Self Supervision Techniques in CNNs

Varsha S (193079005)
Department of Electrical Engineering
IIT Bombay

***Abstract*—The problem of image representation learning without human annotation is presented. By following the principles of self-supervision, a convolutional neural network (CNN) is built that can be trained to solve pretext task including Jigsaw puzzles and Inpainting, which requires no manual labeling, and then later repurposed to solve object classification and detection. To maintain the compatibility across tasks a the context-free network (CFN), a siamese-ennead CNN is studied. The CFN takes image tiles as input and explicitly limits the receptive field (or context) of its early processing units to one tile at a time. It is seen that the CFN includes fewer parameters than AlexNet while preserving the same semantic learning capabilities. By analogy with auto-encoders, Context Encoders is studied – a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. By training the CFN to solve Jigsaw puzzles, and Context Encoder to produce a plausible hypothesis for the missing part(s) in the Inpainting problem, a feature mapping of object parts as well as their correct spatial arrangement and content of the entire image is learnt.**

I. Introduction

Recently, new computer vision methods have relied on large datasets of millions of labeled examples to learn rich, high-performance visual representations [5]. Yet efforts to scale these methods to truly Internet-scale datasets (i.e. hundreds of billions of images) are hampered by the labour intensive annotation generation. A natural way to address this difficulty would be to employ unsupervised learning, which aims to use data without any annotation. Unfortunately, despite several decades of sustained effort, unsupervised methods have not yet been shown to extract useful information from large collections of full-sized, real images. After all, without labels, it is not even clear what should be represented.

The visual world is very diverse, yet highly structured, and humans have an uncanny ability to make sense of this structure. It is explored whether state-of-the-art computer vision algorithms can do the same. Consider the image shown in Figure 1a. Although the center part of the image is missing, most of us can easily imagine and even draw its content from the surrounding pixels, without having ever seen that exact scene. This ability comes from the fact that natural images, despite their diversity, are highly structured (e.g. the regular pattern of windows on the facade). In this work, it is possible to learn and predict this structure using convolutional neural networks (CNNs), a class of models that have recently shown success across a variety of image understanding tasks [1].

Recently, Doersch et al. [3], have explored a novel paradigm for unsupervised learning called self-supervised learning. The main idea is to exploit different labelings that are freely available besides or within visual data, and to use them as intrinsic reward signals to learn general-purpose features.

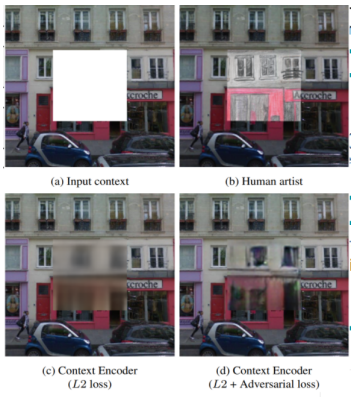

Fig. 1: Image Inpainting

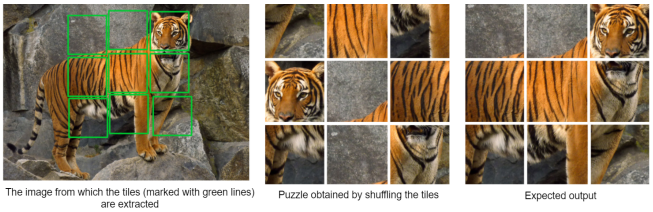

Fig. 2: Jigsaw Puzzle

The features obtained with these approaches have been successfully transferred to classification and detection tasks, and their performance is very encouraging when compared to features trained in a supervised manner. While it is true that biological agents typically make use of multiple images and also integrate additional sensory information, such as ego-motion, it is also true that single snapshots may carry more information than extracted so far. This work shows that this is indeed the case. A novel self-supervised task is introduced, the Jigsaw puzzle reassembly problem (see Figure. 2), which builds features that yield high performance when transferred to detection and classification tasks. The association of each separate puzzle tile to a precise object part might be ambiguous. However, when all the tiles are observed, the ambiguities might be eliminated more easily because the tile placement is mutually exclusive.

II. INPAINTING

Given an image with a missing region (e.g., Figure. 1a), a convolutional neural network is trained to regress to the

missing pixel values (Figure. 1d). and the model is called context encoder, as it consists of an encoder capturing the context of an image into a compact latent feature representation and a decoder which uses that representation to produce the missing image content.

