Context Autoencoder for Self-Supervised Representation Learning

Xiaokang Chen [1 2] **Mingyu Ding** [3] **Xiaodi Wang** [4] **Ying Xin** [4] **Shentong Mo** [4] **Yunhao Wang** [4] **Shumin Han** [4]
Ping Luo [3] **Gang Zeng** [1 2] **Jingdong Wang** [4]

Abstract

We present a novel masked image modeling (MIM) approach, context autoencoder (CAE), for self-supervised learning. We randomly partition the image into two sets: visible patches and masked patches. The CAE architecture consists of: (i) an encoder that takes visible patches as input and outputs their latent representations, (ii) a latent context regressor that predicts the masked patch representations from the visible patch representations that are not updated in this regressor, (iii) a decoder that takes the estimated masked patch representations as input and makes predictions for the masked patches, and (iv) an alignment module that aligns the masked patch representation estimation with the masked patch representations computed from the encoder.

In comparison to previous MIM methods that couple the encoding and decoding roles, e.g., using a single module in BEiT, our approach attempts to *separate the encoding role (content understanding) from the decoding role (making predictions for masked patches)* using different modules, improving the content understanding capability. In addition, our approach makes predictions from the visible patches to the masked patches in *the latent representation space* that is expected to take on semantics. We demonstrate the effectiveness of our CAE through superior transfer performance in downstream tasks: semantic segmentation, and object detection and instance segmentation.

1. Introduction

We study the masked image modeling task for self-supervised representation learning. Masked image modeling (MIM) is a task of masking some patches of the input image and making predictions for the masked patches from the visible patches. It is expected that the resulting encoder network pretrained through solving the MIM task is able to extract the patch representations taking on semantics that are transferred to solving downstream tasks.

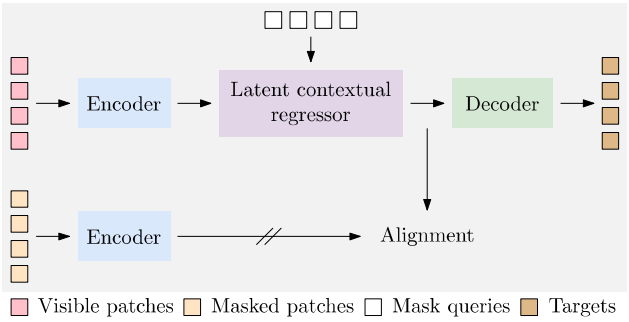

Figure 1: Context autoencoder. Our approach pretrains the encoder by predicting the targets for the masked patches from the visible patches through latent masked patch representation prediction and alignment.

BEiT (Bao et al., 2021) and the method studied in the ViT paper (Dosovitskiy et al., 2021), two MIM methods, learn a ViT (formed with self-attention) to predict the patch tokens and the pixels, respectively, and use the resulting ViT as the pretrained encoder. They take the visible patches and mask tokens representing the masked patches as input, and make predictions for both the visible and masked patches, where the predictions only for masked patches are evaluated during training. The two methods use the single ViT structure simultaneously for both encoding and decoding. Thus, only the partial capacity of the ViT is explored for encoding and representation learning, limiting the representation quality.

We present a context autoencoder (CAE) approach, illustrated in Figure 1, for improving the encoding quality. We randomly partition the image into two sets of patches: visible patches and masked patches. There are four components: an encoder, a latent contextual regressor, a decoder, and an alignment module. The encoder, a ViT structure, takes only the visible patches as input and learns the latent representations only for the visible patches. The latent contextual regressor estimates the masked patch representations according to the visible patch representations. The decoder takes

[1]Key Laboratory of Machine Perception (MOE), Peking University [2]Institute for Artificial Intelligence, Peking University [3]University of Hong Kong [4]Baidu. Correspondence to: Jingdong Wang <wangjingdong@outlook.com>.

