Self-Supervised Learning for Segmentation using Image Reconstruction

Srivallabha Karnam
sk3715@rit.edu
Self-Supervised Learning for Segmentation using Image Reconstruction

By

Srivallabha Karnam

May 2020

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering

Committee Approval:

Dr. Raymond Ptucha, Advisor
Associate Professor: Department of Computer Engineering, RIT.

Dr. Andres Kwasinski, Committee member
Professor: Department of Computer Engineering, RIT.

Dr. Sonia Lopez Alarcon, Committee member
Associate Professor: Department of Computer Engineering, RIT.

Department of Computer Engineering
Acknowledgment

I would like to take this opportunity to thank my advisor Dr. Raymond Ptucha for being instrumental in my journey towards a master’s degree. His continuous support and guidance during my master’s degree are commendable. I am thankful for Dr. Andres Kwasinski and Dr. Sonia Lopez Alarcon for being on my thesis committee. I am also grateful for the support that I received from the CE department faculty and staff over the last two years. I would also like to thank my family and friends; without their support, this journey would not have been as joyful as it was.
Abstract

Deep learning is the engine that is piloting tremendous growth in various segments of the industry by consuming valuable fuel called data. We are witnessing many businesses adopting this technology be it healthcare, transportation, defense, semiconductor, or retail. But most of the accomplishments that we see now rely on supervised learning. Supervised learning needs a substantial volume of labeled data which are usually annotated by humans—an arduous and expensive task often leading to datasets that are insufficient in size or human labeling errors. The performance of deep learning models is only as good as the data. Self-supervised learning minimizes the need for labeled data as it extracts the pertinent context and inherited data content. We are inspired by image interpolation where we resize an image from a one-pixel grid to another. We introduce a novel self-supervised learning method specialized for semantic segmentation tasks. We use Image reconstruction as a pre-text task where pixels and or pixel channel (R or G or B pixel channel) in the input images are dropped in a defined or random manner and the original image serves as ground truth. We use the ImageNet dataset for a pretext learning task, and PASCAL V0C to evaluate efficacy of proposed methods. In segmentation tasks decoder is equally important as the encoder, since our proposed method learns both the encoder and decoder as a part of a pretext task, our method outperforms existing self-supervised segmentation methods.
List of Figures

Figure 1. Typical CNN architecture. An image is passed through multiple convolution and pooling operations to extract features and fed to the FC layer for classification. ........... 13

Figure 2. A general pipeline for self-supervised learning [14].................................................. 16

Figure 3 Image representation learning by solving Jigsaw puzzle [26]. The visualization of the Jigsaw puzzle can be seen where (a) is an image with nine sampled image patches, (b) is an example of rearranged image patches, and (c) shows the output of the network which would predict the correct order of the image patches. .................................................. 17

Figure 4 Colorization network predicts CIELAB color space given the lightness channel [27]............................................................................................................................... 17

Figure 5. Rotation feature decoupling network [29]............................................................... 18

Figure 6 Rotation invariance into a self-supervised learning framework. Decoupled semantic feature containing rotation related and unrelated parts architecture. The first part is trained by predicting image rotations. The other part is trained with a distance penalty loss to enforce rotation irrelevance together with an instance discrimination task by using non-parametric classification [26]. .................................................................................. 18

Figure 7 U-Net architecture consists of an encoder (left part) and a decoder path (right part). In the expansive path at the very last layer, we regress for three feature map outputs instead of two, since an image consists of R, G, and B channel........................................ 20

Figure 8 Very deep convolutional network architecture often termed as VGG [12]........ 22

Figure 9 Residual learning block with identity mapping between input and output [32]. 23
Figure 10 Bed of nails. R, G and B channel of an image (row 1). Every other pixel is dropped in R, G and B channels (row 2). Every other two-pixel is dropped in R, G, and B channels (row 3).

Figure 11 Example of the bed of nails, transformed image (left), and the original image (right).

Figure 12 Random channel drop. R, G and B channel of an image (row 1). A random channel in each pixel is dropped in R, G and B channels (row 2).

Figure 13 Example of random pixel drop, transformed image (left) and original image (right).

Figure 14 Random channel drop. R, G, and B channel of an image (row 1). A random channel in each pixel is dropped in R, G, and B channels (row 2).

Figure 15 Example of random channel drop, transformed image (left) and original image (right).

Figure 16 Block diagram of the proposed self-supervised learning method (a) an image where x% of pixels are dropped, (b) an image with every alternate pixel dropped, and (c) every alternate two pixels are dropped and (d) an image where x% of pixel channels are dropped.

Figure 17 Pipeline for classification task.

Figure 18 A pipeline for segmentation task.

Figure 19 Example image and its object segmentation label and class segmentation label from PASCAL VOC [30].
Figure 20 Pretext task results with L2 norm (Drop=50%). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).

Figure 21 Pretext task results with MS-SSIM (Drop=50%). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).

Figure 22 Pretext task results with MS-SSIM-L1 (Drop=50%, λ=0.5). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).

Figure 23 Pretext task results with MS-SSIM-L1 (Drop=50%, λ=0.25). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).

