DV^2: Double View Network with Dynamic Vocabulary

Haokun Liu
hl3236@nyu.edu

Kawshik Kannan
kk4161@nyu.edu

Rutvik S. Shah
rss638@nyu.edu

Dept. of Computer Science
New York University

Abstract

We explore a set of adaptations to patch-prediction self-supervised learning task. Our main goals are providing additional positive and negative patch samples to strengthen patch prediction task, and bridge the gap between supervised and self supervised learning methods. To achieve this, we propose a novel method called Dynamic Vocabulary to retrieve additional negative patches, a hybrid architecture that add self-attention to existing convolution network to exchange information between patches, and design training objective to pull network outputs from full image and patches together. We compare our method with the baselines, perform ablation study on each individual adaptation, and additional experiments on the details of dynamic vocabulary, downstream supervised training. Our best model achieves 78.6% accuracy in the STL-10 5k task.

Last update: 12/21/2019

1. Introduction

Recently, self-supervised learning has achieved striking progresses across multiple areas of machine learning ([2],[9],[18],[24],[14],[25],[26]) Predicting or retrieving a missing patch from a set of candidate patches is a widely used self-supervised task ([6],[11],[19]).

If we consider tokens to be the counterpart of patches, predicting patches has been widely successful in text [5, 27]. However, when applied to images, predicting patches can be hampered by a few key differences.

First, unlike tokens, which are finite (when using subword tokenization), patches are naturally unique. No two patches are exactly the same, there are always minutiae to tell them apart. Contrasting between the original patch and a negative patch with the same meaning would only encourage the model to focus on details with little semantic value and memorize images.

Secondly, tokens being the lowest level features of text processing, can be directly attached to embedding vectors, while the representation of patches can only be produced from pixel level, usually through a few convolution layers. In doing so, features across patches that could be captured when using full image are broken down which can no longer be learnt by the self-supervised training. As the patch network goes deeper, more layers of parameters can be pretrained, but a larger gap between the receptive field of pretrain and supervised-training also emerges. Any architecture that take patches as input has to weigh in this trade-off.

In this work, we explore a set of adaptations to masked patch predictions to tackle these two problems. We propose a new method for producing comprehensive, stable negative patches at affordable computation cost and an adaptor module that can be inserted into existing architectures and corresponding training objective to encourage a network to produce similar features from both complete image and patches. Figure 3 shows an overview of our full model.

We test the effectiveness (or more precisely speaking, ineffectiveness) of these adaptations in our experiments. We also perform ablation studies to show the effect of each individual adaptation and empirical comparison between detailed design of our adaptions.

2. Related Work

Previous work have explored predicting relative position of patches [6], predicting the permutation of patches [19], Image Colorization [29], affine transform prediction [28] and their combinations [7].

Recent works have been successful in bridging the gap between self supervised learning and fully supervised learning. Deep Infomax [10] experiments contrasting the feature vectors and the global features between images using an adversarial objective. Augmented Multi Scale Deep Infomax [2] builds on this approach to use multiple views (using transformations) to contrast with other images while lso experimenting with using the feature vectors from various levels of the network. Selfie [22] treats the jigsaw task in a new novel way by borrowing ideas from the field of NLP and using transformers in the image domain. We explore this model further in the following sections.

The latest work by FAIR called Pretext Invariant Representation Learning [17] approaches the idea in a quite a similar fashion to our proposal. They use two views of images (Image and patches) to construct corresponding representations for contrastive learning. They employ a Memory bank to sample negatives for the objective.

3. Methods

We introduce models used in our pilot experiments in section 3.1 and 3.2, the baseline model in section 3.3, and each adaptation in our full model in section 3.4, 3.5 and 3.6.

3.1. Selfie

We first implement the Selfie [22] model. This model treats the task of Jigsaw as a query key correspondence approach. In pre-train stage, the model takes a minibatch of N images $\{X_1, X_2, \ldots, X_N\}$. Each image X_i is made of M patches $\{X_i^1, X_i^2, \ldots, X_i^M\}$. Together with a mask,$V = \{v_i^j\}$, where v_i^j is either 0 or 1 and $\sum_j v_i^j = Q$. 1 means the patch is provided as context, while 0 means the patch needs to be predicted. We feed each patch into a convolution network f to get the patch representation P.