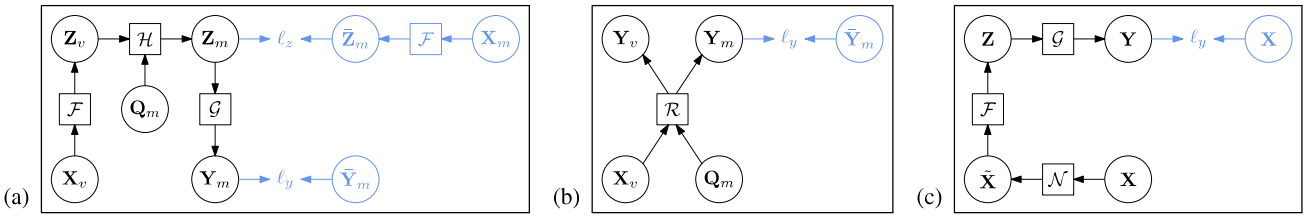

Figure 2: The computational graphs for (a) a context autoencoder (CAE), (b) BEiT (Bao et al., 2021), and (c) a denoising autoencoder (DAE). The parts in cornflower blue are for loss function. (a) The encoder $\mathcal{F}$ receives visible patches $\mathbf{X}_v$ and outputs their latent representations $\mathbf{Z}_v$. The latent contextual regressor $\mathcal{H}$ predicts the latent representations $\mathbf{Z}_m$ for masked patches from $\mathbf{Z}_v$. The decoder predicts the targets $\mathbf{Y}_m$ for masked patches from $\mathbf{Z}_m$. ℓ_z and ℓ_y are the loss functions. During training, the gradient is stopped for $\bar{\mathbf{Z}}_m$. See the detail in Section 2. (b) The input includes both visible patches $\mathbf{X}_v$ and mask queries $\mathbf{Q}_m$ representing masked patches, and the representations for them are updated within the function $\mathcal{R}$. (c) The function $\mathcal{N}$ is a noising function generating the noisy version $\tilde{\mathbf{X}}$ from the input $\mathbf{X}$. $\mathcal{F}$ and $\mathcal{G}$ are the normal encoder and decoder, respectively. For simplicity, the positional embeddings are not included in computational graphs. *(a) CAE and (c) DAE perform the encoding and decoding roles explicitly and separately, and (b) BEiT performs the encoding and decoding roles implicitly and simultaneously.*

the regressed masked patch representations as input and makes predictions for the masked patches. Furthermore, we align the regressed masked patch representations with the masked patch representations computed from the encoder that is the same as the one for encoding visible patches.

The encoder in the top stream in Figure 1 operates only on visible patches, only focusing on learning semantic representations. We also want that representation learning is taken only by the encoder through two things: The latent representations of visible patches are not updated in the other modules; and the alignment module expects that the output representations of latent contextual regressor are close to the latent representations for the masked patches computed from the encoder. In comparison to BEiT and the approach in the ViT paper (See Figures 2 (a) and (b) (black parts) for the difference), our CAE encoder exploits the whole capability for learning the representation, thus improving the representation quality.

Second, the prediction from the visible patches to the masked patches, i.e., generating a plausible semantic guess for the masked patches, is performed on the latent space using the latent contextual regressor. The predicted latent representations for the masked patches are constrained to match with the latent representations computed from the encoder. This expects that the latent representations can capture the semantics of the visual content as well as other information for predicting the targets.

Last, the CAE (including other MIM methods) makes predictions for randomly masked patches, thus caring about the representations for the patches. This indicates that our CAE encoder, e.g., pretrained on ImageNet-1K (Deng et al., 2009), learns semantics, not only for the center regions of the original images where the instances of the 1000 classes in ImageNet-1K usually lie, but also for other regions that potentially do not belong to the 1000 classes. This is different from typical contrastive pretraining methods (e.g., MoCo v3 (Chen et al., 2021) and SimCLR (Chen et al., 2020b)) that often compare global representations of augmented views and empirically exhibit a tendency to learn semantics mainly from the center patches of the original images.

We present the empirical performance of our approach on downstream tasks, semantic segmentation, and object detection and instance segmentation. The superior results over supervised pretraining, contrastive pretraining, and other MIM methods validate the aforementioned three points.

2. Approach

Our context autoencoder (CAE) pretrains the encoder by solving the masked image modeling task. The architecture, shown in Figure 1, consists of four parts: an encoder, a latent contextual regressor, a decoder, and an alignment module.