Figure 24 Pretext task results with MS-SSIM-L1 (Drop=75%, λ=0.5). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).
List of Tables

Table 1 Classification downstream task results using a linear classifier. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And λ represents the loss weight factor in (9). ......................... 45

Table 2 Classification downstream task results using a linear classifier. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And λ represents the loss weight factor in (9). ......................... 46

Table 3 Additional classification results for AlexNet and VGG16 architectures. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And λ represents the weight factor in (9). ......................... 47

Table 4 Segmentation downstream task results. The results are reported for all three variants of the encoder network. However, to be fair we compare our method with other methods with AlexNet variant since other methods use AlexNet architecture in their experiments. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in section 3.3. And λ represents the weight factor in (9). ........................................................................................................................................ 48
Contents

Acknowledgment .................................................................................................................. 2

Abstract .................................................................................................................................. 3

List of Figures .......................................................................................................................... 4

List of Tables ........................................................................................................................... 7

Contents .................................................................................................................................... 8

Chapter 1 Introduction .............................................................................................................. 10

1.1 Introduction ......................................................................................................................... 10

1.2 Contributions ....................................................................................................................... 11

Chapter 2 Background ............................................................................................................. 12

2.1 Deep Learning ..................................................................................................................... 12

2.2 Convolutional Neural Networks .......................................................................................... 12

2.3 Supervised Learning ........................................................................................................... 13

2.4 Unsupervised Learning ....................................................................................................... 14

2.5 Self-Supervised Learning .................................................................................................... 15

Chapter 3 Methodology .......................................................................................................... 20

3.1 Backbone architecture ........................................................................................................ 20

3.2 Encoder networks ............................................................................................................... 21

3.3 Pre-text Learning task ........................................................................................................ 24

3.4 Downstream tasks ............................................................................................................... 29

3.5 Loss function ....................................................................................................................... 30

Chapter 4 Datasets ................................................................................................................. 35
4.1 ImageNet .............................................................................................................. 35
4.2 PascalVOC ........................................................................................................... 35
Chapter 5 Experiments and Results ........................................................................ 37
  5.1 Pretext task Experiments ................................................................................. 37
  5.2 Downstream tasks .............................................................................................. 45
  5.3 Discussion on transfer learning ......................................................................... 49
Chapter 6 Conclusion ............................................................................................... 51
Chapter 7 Future Work ............................................................................................ 52
References .................................................................................................................. 53
Chapter 1 Introduction

1.1 Introduction

The ability of the deep neural networks to learn features has demonstrated exceptional performance on a variety of challenging tasks such as object detection [1], [2], [3], semantic image segmentation[4], [5], [6], and image recognition [7]. Their success relies on a large amount of labeled data. It is arduous and expensive to create annotated datasets. So, building large datasets for different applications and different scenarios is practically not feasible. On the other hand, we generate unlabeled data all the time. This unlabeled data constitutes an important resource. It remains an outstanding research problem to take advantage of a variety of information that is inherited within the data.

Self-supervised learning facilitates learning semantic and structural connotations of the data which provides efficient representations for downstream tasks without the need for extensive annotated data. These tasks often need unlabeled data to devise a pretext learning task such as foretelling image rotation [8] or context [9]. These pretext tasks must be devised in such a way that high-level image understanding emerges which is useful for solving downstream tasks. This method not only helps to overcome the need for large amounts of annotated data but also helps to improve model robustness and uncertainty [8].
1.2 Contributions

The following are the key contributions of this research:

1. We introduce a novel self-supervised label generation method that is specifically targeted towards image segmentation.
2. We study the effect of the complexity in pretext tasks on the downstream tasks.
3. We study the effect of different loss functions for our pretext task and its impact on downstream tasks.
Chapter 2 Background

2.1 Deep Learning

Deep Learning is a sub-field of machine learning and utilizes a hierarchical level of artificial neural networks. Inspired by the human brain, deep learning has empowered many practical applications. Deep learning is a self-adaptive algorithm and gets progressively better at recognizing the underlying patterns with experience and or new data. Due to this ability, it has provided massive breakthroughs in the fields of computer vision, natural language processing, time-series prediction, speech recognition, and many more. LeNet [10] was the first deep convolutional neural network that surpassed the performance of many traditional computer vision methods, involving handcrafted features in the late ’90s. However, deep neural networks like AlexNet [11] and VGGNet [12] gained popularity in this decade due to improvements in computing and data availability (due to internet explosion and crowdsourcing). At the core, deep learning has three different types of neural networks based on the way each of them interacts with the world, namely artificial neural network (ANN), convolutional neural network (CNN), and recurrent neural network (RNN). CNN’s have been a backbone for most recent developments in computer vision. By devising the appropriate loss function, CNN’s can be used in a variety of computer vision applications like object recognition, pixel-level segmentation, and object detection.

2.2 Convolutional Neural Networks

CNN’s are very effective to learn and extract features from gridded data structure like images due to its space invariant and translation invariant characteristics. CNN’s are inspired by the human brain and the connectivity between the neurons resembles the visual cortex. A typical CNN architecture as shown in Figure 1 and consists of basic operations namely convolution layer, pooling layer, activation layer, and a fully connected (FC) layer.
A convolutional layer comprises of learnable filters that have a predefined receptive field. The convolution operation typically preserves the dimensions of the input image. We perform this operation repeatedly along the x and y-axis. The pooling layer downsamples the input that it receives from the previous layer. Typically, an image passes through various levels of convolution and pooling layers before it is fed to a fully connected layer. These layers together learn to extract meaningful features that are then fed to a fully connected layer for classification. Non-linearities are introduced into the network using an activation layer like ReLU after each stage. We can notice that the features and classifier are learned together which makes it powerful when compared to traditional computer vision methods.

In general, there are three ways to learn the network parameters namely supervised learning, unsupervised learning, and self-supervised learning.

