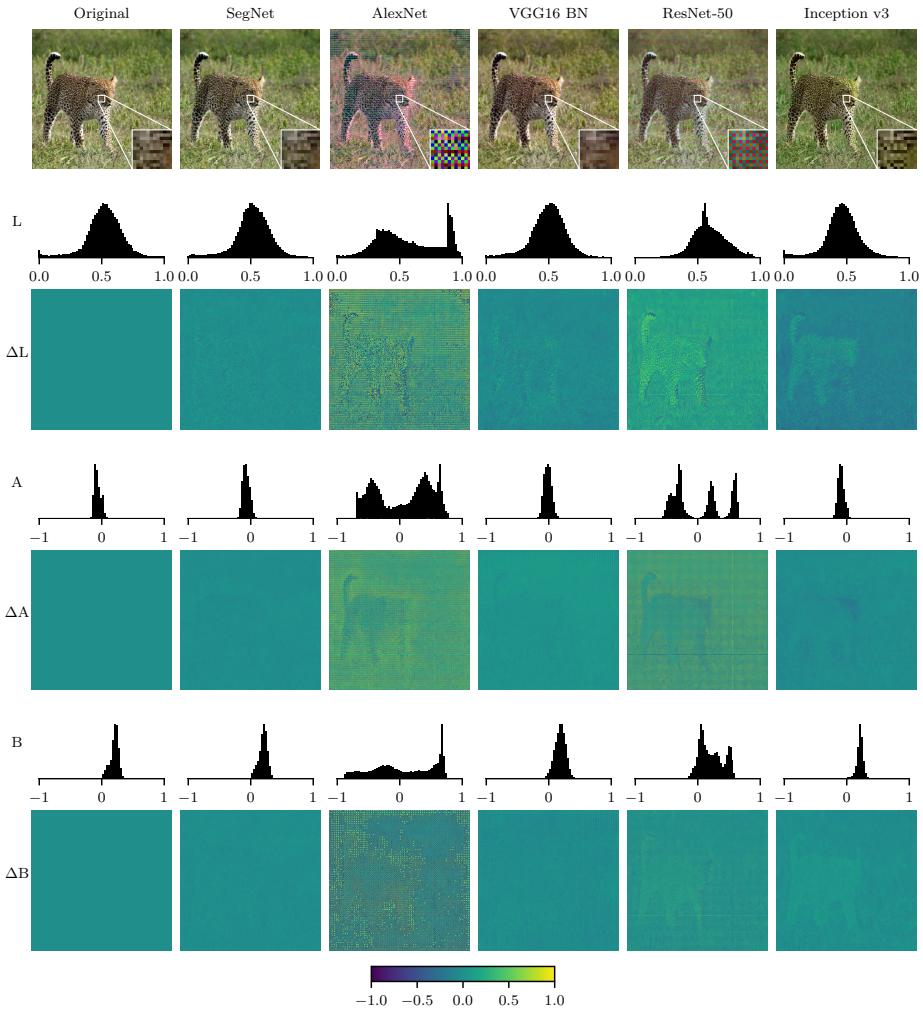


Figure 21: An example of images reconstructed by fine-tined autoencoders.