2.1. Architecture

We randomly split an image into two sets of patches: visible patches $\mathbf{X}_v$ and masked patches $\mathbf{X}_m$. The computational graph is provided in Figure 2 (a).

Encoder. The encoder $\mathcal{F}$ maps the visible patches $\mathbf{X}_v$ to the latent representations $\mathbf{Z}_v$. It only handles the visible patches. We use the ViT to form our encoder. It first embeds the visible patches by linear projection as patch embeddings, and adds the positional embeddings $\mathbf{P}_v$. Then it sends the combined embeddings into a sequence of transformer blocks that are based on *self-attention*, generating $\mathbf{Z}_m$.

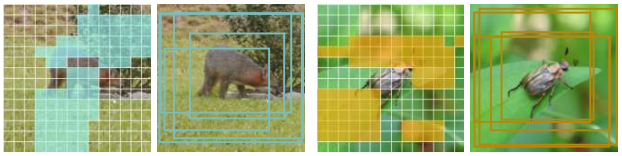

Figure 3: Illustration of random block-wise sampling and random cropping. Random block-wise sampling is used in our approach. Random cropping is a key data-augmentation scheme for contrastive pretraining.

Latent contextual regressor. The latent contextual regressor $\mathcal{H}$ predicts the latent representations $\mathbf{Z}_m$ for the masked patches from the latent representations $\mathbf{Z}_v$ of the visible patches output from the encoder. We form the latent contextual regressor $\mathcal{H}$ using a series of transformer blocks that are based on *cross-attention*.

The initial queries $\mathbf{Q}_m$, called mask queries, are representations of mask patches that are learned as model parameters and are the same for all the masked patches. The keys and the values consists of the visible patch representations $\mathbf{Z}_v$ and the output of the previous cross-attention layer (mask queries for the first cross-attention layer). The corresponding positional embeddings are considered when computing the cross-attention weights between the queries and the keys. In this process, the latent representations $\mathbf{Z}_v$ of the visible patches are not updated.

Decoder. The decoder $\mathcal{G}$ maps the latent representations $\mathbf{Z}_m$ of the masked patches to some forms $\mathbf{Y}_m$ of the masked patches, e.g., discrete tokens this paper takes. The decoder, similar to the encoder, is a stack of transformer blocks that are based on *self-attention*. The decoder only receives the latent representations of the masked patches (the output of the latent contextual regressor), and the positional embeddings of the masked patches as input without directly using the information of the visible patches.

Latent representation alignment. The latent representation alignment module imposes the constraint on the latent representations $\mathbf{Z}_m$ of the masked patches predicted by the latent contextual regressor. We feed the masked patches $\mathbf{X}_m$ into the encoder, which is the same as the one for encoding visible patches, and generate the representations $\bar{\mathbf{Z}}_m$ of the masked patches. We then align the two latent representations $\mathbf{Z}_m$ and $\bar{\mathbf{Z}}_m$ for the masked patches.

2.2. Objective Function

Masking and targets. Following BEiT (Bao et al., 2021), we adopt the random block-wise masking strategy (illustrated in Figure 3) to split the input image into two sets of patches, visible and masked patches. For each image, 75 of 196 (14×14) patches are masked.

We use the pre-trained DALL-E (Ramesh et al., 2021) tokenizer to generate the discrete tokens for forming the targets. The input image is fed into the DALL-E tokenizer, assigning a discrete token to each patch. The target tokens for the masked patches are denoted as $\bar{\mathbf{Y}}_m$.

Loss function. The loss function (illustrated in Figure 2 (a), the part in cornflower blue.) consists of a decoding loss: $\ell_y(\mathbf{Y}_m, \bar{\mathbf{Y}}_m)$, and an alignment loss: $\ell_z(\mathbf{Z}_m, \bar{\mathbf{Z}}_m)$. The whole loss is a weighted sum:

$$\ell_y(\mathbf{Y}_m, \bar{\mathbf{Y}}_m) + \lambda\, \ell_z(\mathbf{Z}_m, \bar{\mathbf{Z}}_m). \tag{1}$$

We use the MSE loss for $\ell_z(\mathbf{Z}_m, \bar{\mathbf{Z}}_m)$ and the cross-entropy loss for $\ell_y(\mathbf{Y}_m, \bar{\mathbf{Y}}_m)$. λ is 2 in our experiments.

