Visual Word Embedding: A Self-Supervised Learning Approach for Image Embeddings

Mahadev Prasad Panda

September 8, 2022

Abstract

Self-supervised learning approaches can lessen the need for large-scale labeled datasets when using deep learning techniques to solve computer vision-related tasks. In this work, we propose a self-supervised learning method - "Visual Word Embeddings". We divide our work into two parts. In the first part, we develop an asymmetric encoder and decoder architecture. Our approach uses a towered architecture for the encoder and a straightforward convolutional model for the decoder, which reconstructs the target from the latent representation of the context received from the encoder. In the second part, we assess the performance of the model using the linear evaluation approach. For this, we use the learned representation to perform a downstream image classification task. Finally, we compare the performance of our model with that of a baseline simple autoencoder.

1 Introduction

With the fast improvement of computing hardware like GPUs deep learning-based models saw rapid growth in their ability and proficiency to perform computer vision applications like image classification [1, 2, 3], object detection [4, 5], segmentation [6, 7, 8], etc. To perform these vision-related tasks to a satisfactory level, the deep learning models in general require a large amount of leveled data. But to collect and annotate a large-scale image dataset is a time-consuming and expensive task that involves human capital.

To avoid this problem different methods based on self-supervised learning have been proposed [9]. These methods try to learn the visual features of images from an unlabeled dataset by self-generating a supervisory signal using the implicit information in the dataset. For this, a general approach is to define a surrogate task or a pretext task and let the model learn generalized features by training on the objective function of the surrogate task. Many surrogate tasks such as image colorization [10, 11], image jigsaw puzzle-solving [12, 13], image inpainting [14, 15, 16, 17], or some combination of these [18] have been proposed. After the model is trained, the learned features are used in downstream tasks such as image classification, image retrieval, segmentation, etc.

Self-supervised methods have greatly helped in the field of natural language processing (NLP) by mitigating the need for a large labeled dataset. Popular models like the GPT- autoregressive model [**?**] or the BERT - masked encoding model [19] can be used to generate very generalized word embeddings. These models learn by removing a portion of the training data and predicting the removed part. The popularity of these methods has generated significant interest in transferring these concepts from the NLP domain to the vision domain. But the progress is less.

For our work, we draw inspiration from the word embedding methods. Word embeddings learn the real value vector representation of words. Each word of the corpus is associated with a vector and the value of the vector is often learned using a deep learning method. There are several methods used for getting word embeddings. In our work, we try to transfer one of the concepts of word2vec [20], i.e., Continuous Bag-of-Words, or CBOW model to the visual field. Like the CBOW method, we define a target and its context and we try to predict the target from its context. In doing so we learn the required embeddings.

To learn the embeddings we use a deep neural network model with an encoder-decoder architecture. Our model has an encoder and a decoder. The encoder takes the context information and creates a lower-dimensional representation of the context. The decoder takes this latent representation and tries to recreate the target.

We have divided our report as follows. In the next section, we discuss the related previous works, followed by methodology where we discuss our method and model in detail. We expand on how we have defined the context and target in the case of an image. After this, we discuss our experiments and their results. Finally, we discuss the conclusion of our work and propose some future works.

2 Related Work

Our task comes under the self-supervised learning domain. Surrogate tasks are used in self-supervised learning to learn representations from the unlabeled data. Tasks like Image inpainting, solving image jigsaw puzzles, or image colorization are used to self-generate a supervised signal for the training of the model.

Autoencoding is a classical method that takes an image and tries to reconstruct the input image thereby learning representations. The encoder creates a mapping of the input to a latent representation and the decoder takes the latent representation to recreate the input [21]. Extensive research on autoencoders has led to models like denoising autoencoder (DAE) [22], variational autoencoder(VAE) [23], etc. Our model architecture loosely follows the structure of the autoencoder.

Colorization is a task of predicting color given the intensity component of the image. Image colorization is a pretext training task in self-supervised learning and

was first introduced by Zhang et al [10]. Their network acted as cross channel encoder which learns representations by predicting ab channels given L channel. Larsson et al also explored a similar approach for a detailed study showing the supervisory signal obtained from colorization is as strong as various ImageNet pre-training methods [11].

Image Inpainting uses the missing image region to create a supervisory signal. To achieve this typically a portion of the image is removed and the model learns by creating a realistic and semantically correct image from the incomplete image. Pathak et al in their proposed context encoder used a reconstruction loss and an adversarial loss to create a realistic and sharp reconstructed image in an autoencoder type model [14]. Generative models like GAN along with attention have been used to do image inpainting [15] [16]. He et al used vision transformers to create an encoder-decoder architecture that takes images with masked random patches and tries to reconstruct the original image [17]. They found that a high proportion of masking makes the model learn more useful features for a downstream task.