Now the masks are applied to the representations.

$$P_i^j = v_i^1 P_i^1, v_i^2 P_i^2, \ldots, v_i^M P_i^M \tag{1}$$
$$C_i^j = v_i^1 P_i^1, v_i^2 P_i^2, \ldots, v_i^C P_i^C \tag{2}$$
$$h_i^j = v_i^1 P_i^1, v_i^2 P_i^2, \ldots, v_i^Q P_i^Q \tag{3}$$
$$P_i^j = C_i^j \bigcup h_i^j \tag{4}$$
$$\tag{5}$$

where P_i^j is the patch representation of the j^{th} patch of the i^{th} instance in the mini batch, M is the total number of patches, C is the number of patches used to build the context vector, Q is the number of patches used as queries. The context patches (v_i^j = 0) are then passed along with a learnable vector u_0 which is of same dimension as each patch representation to a Transformer module.

$$\hat{C}_i^j = h(u_0, v_i^1 P_i^1, v_i^2 P_i^2, \ldots, v_i^C) P_i^C p) \tag{6}$$
$$= (u, p_1, p_2, \ldots\ldots, p_c) \tag{7}$$
$$v_k = u + E^k \tag{8}$$

where Cp is the number of patches used for building the context vector for the image, $u, p_1, p_2 \ldots\ldots, p_c$ are the outputs corresponding to $(u_0, v_i^1 P_i^1, v_i^2 P_i^2, \ldots, v_i^C) P_i^C)$ in C_i^j, E^k is the positional embedding of the kth query h_k, v_k is the final global vector to be used for correspondence, $\hat{C}_i^j$ is the output from the Attention Pooling Layer.
This module attends to different patches in the image and understands the context of each patch. The vector corresponding to u_0 attends to all other patches and learns the global context of the image. Te output from the Transformer layer for the vector u_0, 'u' is then queried by a positional embedding of one of the query patches (v_i^j = 1). This helps the network build a correspondence between the global vector u and the patches not seen by 'u' thus learning the structure of the image. The loss is computed by:

$$L = -\sum_{k=1}^{q} (l_k * \log(\frac{\exp(v_k, q_k)}{\exp(v_k, q_k) + \sum_{i \neq k} \exp(v_k, q_i)})) \tag{9}$$

where q_k is the k^{th} query patch representation, l_k denotes the label for the k^{th} query and q is the number of query patches used for one instance.

3.2. Context from patches (Allp Model)

Inspired by Selfie [22], we explore the representations learnt from patches further. We construct a model Figure 1 which uses a contrastive loss between representations learnt only by using the patches of an image to build the final global representation of each image. Specifically given a minibatch of N images, we covert each image to p patches, ${I_i}^j$.

${p_i}^j$ = P(${I_i}^j$) where P is the patch network (P).

The jigsaw paper [19] uses an architecture where each patch is passed through a patch network and then concatenated together and projected to a single global vector. We try to improve this architecture by using self attention layers in the convolutional neural network. Similar to the selfie model described in section 3.1 we use attention pooling to create the global vector u. We also include self attention layers in between the layers in the resnet, particularly one after each resblock in the resnet model to facilitate exchange of information between the patch network layers corresponding to all the patches for the image. We construct two variants, with and without the self attention layers before the attention pooling layer to understand the impact of these layers. The model proceeds as follows:

$$P_i^j = p_i^1, p_i^2, \ldots, p_i^P \tag{10}$$
$$\hat{P}_i^j = h(P_i^j) \tag{11}$$

where P is the total number of patches in the image, P_i^j is the representations of the patch j in image i. The loss function is computed as follows:

$$L = \sum_{i=1}^{B} \sum_{j=i+1}^{dup} (-log(S(pos_i, pos_j)) - \tag{12}$$
$$\sum_{neg_k \epsilon N_i} \log(1 - S(pos_i, neg_k))) \tag{13}$$