3. Analysis and Conenction

3.1. Analysis

The CAE encoder cares about the patch representations. The CAE makes predictions for randomly masked patches from the visible patches. This requires that the CAE encoder cares about the representations for the patches other than only the global representation so that the CAE explores the relations among the patches for making predictions.

The input and output representations of latent contextual regressor are in the same latent space. Our CAE expects that the representations for the masked patches output from the latent contextual regressor lie in the same space with the input, the representations of the visible patches. We verify this through full image reconstruction where our CAE is trained using the pixel colors as the prediction targets. We feed the full patches (without masking, all the image patches are visible) into the encoder, then skip the latent contextual regressor and directly send the encoded patch representations to the decoder for reconstructing the full image. Figure 4 provides reconstruction results for several examples randomly sampled from the ImageNet-1K validation set, implying that the input and output representations of latent contextual regressor are in the same space.

Probabilistic formulation. The MIM problem can be formulated in the probabilistic form, maximizing the probability of the predictions $\mathbf{Y}_m$ of the masked patches given the conditions, the visible patches $\mathbf{X}_v$, the positions $\mathbf{P}_v$ of the visible patches, and the positions $\mathbf{P}_m$ of the masked patches: $P(\mathbf{Y}_m|\mathbf{X}_v, \mathbf{P}_v, \mathbf{P}_m)$. It can be solved by introducing latent representations $\mathbf{Z}_m$ and $\mathbf{Z}_v$, with the assumption that $\mathbf{Z}_v$ and $\mathbf{P}_m$ ($\mathbf{Y}_m$ and $\mathbf{P}_v$) are conditionally independent:

$$\begin{aligned} & P(\mathbf{Y}_m|\mathbf{X}_v, \mathbf{P}_v, \mathbf{P}_m) \\ = \; & P(\mathbf{Z}_v|\mathbf{X}_v, \mathbf{P}_v) P(\mathbf{Z}_m|\mathbf{Z}_v, \mathbf{P}_v, \mathbf{P}_m) P(\mathbf{Y}_m|\mathbf{Z}_m, \mathbf{P}_m), \end{aligned}$$

where the three terms on the right side correspond to three parts of our CAE: the encoder, the latent contextual regressor, and the decoder, respectively.

Figure 4: Illustrating that the input and output representations of latent contextual regressor are in the same space. We reconstruct the image by feeding the full image (top) into the CAE encoder and then the CAE decoder outputting the reconstructed image (bottom). It can be seen that the image can be constructed with the semantics kept when skipping latent contextual regressor, verifying the same space expectation.

The latent representation alignment module can be written as a conditional probability, $P(\mathbf{Z}_m|\bar{\mathbf{Z}}_m)$, where $\bar{\mathbf{Z}}_m$ is the masked patch representations computed from the encoder.

Intuitive interpretation. Humans are able to hallucinate what appears in the masked regions and how they appear according to the visible regions. We speculate that humans do this possibly in a way similar as the following example: given that only the region of the dog head is visible and the remaining parts are missing, one can (a) recognize the visible region to be about a dog, (b) predict the regions where the other parts of the dog appear, and (c) guess what the other parts look like.

Our CAE encoder is in some sense like the human recognition step (a). It understands the content by mapping the visual patches into latent representations that lie in the subspace that corresponds to the category dog[1]. We empirically verify this by projecting the latent representations of the patches from the images randomly sampled from the ADE20K set[2] to the 2D space using t-SNE (Van der Maaten & Hinton, 2008). The 2D projections shown in Figure 5 implies that the latent representations are clustered to some degree for different categories (though not perfect as our CAE is pretrained on ImageNet-1K).

The latent contextual regressor is like step (b). It produces a plausible hypothesis for the masked patches, and describes the regions corresponding to the other parts of the dog using latent representations. The CAE decoder is like step (c), mapping the latent representations to the targets. It should be noted that the latent representations might contain other information besides the semantic information, e.g., the part information and the information for making predictions.