Image Jigsaw Puzzle refers to solving the image puzzle either by predicting the relative position of patches or by predicting the correct order of the shuffled sequence of patches. Doersch et al did pioneering work in this field. In their method, two random patches of an image are taken and the model is trained to predict the relative position of these two patches of the image [12]. The method of using spatial context like the relative position of image patches in the pretext task was further explored by Noroozi et al. An image is divided into nine patches and shuffled. The model is trained to take shuffled patches and learn to predict the correct order of the spatial locations of the input patches [13]. Kim et al combined all three methods i.e, colorization, inpainting, and jigsaw puzzle into a single pretext task where a model is trained to solve a damaged jigsaw puzzle [18].

3 Methodology

3.1 Data Preparation

To transfer the concept of CBOW to the vision domain we prepare the data in the following way. First, we divide the image into a 4x4 grid, i.e, an image is converted into sixteen patches. For our method, each patch becomes the target patch and the 8 neighboring patches become its context. For the border patches, the number of neighbors is less than eight. To compensate for this, we take patches with zeros as the pixel values, which we name zero patches as shown in figure 1 (c). The number of zero patches depends on the number of missing neighbors. We also apply three image augmentation techniques such as random cropping, random rotation, and color jitter to the original image before we divide the image into patches. For our experimentation, we use the CIFAR-100 dataset, which contains 60000, 32x32 images.

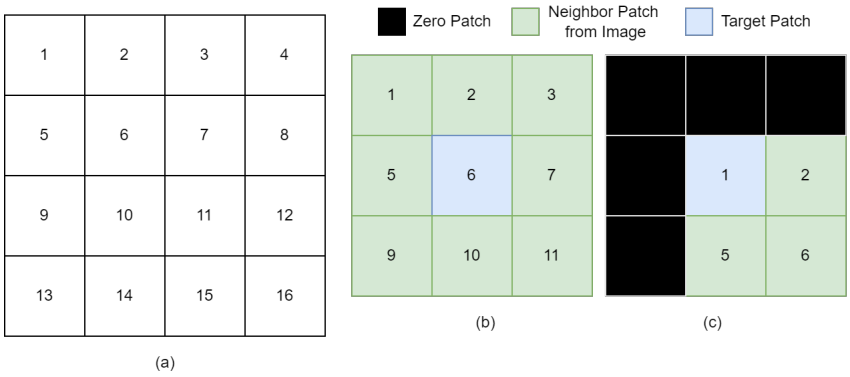

Figure 1: (a) Schematic of an image divided into 4x4 grid, (b) and (c) example of target patches and along with its context which is a combination of zero patches and neighbor patches.

3.2 Model

3.2.1 Architecture

We build an asymmetric encoder-decoder architecture for our model. Figure 2 shows the schematics of our model. Our encoder contains a single tower. Eight neighbor patches going into the encoder go to the tower and the tower maps eight neighbor patches to eight lower-dimensional representations. Finally, the eight lower-dimensional representations are combined through concatenation forming the embedding of a single patch which is the output of the encoder. The tower in the encoder follows a Resnet-32 architecture as used by [24].

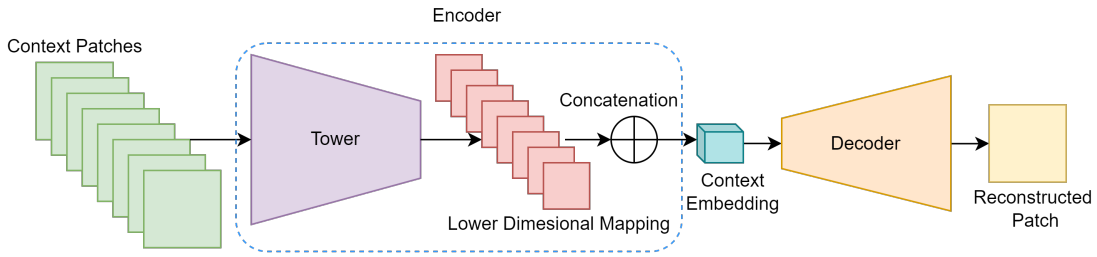

Figure 2: Schematic diagram of the model architecture.

The decoder takes the patch embedding and tries to reconstruct the target patch. We use a simple decoder in our model as shown in figure 3. The decoder has a linear block that contains two fully connected layers with the relu activation function. After the linear layer, there is a reshaping layer. The reshaping layer is followed by the upsampling block. The upsampling block contains an upsampling layer with a scale factor of two and three convolution layers. We use the relu activation function in the convolution layers and the sigmoid activation function in the output layer.