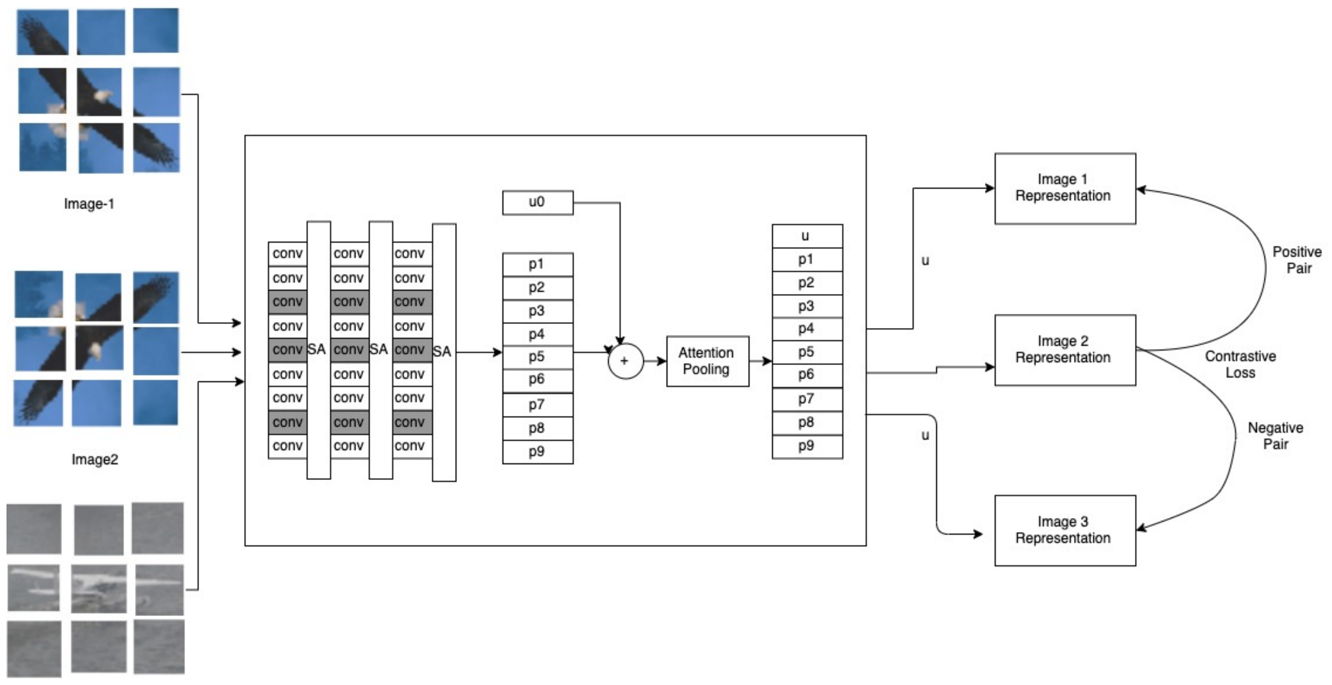

Figure 1. Context embedding from patches

where B represents the batch size, dup represents the number of transformations (including the original image), pos and neg represents the positive and negative image representations, S represents cosine similarity and finally N represents the set of all negatives in the minibatch corresponding to a given positive image representation.

Now in order for the model to find the optimum the network should understand the local content of each patch, relative position of each patch and the relationship between patches to understand the content in the image.

3.3. Baseline

We use a variant of Selfie [22] as our baseline model, and build adaptations upon it. Figure 2 shows our baseline model.

In this variant we experiment providing the positional embeddings and the patch representations in different contexts. The initial stages are similar to the selfie model where each instance in the mini batch is converted to the corresponding patches and passed through a convolution neural network to get the representations.

The masks are applied to patch representations. After adding with the position embedding E^j , the representations are fed into a transformer model h.

$$\hat{P}_i^j = h(v_i^1 P_i^1 + E^1, v_i^2 P_i^2 + E^2, \dots, v_i^M P_i^M + E^M) \quad (14)$$

$\hat{P}_i^j$ is the prediction of P_i^j given the context patches. The loss is computed by

$$L_{\text{mp}} = -\frac{\alpha_{\text{mp}}}{N^2 MQ}\left(\sum_{ij} \log|\hat{P}_i^j, P_i^j| + \sum_{ijkl} \log(1 - |\hat{P}_i^j, P_l^m|)\right) \quad (15)$$

Where $|x, y|$ is the cosine similarity between two vectors x, y after projecting to a 128D space, the cosine value is rescaled from $[-1, 1]$ to $[0, 1]$.