The context encoder is closely related to autoencoders [6], as it shares a similar encoder-decoder architecture. Autoencoders take an input image and try to reconstruct it after it passes through a low-dimensional "bottleneck" layer, with the aim of obtaining a compact feature representation of the scene. But the context encoder needs to solve a much harder task: to fill in large missing areas of the image, where it can't get "hints" from nearby pixels. This requires a much deeper semantic understanding of the scene, and the ability to synthesize high-level features over large spatial extents. It is demonstrated that in order to succeed at this task, a model needs to both understand the content of an image, as well as produce a plausible hypothesis for the missing parts. This task, however, is inherently multi-modal as there are multiple ways to fill the missing region while also maintaining coherence with the given context.

The encoder and the decoder are evaluated independently. On the encoder side, it is showed that encoding just the context of an image patch and using the resulting feature to retrieve nearest neighbor contexts from a dataset produces patches which are semantically similar to the original (unseen) patch. On the decoder side, it is showed that the network is often able to fill in realistic image content with the help of a joint loss function.

A. Related Work

Most closely related to the present paper are efforts at exploiting spatial context as a source of free and plentiful supervisory signal. Visual Memex [8] used context to nonparametrically model object relations and to predict masked objects in scenes, while [7] used context to establish correspondences for unsupervised object discovery. However, both approaches relied on hand-designed features and did not perform any representation learning.

Denoising autoencoders [14] reconstruct the image from local corruptions, to make encoding robust to such corruptions. While context encoders could be thought of as a variant of denoising autoencoders, the corruption applied to the model's input is spatially much larger, requiring more semantic information to undo.

Though the paper shares the same high-level goals with Doersch et al. [3] it fundamentally differ in the approach: while the former are solving a discriminative task (is patch A above patch B or below?), the context encoder solves a pure prediction problem (what pixel intensities should go in the hole?).

Recently, Radford et al. [9] proposed new convolutional architectures and optimization hyperparameters for Generative Adversarial Networks (GAN) producing encouraging results. The context encoders are trained using an adversary jointly with reconstruction loss for generating inpainting results.

Previous scene completion relies on a hand-crafted distance metric, such as Gist [14] for nearest-neighbor computation which is inferior to a learned distance metric. Also it won't perform well when the missing region is too large for non-semantic methods [10] to work well. The method proposed is often able to inpaint semantically meaningful content in a parametric fashion, as well as provide a better feature for nearest neighbor-based inpainting methods.

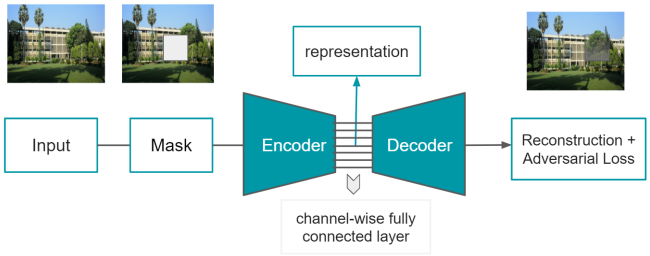

Fig. 3: Context Encoder structure with Channel wise Fully Connected Layers

B. Encoder-decoder pipeline

The encoder takes an input image with missing regions and produces a latent feature representation of that image. The decoder takes this feature representation and produces the missing image content.

Encoder - It is derived from the AlexNet architecture [12]. Given an input image of size 227×227, the first five convolutional layers and the following pooling layer (called pool5) are used to compute an abstract $6 \times 6 \times 256$ dimensional feature representation. The latent feature dimension is $6 \times 6 \times 256 = 9216$ for both encoder and decoder.

Channel-wise fully-connected layer - This layer is essentially a fully-connected layer with groups, intended to propagate information within activations of each feature map. However, unlike a fully-connected layer, it has no parameters connecting different feature maps and only propagates information within feature maps.

Decoder - The "encoder features" are connected to the "decoder features" using a channel-wise fully connected layer. The channel-wise fully-connected layer is followed by a series of five up-convolutional layers with learned filters, each with a rectified linear unit (ReLU) activation function. A up-convolutional is simply a convolution that results in a higher resolution image. The intuition behind this is straightforward – the series of up-convolutions and non-linearities comprises a non-linear weighted upsampling of the feature produced by the encoder until it roughly reaches the original target size.

C. Loss Function

There are often multiple equally plausible ways to fill a missing image region which are consistent with the context. This behavior is modeled by having a decoupled joint loss function to handle both continuity within the context and multiple modes in the output.

Reconstruction Loss - While this simple loss encourages

the decoder to produce a rough outline of the predicted object, it often fails to capture any high frequency detail (see Figure. 1c). This stems from the fact that the L2 (or L1) loss often prefer a blurry solution, over highly accurate textures.