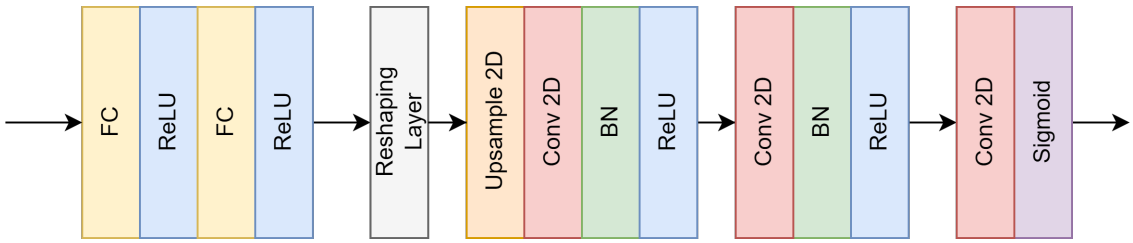

Figure 3: Schematic diagram of decoder architecture.

3.2.2 Loss

We use a pixel-wise L2 reconstruction loss for our loss function, which calculates the loss value between the target patch and the reconstructed patch from the context by our model. The loss function is given below,

$$\mathcal{L}(x) = \|x - D(E(x))\|_2^2 \tag{1}$$

where x stands for groung truth or target image patch, $E(x)$ is the output of the encoder and $D(E(x))$ is the output of the decoder.

4 Experimental Setup

We divide our experimental setup into two parts. In the first part, we train the model to learn features by performing the surrogate or upstream task. In the second part, we evaluate our model using the linear evaluation method proposed by [24].

4.1 Upstream Task Setup

For the upstream task, we train the model for 200 epochs to be consistent with the method suggested in [24]. We use the Adam optimizer with a learning rate of 0.0001 and a batch size of 512. We experimented with and without image augmentation. In the case of image augmentation, we use three methods such as color jitter, random rotation, and random cropping.

4.2 Downstream Task Setup

We perform the linear evaluation in the downstream task to evaluate the quality of the features learned in the upstream task. As our image is divided into 16 patches, we first collect the patch embeddings from the encoder using the already trained weights. Then we concatenate all the patch embedding to get an image embedding. The image embedding goes to the single linear layer that performs a classification task. We train this linear layer only in the linear evaluation step. We use Adam optimizer with its default settings and a batch size of 128. Figure 4 shows a schematic diagram of the linear evaluation process. The test accuracy obtained in the linear evaluation acts as a measure of the quality of the learned representations in the pretext task.

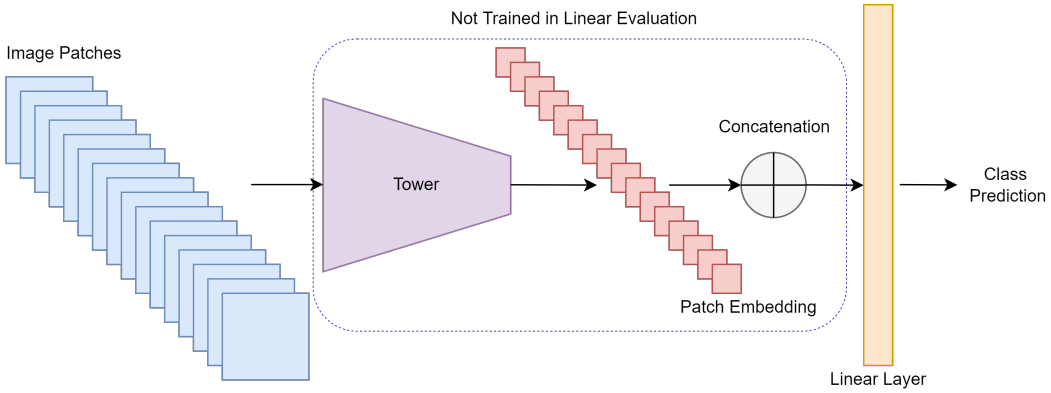

Figure 4: Schematic diagram of the linear evaluation process.

5 Results

We trained two models. The first model is a simple autoencoder that takes a patch and tries to recreate it. The second model is the context patch autoencoder that takes the context of the patch and tries to recreate the patch. In both the models all parameters like encoder-decoder architecture, batch size, learning rate, optimizer, loss function, etc. are kept the same as mentioned in the upstream task setup. Both the models are trained for 200 epochs.