This loss push the prediction patch to be similar to the original patch, and far away from any other patches. The loss is averaged on all the pairs in the minibatch, and the weight is controlled by a coefficient α_{mp}.

3.4. Double view and hybrid architecture

Consider a model that receive two views of the image, a complete image and as patches. The patches are processed independently in the convolutional network and hence might lose access to the information about the correspondence between patches in order to understand the content of the image.

To make up for it, we add self-attention layers between convolution blocks. The representation from previous layer are pooled and fed into the self-attention layer. A residual link connects the representation before and after self-attention, so that the model can choose whether or not exchange information.

In addition, we add another loss term to encourage the model to produce consistent outputs from the patch view

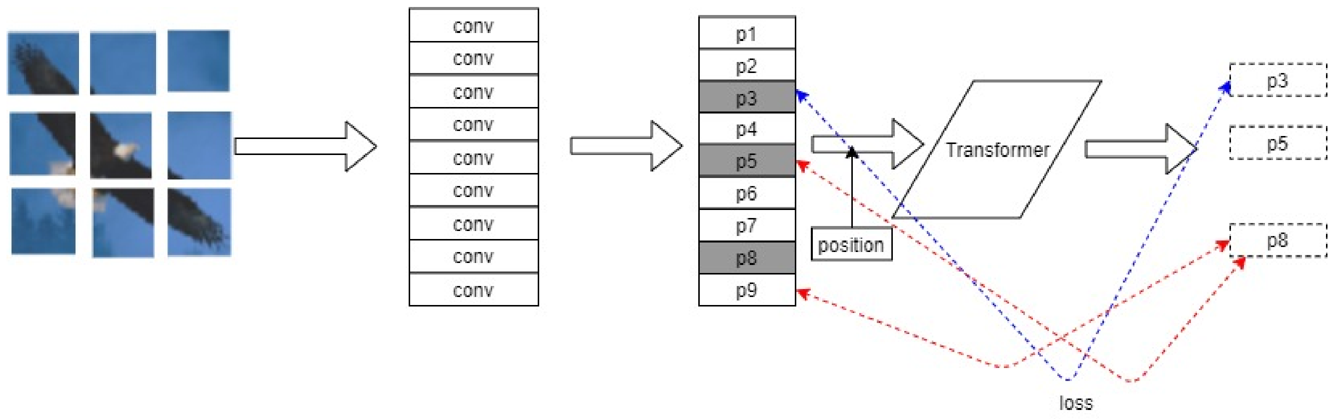

Figure 2. Our variant of the selfie model. The dark grey color marks the representations of masked out patches, they are not visible to the transformer. The red dashed lines are the negative pairs, while the blud dashed lines are the positive pairs. The dashed rectangles are the prediction.

and the complete image view. This is similar to [17].

$$L_{\text{mi}} = -\frac{\alpha_{\text{mi}}}{N^2}(\sum_i \log|\hat{I}_i, I_i| + \sum_{ij} \log(1 - |\hat{I}_i, I_j|)) \quad (16)$$

Where $\hat{I}_i$ is from feeding a special position [IMG] together with patch representations into transformer, and I_i is the output of the convolution network from the full image. The loss is averaged on all the image pairs in the minibatch, and the weight is controlled by a coefficient α_{mi}.

Since both image view and patch view yield a representation for the whole image, in supervised training stage, the classifier can use either or the concatenation of both as input, we leave this option as a hyper-parameter and explore it in 4.4.

3.5. Dynamic vocabulary

Previous works [23, 1] have shown the importance of using large amount of negative samples. Usually a memory bank is used to keep the representation of previous iterations. In this work, we propose an alternative approach called dynamic vocabulary.