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

Adversarial Loss - The adversarial loss for context encoders,

$$\mathcal{L}_{adv} = \max_D \quad \mathbb{E}_{x \in \mathcal{X}}[\log(D(x)) + \log(1 - D(F((1 - \hat{M}) \odot x)))]$$

where, in practice, both F and D are optimized jointly using alternating SGD. Note that this objective encourages the entire output of the context encoder to look realistic, not just the missing regions as in Joint loss.

Joint Loss - The overall loss function is

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}$$

D. Region Masks

The input to a context encoder is an image with one or more of its regions "dropped out"; i.e., set to zero, assuming zero-centered inputs.

1) Central region - The simplest such shape is the central square patch in the image, as shown in Figure 4. While this works quite well for inpainting, the network learns low level image features that latch onto the boundary of the central mask. Those low level image features tend not to generalize well to images without masks, hence the features learned are not very general.
2) Random Blocks - To prevent the network from latching on the the constant boundary of the masked region, the masking process is randomized. Instead of choosing a single large mask at a fixed location, a number of smaller possibly overlapping masks are removed, covering up to 1/4 of the image. An example of this is shown in Figure 4. However, the random block masking still has sharp boundaries convolutional features could latch onto.
3) Random region - To completely remove those boundaries, arbitrary shapes from images are removed. The region masking process is randomized, and no correlation between the source segmentation mask and the image exists. Those regions prevent the network from learning low-level features corresponding to the removed mask.

E. Implementation Pipeline

The pipeline was implemented in Caffe and Torch. Stochastic gradient descent solver, ADAM [13] for optimization. The missing region in the masked input image is filled with constant mean value.

Feature Learning - Unfortunately, the adversarial loss did not converge with AlexNet, so just the reconstruction loss is used. The networks were trained with a constant learning rate of 0.001 for the center-region masks.

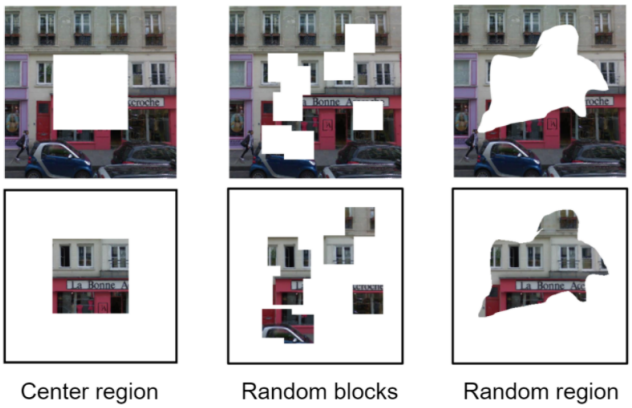

Fig. 4: Three kinds of masks

III. JIGSAW

Jigsaw puzzles have been associated with learning since their inception. Studies in Psychonomic show that Jigsaw puzzles can be used to assess visuospatial processing in humans [16]. Indeed, the Hooper Visual Organization Test [15] is routinely used to measures an individual's ability to organize visual stimuli. This test uses puzzles with line drawings of simple objects and requires the patient to recognize the object without moving the tiles. Instead of using Jigsaw puzzles to assess someone's visuospatial processing ability, it is proposed to use Jigsaw puzzles to develop a visuospatial representation of objects in the context of CNNs.

A. Related Work

Doersch et al. [3] train a convolutional network to classify the relative position between two image patches. One tile is kept in the middle of a 3 × 3 grid and the other tile can be placed in any of the other 8 available locations (up to some small random shift). In Figure. 2 (c) the relative location between the central tile and the top-left and top-middle tiles is ambiguous. In contrast, the Jigsaw puzzle problem is solved by observing all the tiles at the same time. This allows the trained network to intersect all ambiguity sets and possibly reduce them to a singleton.

Agrawal et al. [17] exploits easily available labeling (egomotion) provided by other sensors. They show that egomotion is a useful supervisory signal when learning features. They train a siamese network to estimate egomotion from two image frames and compare it to the egomotion measured with odometry sensors. However, because the object identity is the same in both images, the intraclass variability may be limited. With two images of the same instance, learned features focus on their similarities (such as color and texture) rather than their high-level structure. In contrast, the Jigsaw puzzle approach ignores similarities between tiles (such as color and texture), as they do not help their localization, and focuses instead on their differences.