[1]Our encoder does not know that the subspace is about a dog, and just separates it from the subspaces of other categories.

[2]We choose to use the pixel labels for checking if the representations are correctly clustered.

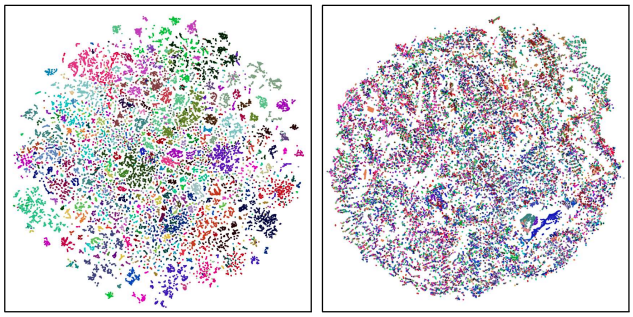

Figure 5: t-SNE visualization (one color for one category) of representations extracted from the images in ADE20K. Left: ViT pretrained with our CAE; Right: ViT with random weights. The latent representations from our CAE encoder for each category tend to be grouped together (though not perfect as our CAE encoder is pretrained on ImageNet-1K instead of ADE20K).

3.2. Connection

Relation to autoencoder. The original autoencoder (LeCun, 1987; Gallinari et al., 1987; Hinton & Zemel, 1994) consists of an encoder and a decoder. The encoder maps the input into a latent representation, and the decoder reconstructs the input from the latent representation. The denoising autoencoder (DAE) (Vincent et al., 2010) (depicted in Figure 2 (c)), a variant of autoencoder, corrupts the input by adding noises and still reconstructs the non-corrupted input.

Our CAE encoder (depicted in Figure 2 (a)) is similar to the original autoencoder and also contains an encoder and a decoder. Different from the autoencoder where the encoder and the decoder process the whole image, our encoder takes a portion of patches as input and our decoder takes the estimated latent presentations of the other portion of patches as input. Importantly, the CAE introduces a latent contextual regressor that makes predictions in the latent space from the visible patches to the masked patches.

Relation to BEiT. BEiT (Bao et al., 2021) (Figure 2 (b)) feeds both visible patches and masked patches (represented by mask tokens) into a ViT that is based on self-attention, and then predicts the discrete patch tokens, where only the tokens for masked patches are counted in the loss function.

The ViT in BEiT simultaneously understands the image content and produces a hypothesis for the masked patches. There is no explicit and separate representation extraction module, indicating that the ViT network uses the partial capability for representation learning. In contrast, the CAE encoder is only for content understanding without making predictions for masked patches. The other parts in CAE do not update the representations for visible patches. The output of the latent contextual regressor is expected to align with the masked patch presentations computed from the