2.3 Supervised Learning

Supervised learning is the most common learning algorithm which is designed to learn by example. As the name “supervised” learning suggests, training this type of algorithm is like having an instructor to supervise the entire learning process. The training data consists of paired data of input and the corresponding target label as output; usually, the corresponding target labels are manually created. During training, the algorithm will
analyze the training data to approximate a mapping function. Ideally, after the training, the algorithm should correctly determine the correct output label for new data. In the simplest form, supervised learning can be represented as,

\[ Y = f(X) \]  

(1)

Where ‘X’ is the input variable, ‘Y’ is an output variable, and function ‘f’ is a mapping function. The goal is to approximate the mapping function to predict the output label for input data. In particular, the mapping function should generalize for unforeseen data.

Supervised learning has demonstrated exceptional performance on a variety of challenging computer vision tasks such as object detection, semantic image segmentation, and image recognition. One of the disadvantages of this method is their success relies on a large amount of labeled data which is arduous and expensive to collect. This motivated the research community to explore methods to overcome the need for labeled data.

### 2.4 Unsupervised Learning

Unsupervised learning overcomes the disadvantages of supervised learning i.e. need for the large labeled dataset. In unsupervised learning, we only have input data and no corresponding output labels. Algorithms are used to group the data that has not been labeled and the most common example of unsupervised learning is clustering, where we group the input samples based on feature similarity. The objective of the unsupervised learning method is to model the underlying probability densities of the input data so that the model can be used for other tasks.

Supervised models always perform better than unsupervised methods since the supervision i.e. labeled data forces the model to encode the characteristics of the dataset effectively. However, the performance decreases when applied to other tasks. Unsupervised learning
can learn more general features that ensure minimal performance drop when applied to other tasks.

Autoencoders [13] is an example of unsupervised learning. These are inspired by the primary visual cortex (V1) of the human brain which creates a minimal set of base functions based on the principles of sparsity to reconstruct the input image i.e. they try to learn an approximation to an identity function so that the output is similar to input data. This method typically does not use a global optimization function and each layer is trained greedily making learning parameters in deep networks difficult. These methods do not match the performance of supervised learning when applied to downstream tasks.

2.5 Self-Supervised Learning

Self-supervised learning methods involve learning visual features from unlabeled data without human annotations. In general, a pretext task is devised for CNN’s to solve. A general pipeline is shown in Figure 2 [14]. During self-supervised training, a pretext task is defined to solve for CNN’s and pseudo labels are automatically generated based on some inherited property of the visual images. CNN’s are trained to solve the objective function of the predefined pretext task and once the training is finished, the learned weights are utilized as a feature extractor or transferred to a downstream task. The downstream task is a method to evaluate the quality of features learned during self-supervised learning and generally are computer vision applications but in some cases, the downstream task is the same as pretext task, for example, image restoration. In practice, during downstream tasks we repurpose the learned weights to achieve best performance on the target application using supervised learning approach. This not only helps in reducing the need for labeled data, but also provides better performance than solely using supervised learning. Self-supervised learning was coined by Yan Lacunae in 2018 and it has recently gained huge popularity. Many self-supervised methods have been developed for visual feature learning tasks [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25] since its inception.
Images are plentiful in spatial context information. Nosroozi et al. [26] introduced the image Jigsaw puzzle which learns spatial relations among the patches of the image. In this work, the pretext task is to recognize the order of the shuffled sequence of patches from the same image as shown in Figure 3. The shuffled images are fed to the network to
recognize the correct order of the shuffle images. Since there are nine patches in the image, there are 9! possible permutations and are very unlikely to recognize all of them. So, they employed a hamming distance to limit the number of permutations. This ensures the task is suitable for CNN which is not too difficult nor too easy.

Figure 3 Image representation learning by solving Jigsaw puzzle [26]. The visualization of the Jigsaw puzzle can be seen where (a) is an image with nine sampled image patches, (b) is an example of rearranged image patches, and (c) shows the output of the network which would predict the correct order of the image patches.

Zhang et al. [27] introduced a method that firstly converted RGB to CIELAB color space. Then given the lightness channel, the network predicts the a* and b* color channels as shown in Figure 4. The network astoundingly performed well for downstream tasks like object classification, detection, and segmentation compared to other methods.

Figure 4 Colorization network predicts CIELAB color space given the lightness channel [27].
Z. Feng et al. [28] employ rotation invariance into a self-supervised learning framework. As shown in Figure 5, the input images are rotated as they are being fed to a neural network to jointly predict image rotations and individual image instance class [28].

Pathak et al. [29] present a context-based pixel prediction where the network understands the context of the entire image as well as the hypothesis for missing parts as shown in Figure 6. The reconstruction of the image might perform well if the missing image is symmetric to parts of the image, but if the missing image is a unique part by itself, it would become harder for the network to converge, failing to learn the generic representation of the image.
Figure 6 Context Encoder architecture. The context image is passed through the encoder to obtain features and then the decoder produces the missing regions in the image [29].

Noroozi et al. [25] introduced a method to transfer the representations learned during pretext task to an object detection problem and it has led to an improvement in mAP scores over supervised learning. Also, Hendrycks [30] provided evidence that the self-supervision improves robustness in a variety of ways, involving robustness to label corruption, adversarial examples, and common input corruption.

The methods which are introduced until now use simple inherited features of images like image patches and rotation. These are simple tasks that may not completely learn the semantic and spatial features of the image. We believe the methods that are proposed in this work would perform better because of how we formulated our self-supervised learning task.
Chapter 3 Methodology

In this section, we enlist the various architecture and networks that are set up and explore the differences between them.

3.1 Backbone architecture

Figure 7 U-Net architecture consists of an encoder (left part) and a decoder path (right part). In the expansive path at the very last layer, we regress for three feature map outputs instead of two, since an image consists of R, G, and B channel.