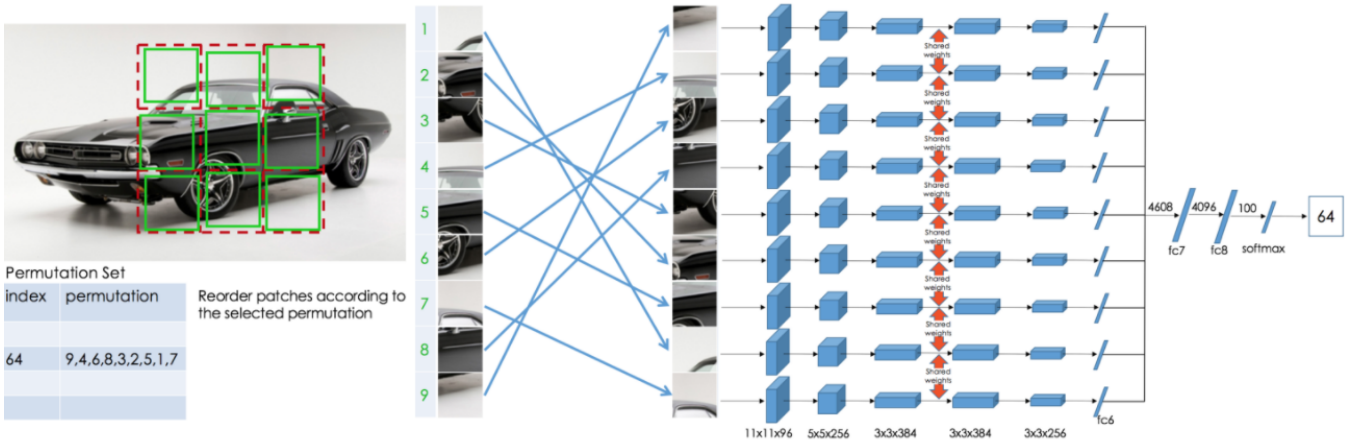

Fig. 5: Context Free Network. The figure illustrates how a puzzle is generated and solved. A 225 × 225 pixel window is cropped from an image (red dashed box), divides it into a 3 × 3 grid, and randomly picked a 64 × 64 pixel tiles from each 75 × 75 pixel cell. These 9 tiles are reordered via a randomly chosen permutation from a predefined permutation set and are then fed to the CFN. The task is to predict the index of the chosen permutation (technically, output is defined as a probability vector with 1 at the 64-th location and 0 elsewhere). The CFN is a siamese-ennead CNN.

B. Solving Jigsaw Puzzles

Low-level statistics, such as similar structural patterns and texture close to the boundaries of the tile, are simple cues that humans actually use to solve Jigsaw puzzles. However, solving a Jigsaw puzzle based on these cues does not require any understanding of the global object. Thus, here the network delays the computation of statistics across different tiles (see Figure. 5). The network first computes features based only on the pixels within each tile (one row in Figure. 5). Then, it finds the parts arrangement just by using these features (last fully connected layers in Figure. 5). The objective is to force the network to learn features that are as representative and discriminative as possible of each object part for the purpose of determining their relative location.

C. Context Free Network

A siamese-ennead convolutional network (see Figure. 5) is built, where each row up to the first fully connected layer (fc6) uses the AlexNet architecture [12] with shared weights. The architecture is called context-free network (CFN) because the data flow of each patch is explicitly separated until the fully connected layer and context is handled only in the last fully connected layers. In this test the input images are resized to 225 × 225 pixels, split them into a 3 × 3 grid and then fed the full 75 × 75 tiles to the network.

This network can thus be used interchangeably for different tasks including detection and classification.

D. The Jigsaw Puzzle Task

To train the CFN a set of Jigsaw puzzle permutations is defined, e.g., a tile configuration S = (3, 1, 2, 9, 5, 4, 8, 7, 6), and assign an index to each entry. One such permutation is randomly picked to rearrange the 9 input patches according that permutation, and the CFN has to return a vector with the probability value for each index. Given 9 tiles, there are 9! = 362,880 possible permutations.

E. Training the CFN

The output of the CFN can be seen as the conditional probability density function (pdf) of the spatial arrangement of object parts (or scene parts) in a part based model,

$$p(S|A_1, A_2, \ldots, A_9) = p(S|F_1, F_2, \ldots, F_9) \prod_{i=1}^{9} p(F_i|A_i)$$

where S is the configuration of the tiles, A_i is the i-th part appearance of the object, and $\{F_i\}_{\{i=1,..,9\}}$ form the intermediate feature representation. Our objective is to train the CFN so that the features F_i have semantic attributes that can identify the relative position between parts.

More in general, a self-supervised learning system might lead to representations that are suitable to solve the pre-text task, but not the target tasks, e.g., object classification, detection, and segmentation. In this regard, an important factor to learn better representations is to prevent the model from taking these undesirable solutions. These solutions are called *shortcuts*. Other shortcuts that the model can use to solve the Jigsaw puzzle task include exploiting low-level statistics, such as edge continuity, the pixel intensity/color distribution.