A vocabulary contains S entries, each entry is made of a 128D vector and v_i a mass scalar m_i, the vectors are randomly initialized and normalized to unit sphere, and the mass values are initialized to 1. When a new representation $\{u_j\}$ from patch or image representation is generated in a minibatch, we first find out the most similar vector in the vocabulary by compute the cosine similarity,

$$c_j = \text{argmax}_i |m_i * v_i + u_j, u_j| \quad (17)$$

this favors entry with smaller mass value, so as to avoid unbalanced clusters. Then v_i and m_i are updated according to

$$v_i = \text{norm}(v_i * m_i + \sum_{j, c_j = i} u_j) \quad (18)$$

$$m_i = (m * i + \sum_{j, c_j = i} 1) * \lambda \quad (19)$$

where λ is a hyper-parameter. This updating process is not accumulating any gradient. Between minibatches, the direction of entries that are not matched by any target vector will stay still, with their mass gradually decay, until the mass fall small enough, then they will be reassigned to a less similar vector.

When feeding with prediction vectors $\{\hat{u_j}\}$ and target vectors $\{u_j\}$, the vocabulary can create a loss by

$$L = \sum_{ij} (|v_i, u_j| - 1) \log(1 - |v_i, \hat{u_j}|) \quad (20)$$

Intuitively, the more dissimilar to the target vector, a vector in the vocabulary is, the larger weight is applied to push the prediction away from it. If the target vector matches a vector in the vocabulary, this vector no longer has any impact on the prediction.

Two other term of losses, L_{vp} and L_{vi}, are introduced by taking this loss and scaling it according to number of prediction vectors the corresponding coefficient, α_{vp} and α_{vi}.

3.6. Using transformed images and patches

Inspired by [2], we use patches from transformed image of the same location as additional positive samples. Specifically, instead of N images, we have $\frac{N}{D}$ image of D transformed versions per minibatch, the input images become $\{X_{11}, X_{12}, \ldots, X_{1D}, X_{21}, X_{22}, \ldots, X_{\frac{N}{D}D}\}$. Comparing to baseline, this makes $(\hat{P}_{ij}, P_{ik})$ positive pair. In