We use U-Net [31] as a backbone architecture. This architecture was developed for a biomedical image segmentation application which gained popularity due to its superior performance with minimal training samples. It consists of two paths as shown in Figure 7; the First path is termed as an encoder or contracting path (left part of Figure 7) which captures the context in the image. The encoder is a typical CNN’s which is often used for classification task which generally consists of a set of convolutional and pooling layers.
The second path is called a decoder or expansive path (right part of Figure 7) and is symmetric to the encoder. The purpose of the decoder is to enable precise localization using deconvolution operation also termed as transposed convolution.

The contraction path applies two repeated unpadded convolutions of size 3×3, followed by a rectified linear unit (ReLU) and 2×2 max-pooling operation of stride 2 which reduces the spatial resolution by half. At each step, the spatial resolution is halved, and the number of features is doubled. The decoder consists of the upsampling of a feature map followed by deconvolution of size 2×2 that halves the feature channels and concatenate with the corresponding feature map from the encoder.

We use U-net as a backbone architecture for all our experiments, however, we experiment with a few variants of encoder networks to see if our proposed method is invariant to network architecture.

### 3.2 Encoder networks

We use AlexNet [11], VGG16 [12], and ResNet50 [32] encoder networks for our experiments in this work. There are many more modern architectures like ResNext [33], Inception V4 [34], and XceptionNet [35] which are shown to provide better performance than AlexNet, VGG16 and ResNet50 on the benchmark datasets. Although it is likely to yield better results with these modern architectures, we use AlexNet, VGG16, and ResNet50 to compare and evaluate our methods directly with existing self-supervised learning methods.

#### 3.2.1 AlexNet

AlexNet [11] was the first deep neural network trained on a large ImageNet dataset [7] and won ILSVRC-2012 competition surpassing many traditional computer vision methods by
a large margin. This network consists of five convolutional layers and three fully connected layers. We use all the convolutional layers as a part of the encoder in the U-Net architecture.

3.2.2 VGG16

The VGGNet [12] architecture used smaller convolutional filters and increased the depth of the network compared to prior methods and were able to achieve a state of the art performance on the ImageNet dataset. There are different variants of this architecture-based number of layers, we use a 16-layer variant network named VGG16 for our experiments in this work. Figure 8 shows the architecture of VGG16. We use all the layers up until fully connected layers as a part of the encoder in the U-Net architecture.

Figure 8 Very deep convolutional network architecture often termed as VGG [12].
3.2.3 ResNet

The ResNet [32] architecture presents a method to reformulate sequential convolutional layers to learn residual parameters. This led to address one of the critical vanishing gradient problems which affect the convergence of the optimization function in deeper networks.

Figure 9 Residual learning block with identity mapping between input and output [32].

The residual learning block is shown in Figure 9 where $F(X) + X$ can be realized with feedforward shortcut connections performing identity mapping between the input and output across stacked layers as shown in Figure 9. Identity shortcut connections do not add additional parameters nor increase the computation complexity of the network. There are many variants of ResNet architecture based on the number of layers and we use the 50-layer variant called ResNet50. Like in AlexNet and VGG16, we use all layers of ResNet50 except the classification layer i.e. fully connected layers.
3.3 Pre-text Learning task

The pretext task drives the model to learn semantic features of the images by creating the ground truth labels using known input properties or known input transformations. The following are the methods that we propose to generate self-supervised labels.

3.3.1 Method 1: Bed of nails

Image interpolation is a well-known technique in the field of digital image processing popularly used for image resize and to remap images. Image resize is used when we need to increase or decrease the number of pixels in an image. Whereas remap is essential under circumstances like correcting lens distortion, changing perspectives, and rotating images.

For example, consider an image of size 256×256. To increase the resolution to 512×512 we use interpolation to fill the empty pixels.

The following self-supervised label generation is inspired by the interpolation technique. For example, take a toy 5×5 input image with a red, green, and blue channel as shown in Figure 10 (row 1). A simple way to reduce the image resolution would be to drop pixels along x and y-axis of an image as shown in Figure 10 (row 2 & 3 respectively), and an example of an image is shown in Figure 11. This is used as input to the neural network and original image as ground truth as shown in Figure 16.
Figure 10 Bed of nails. R, G and B channel of an image (row 1). Every other pixel is dropped in R, G and B channels (row 2). Every other two-pixel is dropped in R, G, and B channels (row 3).

Figure 11 Example of the bed of nails, transformed image (left), and the original image (right).
3.3.2 Method 2: Random pixel drop

In this method, we randomly drop the pixel of all channels. For example, as shown in Figure 12, the input image of size 5×5 in row 1 and random pixel being dropped is shown in row 2 and an example of an image is shown in Figure 13. We experiment with randomly drop X\% of pixels in an image to understand the impact of increased pretext task complexity on downstream tasks, where X equals 50\% and 75\%.

Figure 12 Random channel drop. R, G and B channel of an image (row 1). A random channel in each pixel is dropped in R, G and B channels (row 2).

Figure 13 Example of random pixel drop, transformed image (left) and original image (right).
**3.3.3 Method 3: Random channel drop**

In this method, we randomly drop the red, blue, or green channel of a pixel. For example, as shown in Figure 14, the input image of size 5×5 in row 1 and random channels of each pixel being dropped is shown in row 2, and an example of an image is shown in Figure 15. We experiment with randomly drop $X\%$ of pixels in an image to understand the impact of increased pretext task complexity on downstream tasks, where $X$ equals 50% and 75%.