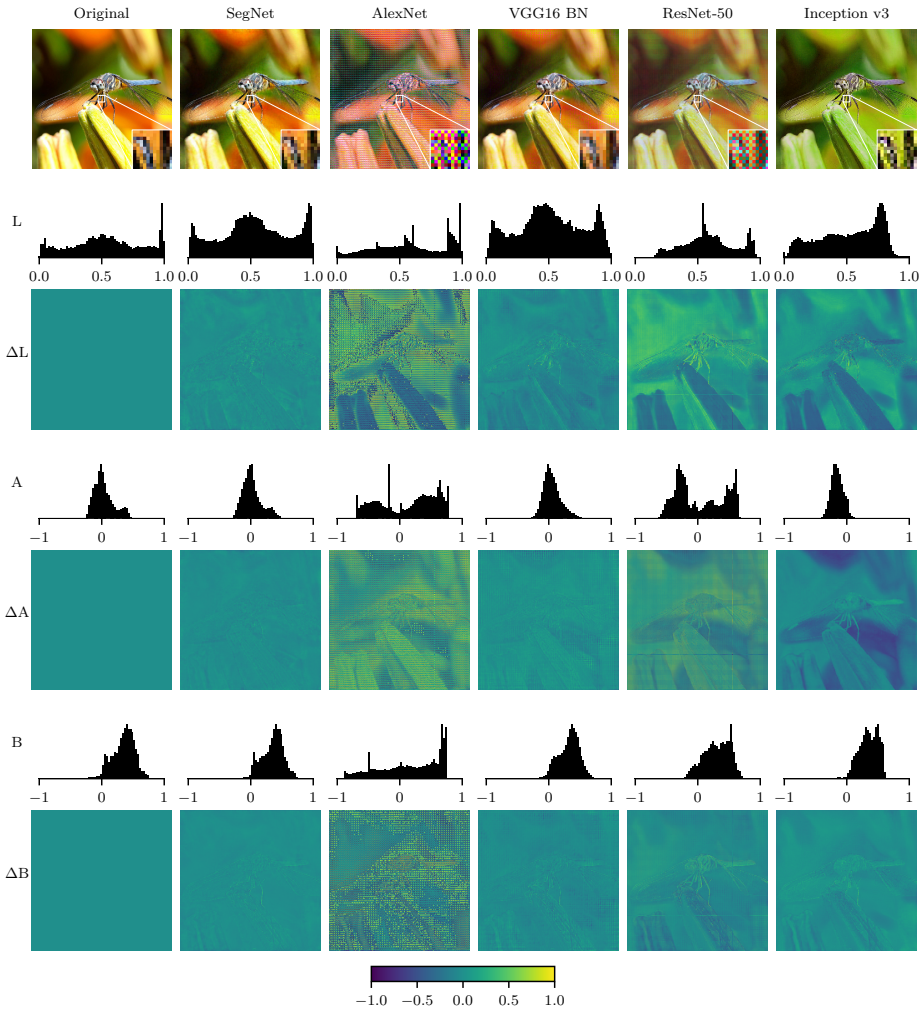


Figure 22: An example of images reconstructed by fine-tined autoencoders.