To avoid shortcuts multiple techniques are employed.

- To prevent mapping the appearance to an absolute position multiple Jigsaw puzzles of the same image are fed to the CFN (an average of 69 out of 1000 possible puzzle configurations) and make sure that the tiles are shuffled as much as possible by choosing configurations with sufficiently large average Hamming distance. The

permutation set controls the ambiguity of the task. If the permutations are close to each other, the Jigsaw puzzle task is more challenging and ambiguous.

- To avoid shortcuts due to edge continuity and pixel intensity distribution a random gap is left between the tiles. This discourages the CFN from learning low-level statistics and was also done in [3]. 64×64 pixel tiles are randomly selected from the 85×85 pixel cells.This allows us to have a 21 pixel gap between tiles.
- To avoid shortcuts due to chromatic aberration the color channels the color channels of the color images of each tile are jittered randomly by ±0; ±1; ±2 pixels. Grayscale images are used with a ratio of 30% to 70%).Chromatic aberration is a relative spatial shift between color channels that increases from the images center to the borders. This type of distortion helps the network to estimate the tile positions.
- To avoid shortcuts due to adjacent patches including similar low-level statistics like the mean and standard deviation of the pixel intensities(which allows the model to find the arrangement of the patches), normalization of the mean and the standard deviation of each patch is done independently.

F. Implementation

Stochastic gradient descent without batch normalization [18] is used on one Titan X GPU. The training uses 1.3M color images of 256×256 pixels and minibatches with a batch size of 256 images. The images are resized by preserving the aspect ratio until either the height or the width matches 256 pixels. Then the other dimension is cropped to 256 pixels. Average of 69 Jigsaw puzzles per image is solved.

IV. Discussion

- I implemented the Feature learning with Inpainting [1] from the scratch on Pytorch with AlexNet as the backbone architecture. The paris-street-view dataset (as used by the authors) was used to train the network. The network still is not reproducing the results in the paper. The output image is severely distorted and needs further debugging to figure out the issue. Here is the link to colab notebook - Inpainting Notebook
- I also implemented paper [2] to solve jigsaw puzzles on Pytorch with Alexnet as the backbone architecture. The work is in progress. Here is the link to colab notebook - Jigsaw Puzzle Notebook
- To meet the goal which is to compare the performance of the two unique pretext tasks on image classification, further work must be carried out.

References

[1] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell and Alexei A. Efros *Context Encoders: Feature Learning by Inpainting*. http://arxiv.org/abs/1604.07379

[2] Mehdi Noroozi and Paolo Favaro *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles* http://arxiv.org/abs/1603.09246.pdf

[3] Carl Doersch, Abhinav Gupta, and Alexei A. Efros *Unsupervised Visual Representation Learning by Context Prediction. In ICCV 2015* https://arxiv.org/pdf/1505.05192.pdf

[4] Github Code Reference https://github.com/bbrattoli/JigsawPuzzlePytorch

[5] A. Krizhevsky, I. Sutskever, and G. Hinton *Imagenet classification with deep convolutional neural networks. In NIPS, 2012*

[6] Y. Bengio. *Learning deep architectures for ai. Foundations and trends in Machine Learning, 2009.*

[7] C. Doersch, A. Gupta, and A. A. Efros. *Context as supervisory signal: Discovering objects with predictable context. In ECCV, 2014*

[8] T. Malisiewicz and A. Efros. *Beyond categories: The visual memex model for reasoning about object relationships. In NIPS, 2009*

[9] A. Radford, L. Metz, and S. Chintala. *Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR, 2016*

[10] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. *Image inpainting. In Computer graphics and interactive techniques, 2000*

[11] A. Oliva and A. Torralba. *Building the gist of a scene: The role of global image features in recognition. Progress in brain research, 2006*

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks. In NIPS, 2012*

[13] D. Kingma and J. Ba. Adam *A method for stochastic optimization. ICLR, 2015.*

[14] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol.*Extracting and composing robust features with denoising autoencoders. In ICML, 2008*

[15] Hooper, H. *The Hooper Visual Organization Test. Western Psychological Services, Los Angeles, CA (1983)*

[16] Richardson, J., Vecchi, T. *A jigsaw-puzzle imagery task for assessing active visuospatial processes in old and young people. Behavior Research Methods, Instruments, Computers (2002)*

[17] Agrawal, P., Carreira, J., Malik, J. *Learning to see by moving. ICCV (2015)*

[18] Ioffe, S., Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML (2015)