![Figure 14 Random channel drop. R, G, and B channel of an image (row 1). A random channel in each pixel is dropped in R, G, and B channels (row 2).](image)

![Figure 15 Example of random channel drop, transformed image (left) and original image (right).](image)
3.3.4 Architecture

Figure 16 shows the proposed architecture. We use UNet as a backbone architecture with AlexNet, VGG16, and Resnet 50 as an encoder. The input to the network would be transformed images, which as shown in Figure 16 are: (a) an image with every alternate pixel dropped, (b) an image with random pixel dropped, and (c) an image with a random channel of a pixel dropped. Each transformation is randomly applied with equal probability to deform the images to feed the network and the original images serve as ground truth labels.

Figure 16 Block diagram of the proposed self-supervised learning method (a) an image where x% of pixels are dropped, (b) an image with every alternate pixel dropped, and (c) every alternate two pixels are dropped and (d) an image where x% of pixel channels are dropped.
3.4 Downstream tasks

The pretext task challenges the network to learn visual features with pseudo-labels that are automatically generated. We hope the network will learn the general-purpose features that capture the salient characteristics of the data. Once the objective is achieved, we use the learned features to transfer the knowledge to a secondary problem. Typically, these downstream tasks are the ones that we are concerned with. This method helps to achieve improvements in performance in the downstream tasks that generally have limited data and overfitting is a big threat. In this work, we evaluate our proposed method with two downstream tasks: classification and semantic segmentation.

3.4.1 Classification

We use the contraction path or encoder section of our model to extract features from our network which is trained in a self-supervised way. We then train a classifier on these feature representations and evaluate our performance on a held-out dataset. We evaluate the performance of our network on PASCAL VOC and ImageNet as per the evaluation procedure defined in [36].

Figure 17 Pipeline for classification task.
3.4.2 Segmentation

For segmentation we apply transfer learning on the pretext network, i.e. we change the last layer to match the number of output classes. We repurpose the network for image segmentation task. Since, both the encoder and decoder are trained in tandem in pretext task, we hope to get better results for the segmentation task.

![A pipeline for segmentation task.](image)

3.5 Loss function

In this section, we step through different loss functions that we considered and discuss the limitations concerning our task.

3.5.1 L2 Loss

L2 norm or mean squared error (MSE) is by far the most widely used metric in the image processing community since it is simple and computationally friendly. However, this metric poorly correlates with human perception w.r.t image quality due to underlying assumptions. First, L2 assumes white gaussian noise which is not well-grounded in general. Second, L2 does not account for the impact of noise on local characteristics of the image but the sensitivity of human perception depends on luminance, contrast, and structure [37].
Also, L2 disproportionately weighs outliers heavily irrespective of the underlying structure due to the squaring of each term which may lead to slower convergence.

\[
L_2 = \frac{1}{N} \sum_{i=1}^{N} \| y_i - \hat{y}_i \|_2^2
\]  

(1)

Where,

- \( N \) is the total number of pixels in an image
- \( i \) is an index of a pixel
- \( y_i \) is the ground truth value of the pixel index \( i \)
- \( \hat{y}_i \) is the predicted value of the pixel index \( i \)

### 3.5.2 L1 Loss

L1 loss helps to overcome the problems with the L2 loss function. L1 does not heavily penalize large outliers and has different convergence properties than L2. L1 preserves colors and luminance but falls short to retain contrast. Also, L1 has demonstrated marginally faster convergence than L2 as shown in [38].

\[
L_1 = \frac{1}{N} \sum_{i=1}^{N} \| y_i - \hat{y}_i \|_1
\]  

(3)

Where,

- \( N \) is the total number of pixels in an image
- \( i \) is an index of a pixel
- \( y_i \) is the ground truth value of the pixel index \( i \)
- \( \hat{y}_i \) is the predicted value of the pixel index \( i \)
3.5.3 SSIM Loss

Structure similarity index often termed as SSIM is a perceptually motivated measure [37]. We use SSIM loss as shown in (4)-(5) to reconstruct images in accordance with human perception.

\[
SSIM(p) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \cdot \frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}
\]

\[
= l(p) \cdot cs(p)
\]

Where,

- \(p\) is pixel
- \(\mu_x\) is mean of \(x\)
- \(u_y\) is mean of \(y\)
- \(\sigma_x\) is the variance of \(x\)
- \(\sigma_y\) is the variance of \(y\)
- \(\sigma_{xy}\) is the covariance of \(x\) and \(y\)
- \(C_1\) is stabilizing variable
- \(C_2\) is stabilizing variable

An image is divided into multiple windows and SSIM is computed in each window. The derivatives cannot be computed on the boundaries because SSIM\((p)\) considers the neighborhood of pixel \(p\) to compute the loss as seen in (4) i.e. to compute mean and variance, we need neighboring pixels which don’t exist on the boundary. On the other hand, L1 and L2 only need the value of the processed and reference pixel. Also, instead of
computing standard deviation and means of pixel $p$, the standard deviation and means are computed with a gaussian filter with a standard deviation $\sigma_G$.