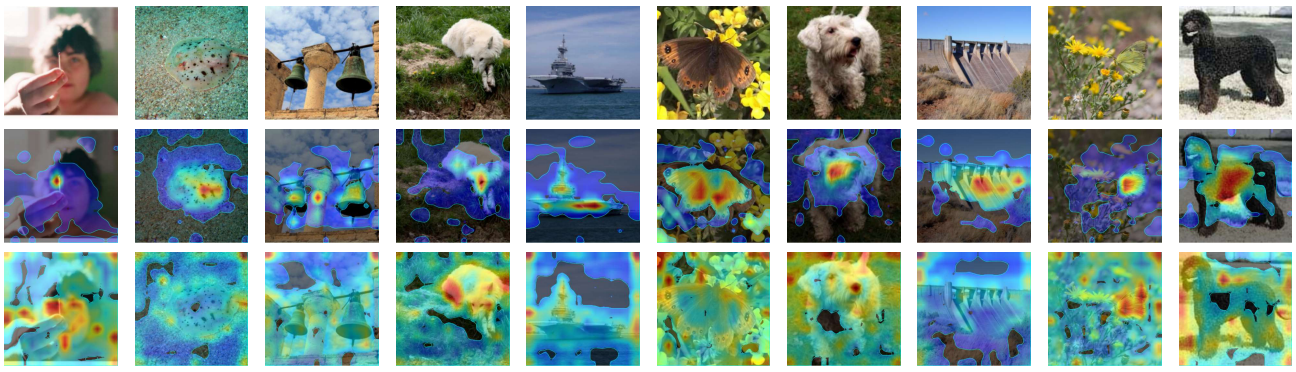

Figure 6: Illustrating the attention map averaged over 12 attention heads between the class token and the patch tokens in the last layer of the ViT encoder pretrained on ImageNet-1K. The region inside the blue contour is obtained by thresholding the attention weights to keep 50% of the mass. Top: Input image, Middle: MoCo v3, a typical contrastive learning method, and Bottom: our CAE. One can see that MoCo v3 tends to focus mainly on the centering regions and little on other patches, and our CAE tends to consider almost all the patches.

encoder, constraining that the representation extraction role is only by the encoder. This implies that our CAE encoder exploits the whole capability for representation learning.

Comparison to contrastive learning. Typical contrastive learning methods, e.g., SimCLR (Chen et al., 2020b) and MoCo (He et al., 2020; Chen et al., 2021), pretrain the networks by solving the pretext task, maximizing the similarities between augmented views (e.g., random crops) from the same image and minimizing the similarities between augmented views from different images.

It is shown that in (Chen et al., 2020b) random cropping plays an important role in view augmentation for contrastive learning. Through analyzing random crops (illustrated in Figure 3), we observe that the center pixels in the original image space have large chances to belong to random crops. We suspect that the global representation, learned by contrastive learning for a random crop possibly with other augmentation schemes, tends to focus mainly on the center pixels in the original image, so that the representations of different crops from the same image can be possibly similar. Figure 6 (the second row) shows that the center region of the original image for the typical contrastive learning approach, MoCo v3, is highly attended.

In contrast, our MIM approach, CAE, randomly samples the patches from the augmented views to form the visible and masked patches. All the patches are possible to be randomly masked for the augmented views and accordingly the original image. Thus, the CAE encoder needs to learn good representations for all the patches, to make good predictions for the masked patches from the visible patches. Figure 6 (the third row) illustrates that almost all the patches in the original images are considered in our CAE encoder.

Considering that the instances of the 1000 categories in ImageNet-1K locate mainly around the center of the original images, typical contrastive learning methods, e.g., MoCo v3, learn the knowledge mainly about the 1000 categories, which is similar to supervised pretraining. But our CAE and other MIM methods are able to learn more knowledge beyond the 1000 categories from the non-center image regions. This indicates that the CAE has the potential to performs better for downstream tasks.

4. Experiments

4.1. Implementation

We study the standard ViT small and base architectures, ViT-S (12 transformer blocks with dimension 384) and ViT-B (12 transformer blocks with dimension 768). The latent contextual regressor consists of 4 transformer blocks based on cross-attention, and the decoder consists of 4 transformer blocks based on self-attention, and an extra linear projection for making predictions.

We follow BEiT (Bao et al., 2021) to train the CAE on ImageNet-1K. We partition the image of 224×224 into 14×14 patches with the patch size being 16×16. We use standard random cropping and horizontal flipping for data augmentation. The pretraining settings are almost the same as BEiT (Bao et al., 2021) (See Appendix A for details).

4.2. Pretraining Evaluation

Linear probing. Linear probing is widely used as a proxy of pretraining quality evaluation for self-supervised representation learning. It learns a linear classifier over the image-level representation output from the pretrained encoder by using the labels of the images, and then tests the performance on the validate set.

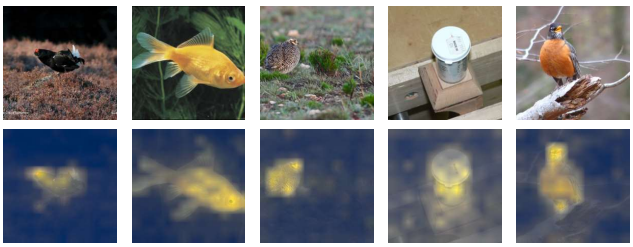

Figure 7: Illustrating the cross-attention unit in attentive probing. The attention map (bottom) is the average of cross-attention maps over 12 heads between the extra class token and the patches. One can see that the attended region lies mainly in the object, which helps image classification.