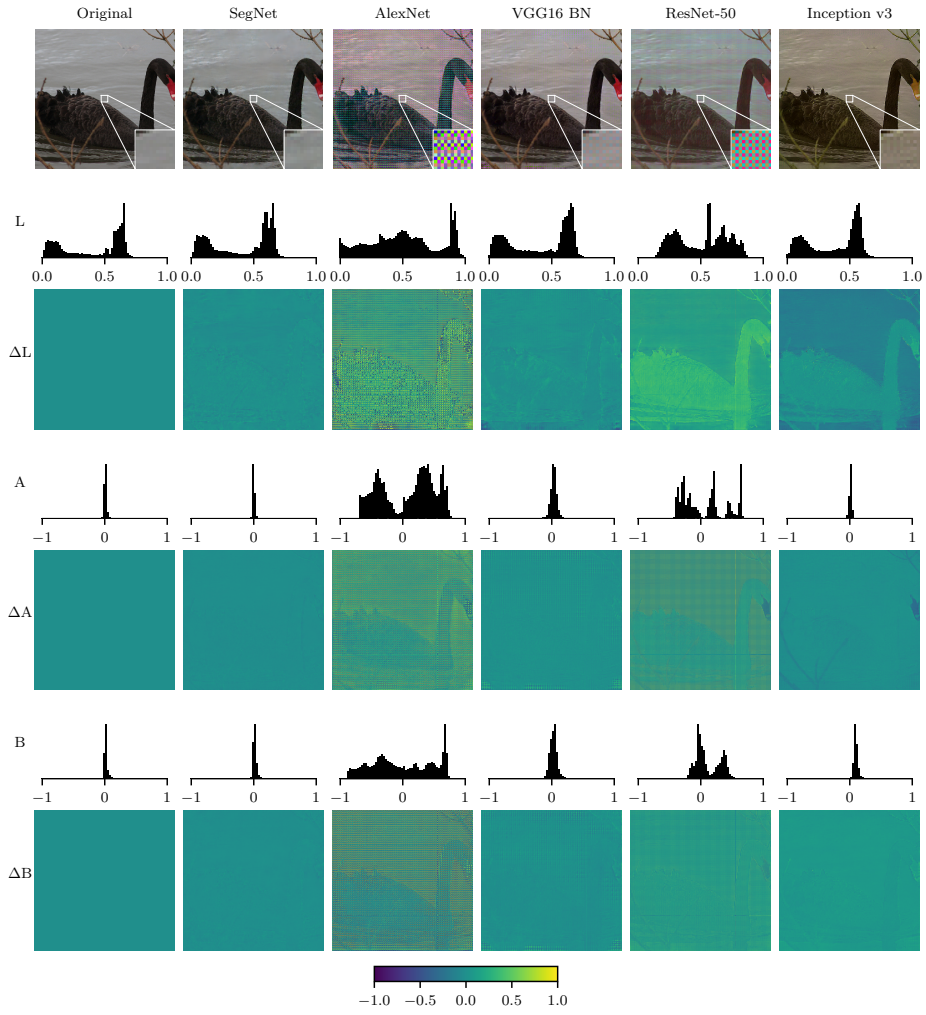


Figure 23: An example of images reconstructed by fine-tined autoencoders.

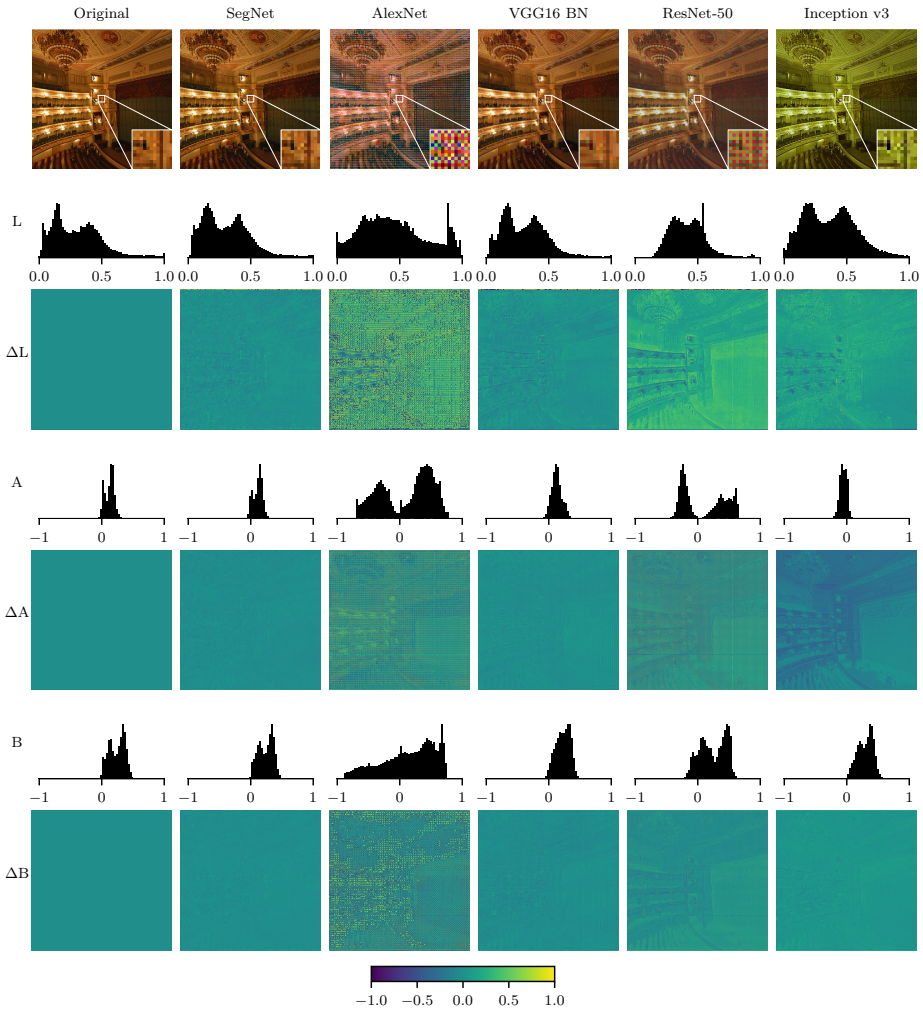


Figure 24: An example of images reconstructed by fine-tined autoencoders.