Equation (4) is repurposed to maximize SSIM for center pixels and the convolutional nature allows one to write the loss function as:

$$L^{SSIM}(P) = 1 - SSIM(\hat{p}) \quad (6)$$

Where,

$\hat{p}$ is a center pixel of patch $P$

### 3.5.4 MS-SSIM Loss

$\sigma_G$ value impacts the quality of reconstructed images. Smaller values of $\sigma_G$ do not produce the local structure and larger values of $\sigma_G$ forces the network to preserve noise around the edges. Instead of tuning $\sigma_G$, Zhao et al. [38] introduced a multi-scale version of SSIM (MS-SSIM). Given a pyramid of $M$ levels, MS-SSIM is defined as:

$$MS\cdot SSIM(p) = l^G_M(p) \cdot \prod_{j=1}^{M} c^\beta_j(p) \quad (7)$$

Where,

$p$ is pixel

$M$ is the number of levels in the pyramid

$\alpha$ and $\beta_j$ are set to 1 for $j=\{1,..M\}$
\( l_M \) and \( cs_j \) are like equation 5

Similar to (6), the loss is approximated for patch \( P \) at its center pixel \( \bar{p} \).

\[
\mathcal{L}_{MS-SSIM}(P) = 1 - MS-SSIM(\bar{p})
\] (8)

### 3.5.5 MS-SSIM-L1 Loss

In our experiments, the MS-SSIM loss did reconstruct the images with better quality images compared to L1 and L2. However, in agreement with [38], the MS-SSIM is not sensitive to uniform biases which often leads to a change in brightness or a shift in colors. As an alternative, a combination of L1 and MS-SSIM is used where MS-SSIM preserves the contrast in high-frequency regions, and L1 preserves luminance and colors.

\[
MS-SSIM-L_1 = \lambda(L_1) + (1 - \lambda)MS-SSIM
\] (9)

Where,

\( \lambda \) is a constant and we experiment with 0.5 and 0.75.


Chapter 4 Datasets

In this section, we describe the datasets used in this work. We use two datasets which are used for pretext learning task and downstream tasks by other similar self-supervised learning methods. We use the same train and test splits as used by other works to compare our results with existing work.

4.1 ImageNet

ImageNet is part of ILSVRC (ImageNet Large Scale Visual Recognition Challenge) [7]. It is a benchmark dataset used for object classification and detection. This dataset consists of 1000 object categories and each image consists of one ground-truth label. The dataset consists of 1,281,167 images in the training split and 50,000 images in the test split. Each category has about 732-1300 images. We use this dataset for image reconstruction pretext task without using the labels.

4.2 PascalVOC

The PASCAL VOC 2012 [39] dataset was released as part of the visual object class challenge 2012 for classification, segmentation, and action classification tasks. The main goal of this dataset was to recognize objects in realistic scenes. This dataset consists of twenty different classes and it has 9,993 number of images with pixel-level segmentation. Figure 19 shows an example from the dataset.
Figure 19 Example image and its object segmentation label and class segmentation label from PASCAL VOC [30].
Chapter 5 Experiments and Results

In this section, we will discuss the results of our experiments concerning pretext task and downstream tasks.

5.1 Pretext task Experiments

We train both Resnet50 and VGG16 variants of the network as described in Section 3.3.4. We optimize the models using Adam optimizer with a reduce learning rate on plateau scheme. The initial learning rate of 0.001 is used and weight decay is set to 0.001. To compare our results with other similar work, we apply standard data augmentation techniques of flip, random resized crop, color-jitter, hue, contrast, and saturation. Both variants of the network accept 224 × 224 input image. We use all three methods discussed in Section 3.3 to generate self-supervised labels with equal probability.

We discuss the pretext learning task results for different loss functions in the following sections. We experiment with different values of $\lambda$ and $Drop$. $\lambda$ is a factor with which we take a convex combination of L1 and MSSSIM as per (9) and $Drop$ is a variable that indicate the percentage of pixel values that are dropped using one of the three methods defined in Section 3.3. All models which used convex combination of L1 and MSSSIM as per (9) have been trained for 60 epochs and all other models have been trained for 50 epochs.

5.1.1 Pretext task results with L2 norm

Image reconstruction with L2 norm was reasonable, however, the output image quality was not pleasing for human perception. As shown in Figure 20 where column 1 consists of the original image, column 2 consists of transformed images that serve as input the network, and column 3 consists of the output images generated by the network. For a bed of nails self-supervised label generation method, the quality of reconstruction was good as shown
However, in general, the reconstructed images were noisy and suffer from perceptual quality due to the assumptions of L2 as discussed in Section 3.5.1. We experiment with a different combination of $\lambda$ and $Drop$, but were not able to find a combination that performed well in either the image reconstruction or downstream tasks which can be observed in Tables 1, 2, 3, and 4. When using the L2 norm, we observe poor image quality in the reconstructed image and slower convergence which is in line with results published in [38].

5.1.2 Pretext task results with MS-SSIM

The MS-SSIM is a perceptually motivated measure. With this loss function, we observe significant improvement in image reconstruction quality compared to the L2 norm. Figure 21 shows some example images reconstructed with this loss function. We can notice the MS-SSIM lacks to retain color which led to the use of the L1 norm in combination with MS-SSIM with L1. We also observe some artifacts (box-like structure dividing the images into nine parts) in the output images. We believe this is because the derivatives cannot be computed for some pixels on the boundary region of window $P$ as described in section 3.5.3. A simple fix could be to use padding while computing the loss at the boundary or using overlapping patches.

5.1.3 Pretext task results with MS-SSIM-L1

We use a weighted combination of MS-SSIM and L1 as discussed in section 3.5.5. The L1 does a better job of retaining colors than MS-SSIM. We observe the results of using a weighted combination of MS-SSIM and L1 in Figure 22 where the $Drop$ is set to 50% and $\lambda$ to 0.5. We also experiment with different values of $Drop$ percentage and $\lambda$ as shown in Figures 23 and 24 but we get the best results when $\lambda$ is 0.5 and $Drop$ is set to 50%. With $Drop$ set to 75%, the network hallucinates to reconstruct the image and fails to learn general semantic features due to an increase in complexity of pretext task. We observe similar
artifacts as discussed previously, but the magnitude of the artifacts is less than MS-SSIM since L1 does not have color and tone limitations like MS-SSIM.
Figure 20: Pretext task results with L2 norm (Drop=50%).
Column 1: Original image
Column 2: Input image to the network after transformation
Column 3: Output image (Images are randomly picked).
Figure 21: Pretext task results with MS-SSIM (Drop=50%). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).
Figure 22 Pretext task results with MS-SSIM-L1 (Drop=50%, λ=0.5). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).
<table>
<thead>
<tr>
<th>Original Image</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
<td><img src="image9.png" alt="Image 9" /></td>
</tr>
<tr>
<td><img src="image10.png" alt="Image 10" /></td>
<td><img src="image11.png" alt="Image 11" /></td>
<td><img src="image12.png" alt="Image 12" /></td>
</tr>
</tbody>
</table>