Attentive probing. The output of the CAE encoder is representations for all the patches. It is not suitable to linearly probe the representation, averagely-pooled from patch representations, because the image label in ImageNet-1K only corresponds to a portion of patches. It is also not suitable to use the default class token within the encoder because the default class token serves as a role of aggregating the patch representations for better patch representation extraction and is not merely for the portion of patches corresponding to the image label.

To use the image-level label as a proxy of evaluating the pretraining quality for our CAE encoder, we need to attend the patches that are related to the label. We introduce a simple modification by using a cross-attention unit with an extra class token (that is different from the class token in the encoder) as the query and the encoder output as the keys and the values, followed by a linear classifier. The introduced cross-attention unit is able to care mainly about the patches belonging to the 1000 classes in ImageNet-1K and remove the interference of other patches. Figure 7 illustrates the effect of the cross-attention unit, showing that the extra cross-attention unit is able to to some degree attend the regions that are related to the 1000 ImageNet-1K classes.

Results. Table 1 shows the results with three schemes, linear probing (LIN), attentive probing (ATT), and fine-tuning (FT) for representative contrastive pretraining (MoCo v3 and DINO) and MIM (BEiT and MAE) methods, and our approach CAE. The models of MAE with 300 epochs and BEiT are pretrained by us using the official implementations, and other models are the officially released models.

We highlight a few observations. The fine-tuning performance for these methods are very similar and there is only a minor difference. We think that the reason is that self-supervised pretraining and fine-tuning are conducted on the same dataset and no extra knowledge is introduced for image classification. The minor difference might come from the optimization aspect: different initialization (provided by pretrained models) for fine-tuning.

Table 1: Pretraining quality evaluation in terms of fine-tuning (FT), linear probing (LIN), and attentive probing (ATT). #Epochs refers to the number of pretraining epochs. For reference, we report the top-1 accuracy (in the column ATT) of the supervised training approach DeiT (Touvron et al., 2020) to show how far our ATT score is from supervised training. The results for other models and our models are based on our implementations for fine-tuning, linear probing, and attentive probing. MoCo v3 and DINO adopt multi-crop augmentation in each minibatch. MoCo v3: 2 global crops of 224×224. DINO: 2 global crops of 224×224 and 10 local crops of 96×96.

Method	#Epochs	#Crops	FT	LIN	ATT
Methods using ViT-S:					
DeiT	300	-	-	-	79.9
MoCo v3	300	2	81.7	73.1	73.8
BEiT	300	1	81.7	15.7	23.6
CAE	300	1	81.8	50.8	64.8
Methods using ViT-B:					
DeiT	300	-	-	-	81.8
MoCo v3	300	2	83.0	76.2	77.0
DINO	400	12	83.3	77.3	77.8
BEiT	300	1	83.0	37.6	49.4
MAE	300	1	82.9	61.5	71.1
MAE	1600	1	83.6	67.8	74.2
CAE	300	1	83.3	64.2	73.7
CAE	800	1	83.6	68.3	76.0

In teams of linear probing, the scores of the contrastive learning methods, MoCo v3 and DINO, are higher than the MIM methods. This is as expected because contrastive learning focuses mainly on learning the representations for 1000 classes (See discussion in Section 3). The pretraining is relatively easier than existing MIM methods as contrastive learning mainly cares about the 1000 classes and MIM methods may care about the classes beyond the 1000 classes.

For the MIM methods, the scores of attentive probing are much larger than linear probing. This validates our analysis: the MIM methods extract the representations for all the patches, and the classification task needs to attend the corresponding portion of patches.

The LIN and ATT scores are similar for contrastive pretraining: e.g., with ViT-B, (76.2 vs 77.0) for MoCo v3, and (77.3 vs 77.8) for DINO. This means that the extra cross-attention in attentive probing does not make a big difference, which is one more evidence for our analysis in Section 3 that they already focus mainly on the region where the instance in the 1000 categories lies.

Table 2: Ablation studies for the decoder and the alignment module in our CAE. All the models are pretrained on ImageNet-1K with 300 epochs.