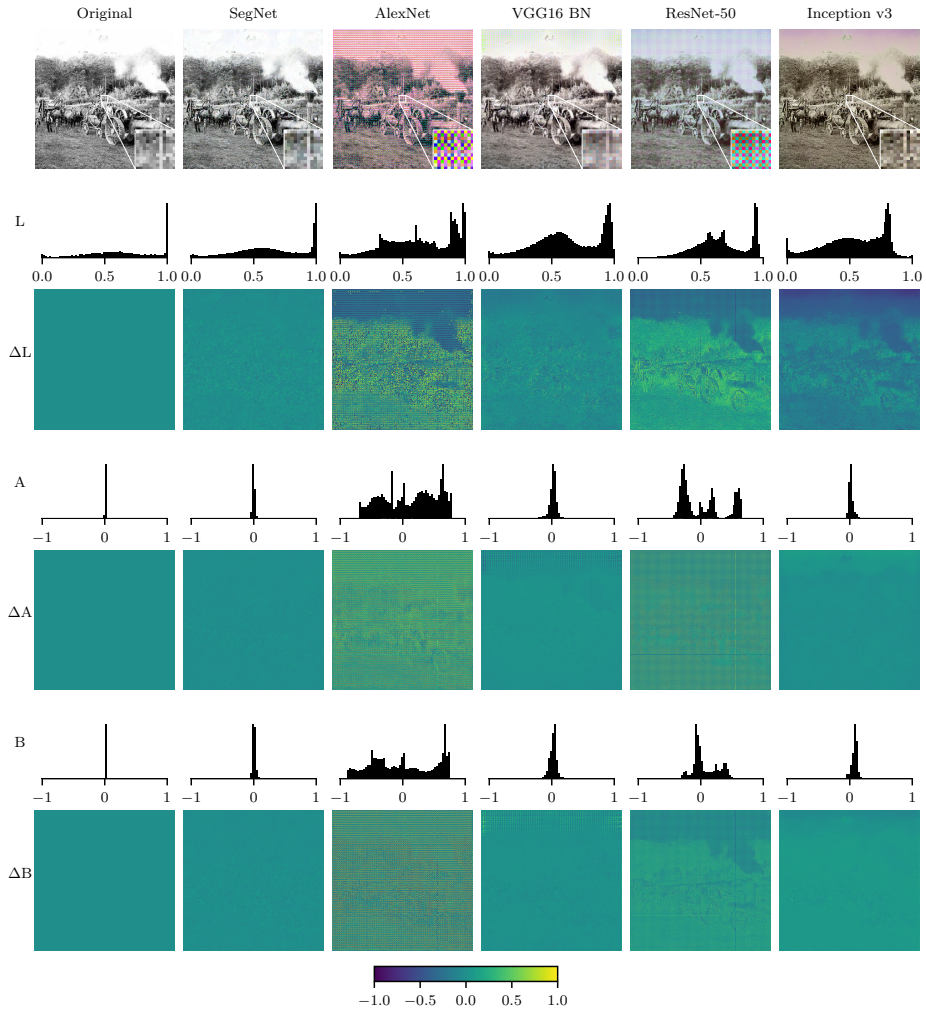


Figure 25: An example of images reconstructed by fine-tined autoencoders.