Figure 23: Pretext task results with MS-SSIM-L1 (Drop=50%, λ=0.25). Column 1: Original image Column 2: Input image to the network after transformation Column 3: Output image (Images are randomly picked).
Figure 24: Pretext task results with MS-SSIM-L1 (Drop=75%, $\lambda=0.5$). Column 1: Original image. Column 2: Input image to the network after transformation. Column 3: Output image (Images are randomly picked).
5.2 Downstream tasks

We evaluate our proposed methods using a standard benchmarking framework for self-supervised task proposed by Goyal et al. [36]. We perform classification and segmentation downstream tasks.

5.2.1 Classification

Once the network is trained with a pretext task, we use the encoder to extract features and train a linear classifier on a fixed image representation extracted from the network. The combination of MSSIM and L1 achieves the best performance among our other approaches. Our methods surpass other methods that we compare against.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>75.9</td>
</tr>
<tr>
<td>Rotation [8]</td>
<td>48.9</td>
</tr>
<tr>
<td>Colorization [36]</td>
<td>39.6</td>
</tr>
<tr>
<td>NPID++ [40]</td>
<td>59</td>
</tr>
<tr>
<td>MoCo [41]</td>
<td>60.6</td>
</tr>
<tr>
<td>Jigsaw [36]</td>
<td>45.7</td>
</tr>
<tr>
<td><strong>Ours- MS-SSIM -L1 (Drop=50%, λ=0.5)</strong></td>
<td><strong>60.9</strong></td>
</tr>
<tr>
<td>PIRL [42]*</td>
<td>63.6</td>
</tr>
</tbody>
</table>

Table 1 Classification downstream task results using a linear classifier. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And λ represents the loss weight factor in (9).

*This framework can be applied to any self-supervised method but requires large amounts of computational resources to accommodate many negative samples.

We trained our network with all three proposed self-supervised label generation methods discussed in Section 3.3. One of the proposed methods is randomly applied on the input with equal probability and we experiment with different percentages of pixel or channel...
drop in the input image. We get better results when we drop 50% of pixels/channel (indicated as Drop=50%) than 75% pixel drop. By dropping 75% of the information, we make the pretext task much harder for the network, making it difficult for the network to learn the general features. This would make the network to reconstruct most parts by hallucinating. So, this substantiates that the pretext task needs to have just enough complexity for the network to understand the underlying feature. This hypothesis can be verified by observing Figure 22, with 75% of pixels being dropped the network struggles to reconstruct quality images due to increase complexity.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ImageNet</td>
</tr>
<tr>
<td>Ours-L2 (Drop=50%)</td>
<td>49.2</td>
</tr>
<tr>
<td>Ours-L2 (Drop=75%)</td>
<td>40.3</td>
</tr>
<tr>
<td>Ours- MS-SSIM (Drop=50%)</td>
<td>57.8</td>
</tr>
<tr>
<td>Ours- MS-SSIM (Drop=75%)</td>
<td>56.7</td>
</tr>
<tr>
<td>Ours- MS-SSIM -L1 (Drop=50%, (\lambda=0.75))</td>
<td>58.7</td>
</tr>
<tr>
<td>Ours- MS-SSIM -L1 (Drop=75%, (\lambda=0.5))</td>
<td>59.8</td>
</tr>
</tbody>
</table>

Table 2 Classification downstream task results using a linear classifier. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And \(\lambda\) represents the loss weight factor in (9).

The PIRL framework [42] utilizes jigsaw puzzles as a pretext learning task. This is a framework for self-supervised learning where negative samples are utilized for contrastive learning. PIRL encourages the representation of an image and its transformation to be similar i.e. it forces the image representations to be invariant to image transformations. This framework claims to improve performance for any self-supervised method from its baseline performance but requires a huge computational resource for negative samples. However, it would be interesting to experiment with our proposed method in the PIRL framework in the future.
<table>
<thead>
<tr>
<th>Network</th>
<th>Method</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ImageNet</td>
</tr>
<tr>
<td>AlexNet</td>
<td>Ours-L2 (Drop = 50%)</td>
<td>39.4</td>
</tr>
<tr>
<td></td>
<td>Ours-L2 (Drop = 75%)</td>
<td>36.1</td>
</tr>
<tr>
<td></td>
<td>Ours-MS-SSIM (Drop=50%)</td>
<td>49.8</td>
</tr>
<tr>
<td></td>
<td><strong>Ours-MS-SSIM-L1 (Drop=50%, (\lambda=0.5))</strong></td>
<td><strong>51.4</strong></td>
</tr>
<tr>
<td></td>
<td>Ours-L2 (Drop = 50%)</td>
<td>42.2</td>
</tr>
<tr>
<td></td>
<td>Ours-L2 (Drop = 75%)</td>
<td>39.6</td>
</tr>
<tr>
<td></td>
<td>Ours- MS-SSIM (Drop=50%)</td>
<td>50.2</td>
</tr>
<tr>
<td></td>
<td><strong>Ours- MS-SSIM -L1 (Drop=50%, (\lambda=0.5))</strong></td>
<td><strong>52.1</strong></td>
</tr>
</tbody>
</table>

Table 3 Additional classification results for AlexNet and VGG16 architectures. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And \(\lambda\) represents the weight factor in (9).