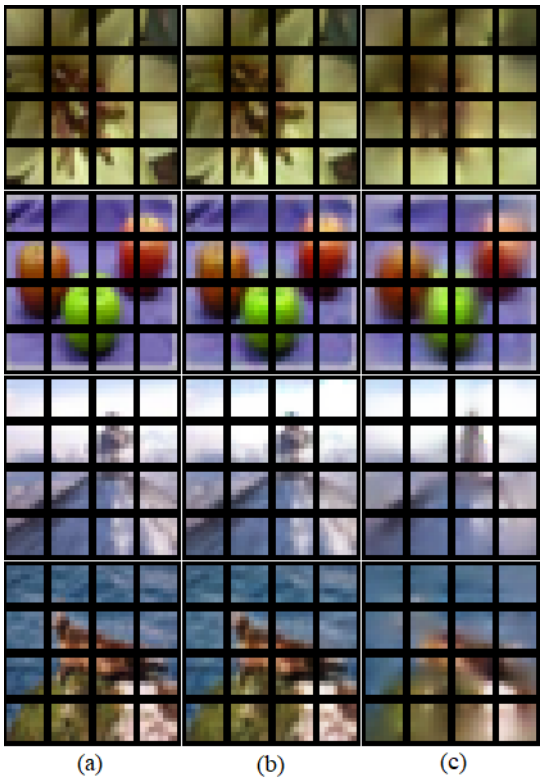

Figure 5: Column (a) shows the ground truth. Column (b) and column (c) show the reconstructions from the simple autoencoder and context patch autoencoder respectively.

Figure 5 shows the reconstructions from these two models along with the ground truth. We can see that the simple autoencoder reconstructs better patches compared to the context patch auto encoder which is expected as the reconstruction task is much tougher while reconstructing from the neighboring patches compared to the direct reconstruction from the target patch as in the case of the simple autoencoder.

Next, we report the accuracies obtained from the linear evaluation for different models in table 1. We find that the context patch autoencoder obtains higher accuracies compared to the simple autoencoder even though the reconstructions from the simple autoencoder are better. One of the reasons for this might be the fact that spatial redundancies are much higher in the case of images and the simple autoencoder model does not need a deeper understanding level to reconstruct the patch from the patch itself. Whereas in the case of the context patch autoencoder reconstruction of the target patch from the context patches is a more difficult task which results in a deeper and more generalized understanding of images.

Method Name	Backbone	Augme--ntation	Linear Evaluation CIFAR-100
Supervised (upper bound) [24]	–	–	65.32
Random Weights (lower bound) [24]	–	–	7.65
Relational Reasoning [24]	✗	✓	46.17
Simple Autoencoder	✗	✓	17.81
Context Patch Autoencoder	Resnet-32	✗	30.46
Context Patch Autoencoder	*Resnet-32*	✓	*34.08*
Context Patch Autoencoder	Conv4	✓	24.55
SimCLR [24, 25]	Resnet-50	✓	42.13
RotationNet [24]	—	—	29.02
Deep InfoMax [24]	—	—	24.07
DeepCluster [24]	—	—	20.44

Table 1: Comparison of results for linear evaluation accuracy in percentage.

We experimented with two different tower architectures, i.e., a simple convolutional and Resnet 32 for our encoder. Our tower architectures are the same as the backbone network used in [24]. We obtained that the Resent 32 tower performs better compared to the convolutional tower for CIFAR 100 dataset in our method also. We also experimented with image augmentation. In our case with three augmentation techniques, we saw a three percent increase in linear evaluation test accuracy.

6 Conclusions

Even though context patch autoencoder shows promising potential to be a self-supervised learning approach but we fail to beat the current state-of-the-art [24].

We observed the reconstruction of patches from their context helps in learning more generalized features. In addition to this image augmentation methods further, improve the quality of the learned features. The quality of learned representation might improve more with the addition of adversarial loss as it has helped to create sharper reconstruction in image inpainting cases. Furthermore, a different tower architecture may be using transformers and attention mechanisms could help in learning even better representations.

References

[1] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," *arXiv preprint arXiv:2010.01412*, 2020.

[2] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] Y. Wei, H. Hu, Z. Xie, Z. Zhang, Y. Cao, J. Bao, D. Chen, and B. Guo, "Contrastive learning rivals masked image modeling in fine-tuning via feature distillation," *arXiv preprint arXiv:2205.14141*, 2022.

[5] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," *arXiv preprint arXiv:2203.03605*, 2022.

[6] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2918–2928.

[7] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som *et al.*, "Image as a foreign language: Beit pretraining for all vision and vision-language tasks," *arXiv preprint arXiv:2208.10442*, 2022.

[8] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao, "Vision transformer adapter for dense predictions," *arXiv preprint arXiv:2205.08534*, 2022.

[9] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[10] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[11] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6874–6883.

[12] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[13] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision*. Springer, 2016, pp. 69–84.

[14] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[15] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5505–5514.

[16] ——, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4471–4480.

[17] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[18] D. Kim, D. Cho, D. Yoo, and I. S. Kweon, "Learning image representations by completing damaged jigsaw puzzles," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 793–802.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[21] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[22] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[24] M. Patacchiola and A. J. Storkey, "Self-supervised relational reasoning for representation learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4003–4014, 2020.

[25] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.