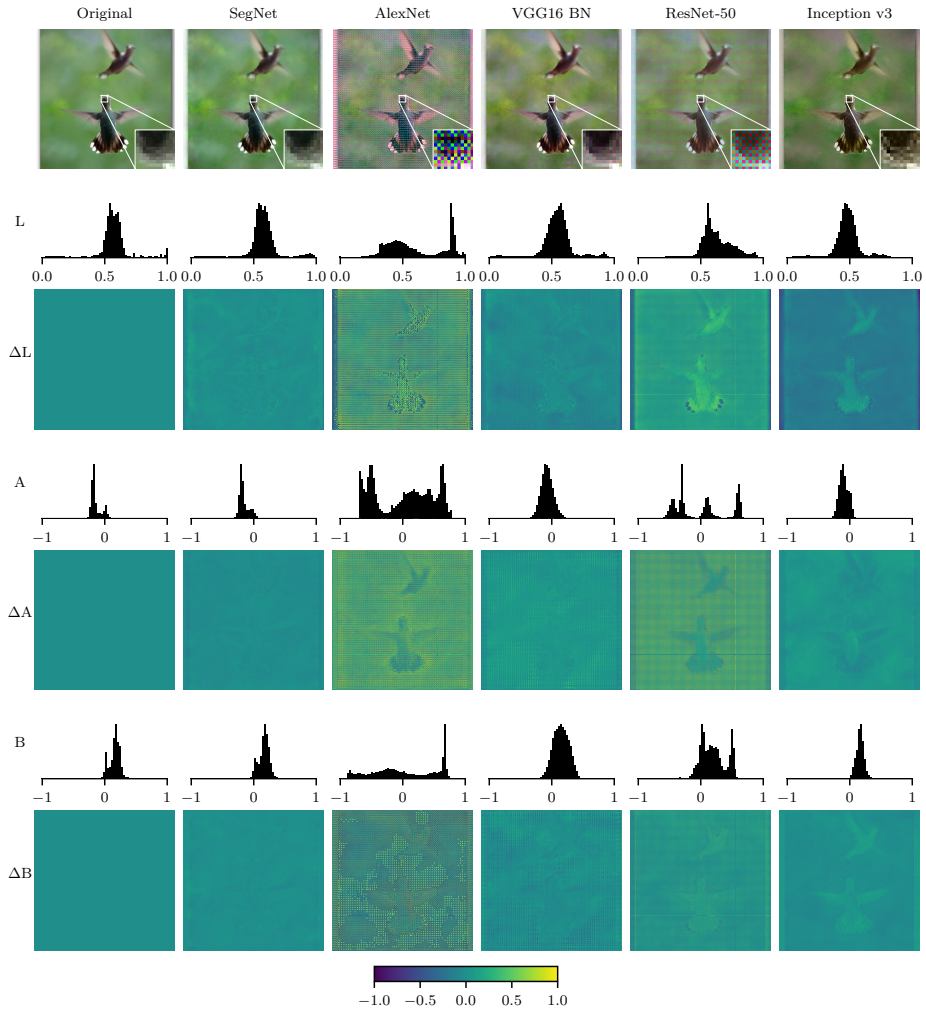


Figure 26: An example of images reconstructed by fine-tined autoencoders.