We also did some ablation studies by varying Drop and \(\lambda\) for ResNet50 and results are shown in Table 2. We can notice from the results that increasing Drop to 75% decreased the performance indicating the increased complexity in the task. We experiment with \(\lambda\) set to 0.5 and 0.75. When \(\lambda\) is set to zero it will yield us (8) which is equivalent to our MS-SSIM results. The rationale behind not using \(\lambda=0.25\) is this would weight MS-SSIM too much, resulting in similar artifacts as when \(\lambda=0\). Also, L1 norm is not a perceptually motivated measure, setting \(\lambda=1\) would mean to use L1 as our loss function. We think L1 would have given similar or marginally better performance than L2. We observe that weighting L1 and MS-SSIM equally yielded the best results. In addition to ResNet50, we experiment with AlexNet and VGG encoder architectures, results are shown in Table 3. Our results for AlexNet and VGG16 architectures agree with the results in Tables 1 and 2.
5.2.2 Segmentation

Our proposed method is a specialized pretext task which is most suitable for segmentation tasks since the encoder and decoder are trained in tandem. For the segmentation task, the decoder is equally important to an encoder. We outperform all the existing work for the segmentation task as shown in Table 4.

<table>
<thead>
<tr>
<th>Method</th>
<th>PASCAL VOC (mIOU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predict noise [43]</td>
<td>37.1</td>
</tr>
<tr>
<td>Colorization [27]</td>
<td>35.6</td>
</tr>
<tr>
<td>Learning2Count [44]</td>
<td>36.6</td>
</tr>
<tr>
<td>Jigsaw Puzzle [26]</td>
<td>37.6</td>
</tr>
<tr>
<td>Deep Clustering [45]</td>
<td>45.1</td>
</tr>
<tr>
<td>Split-Brain [46]</td>
<td>36</td>
</tr>
<tr>
<td>Ours-L2 (Drop=50%)</td>
<td>38.8</td>
</tr>
<tr>
<td>Ours-L2 (Drop=75%)</td>
<td>34.7</td>
</tr>
<tr>
<td>Ours- MS-SSIM (Drop=50%)</td>
<td>43.5</td>
</tr>
<tr>
<td><strong>Ours- MS-SSIM -L1 (Drop=50%, (\lambda=0.5))</strong></td>
<td><strong>45.4</strong></td>
</tr>
</tbody>
</table>

Table 4 Segmentation downstream task results. The results are reported for all three variants of the encoder network. However, to be fair we compare our method with other methods with AlexNet variant since other methods use AlexNet architecture in their experiments. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in section 3.3. And \(\lambda\) represents the weight factor in (9).

We perform additional experiments with VGG and ResNet50 encoders, results (refer Table 5) are consistent with the AlexNet segmentation results. Even though we get better results with ResNet50, to be fair we compare our AlexNet results with existing work.
Table 5 Additional segmentation results for VGG and ResNet50 architectures. Drop = 50% indicates that 50% of pixels/channel information is dropped using one of the three methods defined in Section 3.3. And λ represents the weight factor in (9).

5.3 Discussion on transfer learning

In general we think applying transfer learning on models learned with self-supervised learning method is more likely to yield better results over supervised learning method since self-supervised learning enables the network to learn features that are often not found in target datasets. Also, self-supervised learning motivates the network to learn general spatial features and likely to handle unforeseen samples better than models learnt only with supervised learning. Deep networks have been shown many times to yield better results with increase in data size. Self-supervised learning enables these models to tap into the virtually unlimited resource of unlabeled data, and then fine-tune models with regular supervised learning on the standard benchmark datasets. Hence, self-supervised learned weights would be a good starting point in most cases. However, in certain situations transfer learning on supervised learning models would yield better results if the end application is subset of the primary application for which the network was trained for. For example, if we train a model for 100 class classification and if we need an another model
to classify 10 classes which is subset of the 100 classes, then it would be appropriate to apply transfer learning on models trained with the supervised learning approach.
Chapter 6 Conclusion

We propose a self-supervised learning method based on image reconstruction specifically targeted for an image segmentation task. Our work highlights the importance of the decoder in segmentation tasks. Existing self-supervised methods do not allow us to train both the encoder and decoder in tandem. Inspired by digital camera color filter array processing, our proposed method overcomes this limitation and is able to outperform existing approaches. We experiment with different loss functions and different pixel drop percentages. The L2 norm suffered due to over penalizing the outliers, and MSSSIM did perform reasonably well in the pretext task but suffered to reproduce colors. We find the linear mixing of L1 and MSSSIM with a pixel drop set to 50% produced the best results. In our experiments, we learned the importance of devising the self-supervised label generation method with the right complexity to enable the network to learn generalized features.
Chapter 7 Future Work

The L1 and MS-SSIM loss function are not invariant to rotation meaning these measures treat an image and its rotation as two different images. It would be beneficial to have a measure that is invariant to image transformations like rotation, translation, and scaling. This can make the network learn underlying semantic features with improved efficiency. To overcome this limitation, the exploration of methods that are invariant to translation, rotation, and scaling, such as the complex wavelet, should be investigated. In the proposed work due to the nature of the UNet architecture, the information flow happens via the encoder and skip connections from the contraction path to the expansion path. We believe potential improvements can be achieved for the classification task by forcing the information flow via the bottleneck layer. It would also be interesting to extend this work for object detection since it is a popular computer vision task.
References