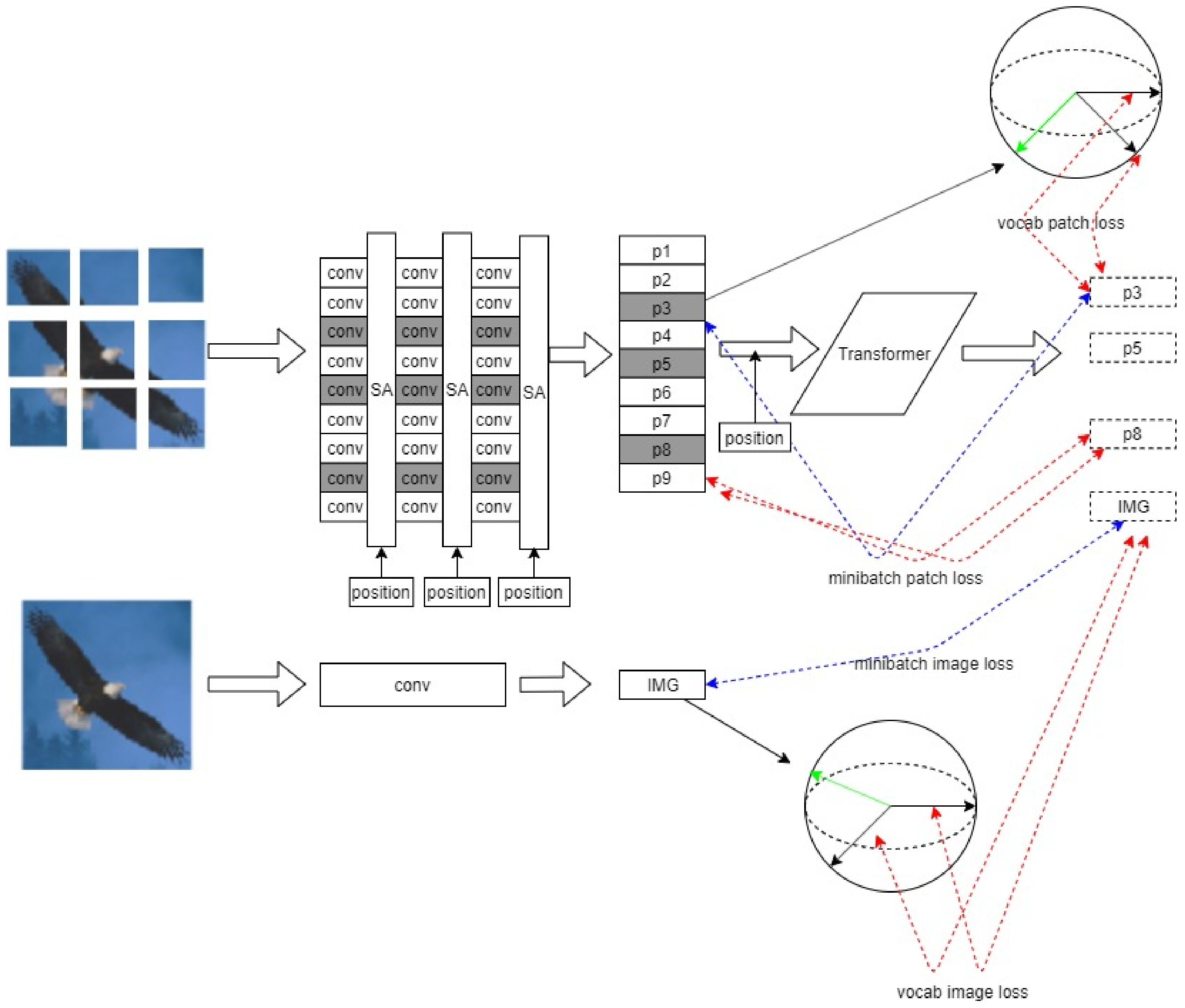

Figure 3. Our full model. The spheres represent the dynamic vocabularies for images and patches. The SA rectangles are the self-attention layers that exchange information between patches. The dashed rectangles are the predictions. Grey color masks the masked patches, they are not visible to Transformer or self-attention layers.

this way, we are forcing the model to match patches regardless of superficial features that are subject to the transformations. Similarly for image loss, the transformed image also make up positive pairs with prediction from the patches.

4. Experiments

4.1. Setup

Dataset. We run all our experiments on the STL-10 dataset[4] and the CIFAR10 dataset. In case of CIFAR10, we use the full training data without labels for pre-train, then use training data with labels for supervised training. For STL10, we use the 5k 1-fold test protocol, first pre-train the model with 100k unlabeled images, then train the model with 5k labeled image. For both datasets, We kept 10% of the labeled data for validation in supervised training. We use the same data augmentation methods as [2].

Model Architecture. We use ResNet-50v2 as the backbone network. We use all four residual blocks when the whole image is used as the input and use the first 3 three residual blocks if patches are used as the input. We use three layers of transformers, with hidden dimension h as 512 for the information exchange layers. The final attention pooling layer has the dimension 1024. When using exchange layers (self attention layers) in the model, we insert self-attention layers after res2, res3 and res4. We used dropout 0.1 for all dropout layers in ResNet, Transformer and self-attention. We initialize convolution weights with Kaiming Norm [26], linear weights with 0 centered, 1e-3 standard variance normal distribution, and bias with 0.

Model training. We run self-supervised pre-training for 100k iterations and supervised training for 5k iterations. There are two different evaluation strategies that we follow for finetuning the model. One, we use linear classifiers on top of the representations of each residual block and freeze the remaining layers to evaluate the linear separability of the representations at different levels in the network. Two, we finetune the whole model for the downstream supervised classification task which test the power of the self supervised task as a good weight initializer. We also compare classifiers that takes the image representation from the image view$I$ or patch view $\hat{I}$ or the concatenation of both, trained under frozen and finetuning settings in case of the hubrid model. We used AdamW optimizer[12] with 1e-4 weight decay and single cycle cosine learning rate scheduler[15] with 250 warmup steps. We set peaking learning rate as 3e-4 and batch size as 256 during pre-training, 1e-4 and 32 correspondingly for supervised-training.

Other details. We use pytorch[20] to implement our experiments. We break each images into $M = 16$ patches, and randomly mask out 4 patches. We project all representations to 128 dimension before computing similarity. Except in **??**, the dynamic vocabularies for both patches and images has $S = 16384$ entries and momentum of $\lambda = 0.99$. When transformed patches is used to provide additional positive samples, we choose $D = 4$, otherwise D is set to 1. For loss weights, we use $\alpha_{\text{mp}} = 8.0$, $\alpha_{\text{vp}} = 0.5$, $\alpha_{\text{mi}} = 4.0$, $\alpha_{\text{vi}} = 0.5$.