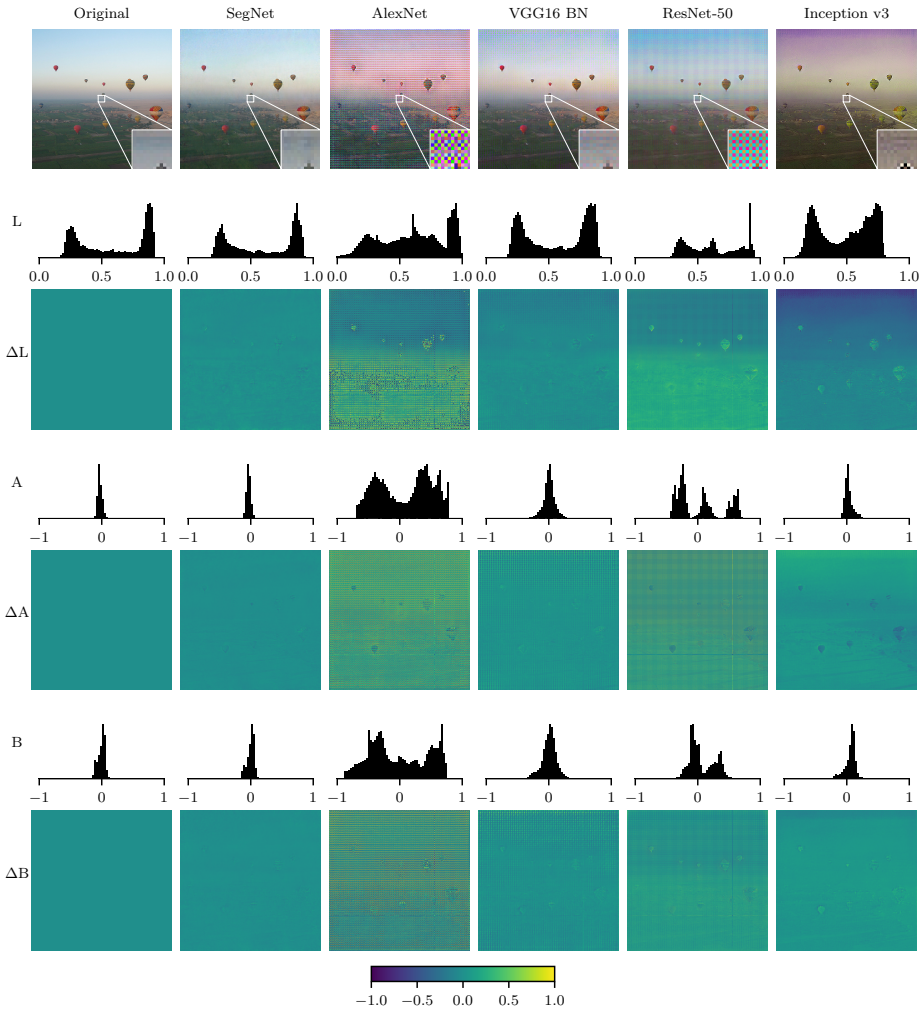


Figure 27: An example of images reconstructed by fine-tined autoencoders.

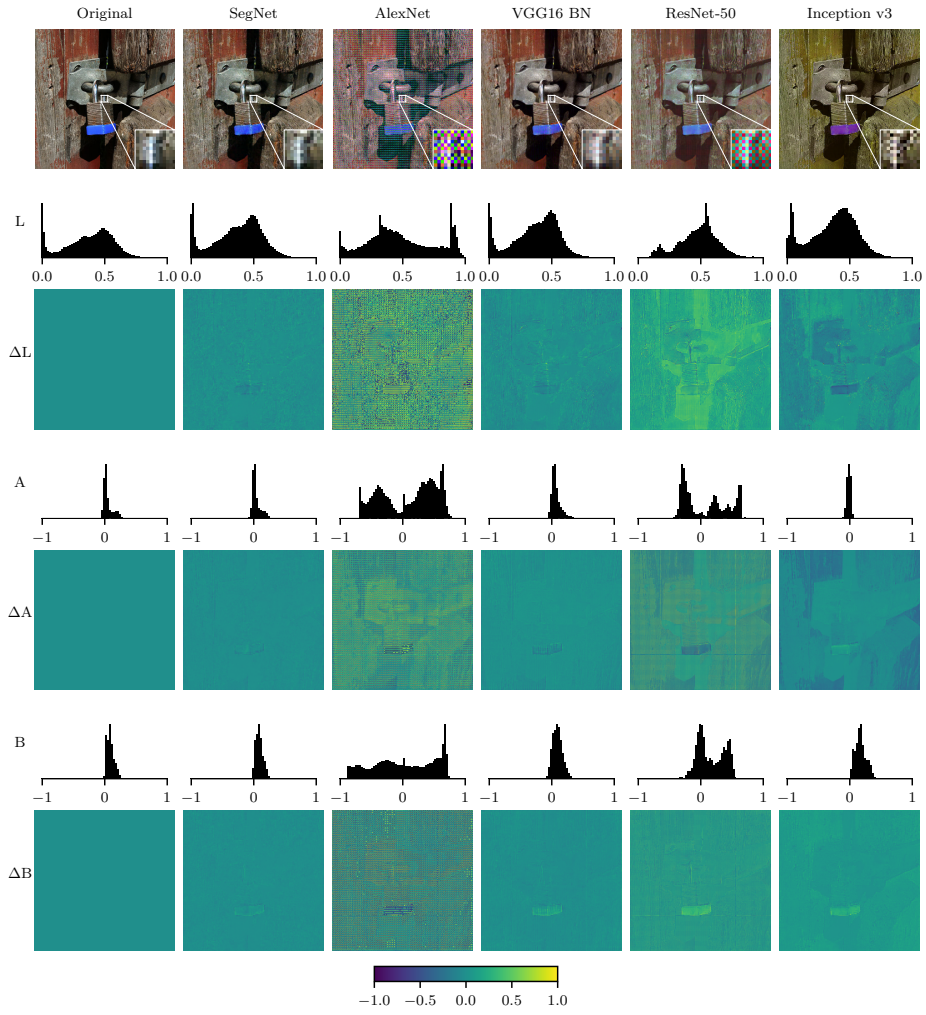


Figure 28: An example of images reconstructed by fine-tined autoencoders.

References

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

[2] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.