4.2. Results on CIFAR-10

We compare our result with state-of-the-art models and self-supervised learning methods. For comparison, we also include result using the same architecture as ours, but without any pre-training.

We report accuracy in Table [**?**] on the CIFAR-10 dataset for our basic implementations to help understand the reliability of the self supervised tasks that we propose. These are accuracy based on the linear separability of the representations learnt from the self supervised tasks. We see that the selfie model performs the best among them although some models report accuracy based on SVM classifiers.

4.3. Results on STL-10

We show the accuracy of our models on STL-10 task in Table 4.3. Our result falls short against state-of-the-art results. But the relative gain over baseline suggests potential improvements.

Model	**Accuracy**
VAE[13]*	0.50
AAE [16]*	0.55
BiGAN [8]*	0.56
NAT [3]*	0.56
Deep Info Max [10] *	0.63
Allp	0.58
Selfie[22]	**0.65**

Table 1. Accuracy on CIFAR-10 of linear classifiers trained at various layers of the model. Rows denoted by * denotes evaluation was done using SVM instead of linear layers

Model	Acc.(tune)	Acc.(frozen)
No-pretrain	0.70	N/A
Baseline	0.72	N/A
DV2	0.79	N/A

Table 2. Accuracy on STL-10 from finetuning pre-trained models and train a linear classifier on top of frozen model.

4.4. How to use a pre-trained model?

Recent work in NLP [21] has shown finetuning the full model gives considerably better results than freezing the model, we wonder if this holds true for image classification.

In this experiment, we test a few models with linear classifier on the fixed representation of different layers. We also compare classifier that take the image representation from the image view$I$, patch view $\hat{I}$ or the concatenation of both, trained under frozen and finetuning settings.

4.5. Ablation Analysis

In this experiment, we add individual adaptations to the baseline model or remove it from the full model, to study the effect of each adaption.

4.6. Details of Dynamic Vocabulary

We try to vary the hyper-parameters of the dynamic vocabulary and see how it affects the performance.

5. Discussion

We show that learning from patches as a pretext task is promising. The novel objective relies on the model learning the structure of each patch, understanding the relationship between patches to realize the content in the image. Although we have not close the gap with the state of the art results, we believe with further experiments we can improve our results. In the future work we aim to adjust the hyper-parameters, apply our method to other datasets, experiment with more types of architectures and with providing positional embeddings at different contexts to see how and what the model learns. We believe our approach will be helpful to learning good image representations.

References

[1] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

[2] Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *CoRR*, abs/1906.00910, 2019.

[3] Piotr Bojanowski and Armand Joulin. Unsupervised learning by predicting noise. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 517–526. JMLR. org, 2017.

[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[7] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. *CoRR*, abs/1708.07860, 2017.

[8] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[9] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019.

[10] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization, 2018.

[11] Dahun Kim, Donghyeon Cho, Donggeun Yoo, and In So Kweon. Learning image representations by completing damaged jigsaw puzzles. *CoRR*, abs/1802.01880, 2018.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[14] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Self-supervised learning of 3d human pose using multi-view geometry. *CoRR*, abs/1903.02330, 2019.

[15] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

[17] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations, 2019.

[18] Pedro Morgado, Nuno Vasconcelos, Timothy R. Langlois, and Oliver Wang. Self-supervised generation of spatial audio for 360 video. *CoRR*, abs/1809.02587, 2018.

[19] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246, 2016.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[21] Matthew Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

[22] Trieu H. Trinh, Minh-Thang Luong, and Quoc V. Le. Selfie: Self-supervised pretraining for image embedding. *CoRR*, abs/1906.02940, 2019.

[23] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

[24] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. *CoRR*, abs/1904.03597, 2019.

[25] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. *CoRR*, abs/1903.07593, 2019.

[26] Yuxin Wu and Kaiming He. Group normalization. *CoRR*, abs/1803.08494, 2018.

[27] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

[28] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. AET vs. AED: unsupervised representation learning by auto-encoding transformations rather than data. *CoRR*, abs/1901.04596, 2019.

[29] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.