	Decoder	Align	ATT	ADE	COCO
CAE	×	×	70.7	46.2	45.6
CAE	√	×	72.6	46.4	45.7
CAE	√	√	73.7	47.7	48.0

4.3. Ablation Studies

The CAE architecture contains three components for pretraining the encoder: a latent contextual regressor, a decoder, and an alignment module. We cannot remove the latent contextual regressor that is the only unit to make predictions for masked patches from visible patches in our architecture. We study the other two components, the decoder and the alignment module.

Table 2 shows the ablation results. We report the scores for attentive probing, and downstream tasks: semantic segmentation on ADE and object detection on COCO. One can see that the downstream task performance is almost the same when only the decoder is added and that the performance increases when the decoder and the alignment module are both added. This also verifies that the alignment module is important for constraining that the input and output representations of the latent contextual regressor lie in the same space and accordingly improving the representation quality.

4.4. Downstream Tasks

Semantic segmentation on ADE20K (Zhou et al., 2017)**.** We follow the code (Bao et al., 2021) to use UperNet (Xiao et al., 2018) (See Appendix A for training details). Table 3 shows that our CAE with 300 training epochs performs better than DeiT, MoCo v3 and DINO (400 epochs), MAE (300 epochs) and BEiT expect MAE (1600 epochs). Our CAE (800 epochs) further improves the segmentation scores and outperforms BEiT (800 epochs), MAE 1600 epochs, MoCo v3 and DeiT by 2.3, 0.7, 1.6 and 1.8, respectively.

The superior results over supervised and contrastive pretraining methods, DeiT, MoCo v3 and DINO, stem from that our approach captures the knowledge beyond the 1000 classes in ImageNet-1K. The superior results over BEiT and MAE stem from that our CAE decouples the encoding and decoding roles and achieves a better-pretrained encoder.

Object detection and instance segmentation on COCO (Lin et al., 2014)**.** We adopt the Mask R-CNN approach (He et al., 2017) that produces bounding boxes and instance masks simultaneously, with the ViT as the backbone (See Appendix A for training details). We apply the same object detection system to the methods in Table 4.

Table 3: Semantic segmentation on ADE20K. All the methods use a ViT-B architecture. All the results are based on the same implementation for semantic segmentation. #Epochs refers to the number of pretraining epochs. *: use multi-crop augmentation (See Table 1) and equivalently take a larger number of epochs compared to one-crop augmentation.

Method	#Epochs	Supervised	Self-supervised	mIoU
DeiT	300	√	×	47.0
MoCo v3*	300	×	√	47.2
DINO*	400	×	√	47.2
BEiT	300	×	√	45.5
BEiT	800	×	√	46.5
MAE	300	×	√	45.8
MAE	1600	×	√	48.1
CAE	300	×	√	47.7
CAE	800	×	√	**48.8**

We report the box AP for object detection and the mask AP for instance segmentation. The observations are consistent to those for semantic segmentation in Table 3. Our approach (300 epochs, ViT-B) is superior to all the other models except that a little lower tha MAE (1600 epochs). Our approach (800 epochs) outperforms MAE (1600 epochs), MoCo v3 and DeiT by 0.8, 3.7 and 2.3, respectively.

5. Related Work

Self-supervised representation learning has been widely studied in computer vision (Dosovitskiy et al., 2014; 2015; Doersch et al., 2015; Xie et al., 2016; Yang et al., 2016; van den Oord et al., 2018; Caron et al., 2018; Asano et al., 2019; Caron et al., 2019; Huang et al., 2019; Zhuang et al., 2019; Ermolov et al., 2021; Li et al., 2020; Gidaris et al., 2020a; Henaff, 2020; Gidaris et al., 2020b; Goyal et al., 2021; Zbontar et al., 2021; Bardes et al., 2021). The following mainly reviews closely-related methods.

Autoencoding. Traditionally, autoencoders were used for dimensionality reduction or feature learning (LeCun, 1987; Gallinari et al., 1987; Hinton & Zemel, 1994; Hinton & Salakhutdinov, 2006; Ranzato et al., 2007; Vincent et al., 2008; Kingma & Welling, 2013).

The denoising autoencoder (DAE) is an autoencoder that receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output. The variants or modifications of DAE were adopted for self-supervised representation learning, e.g., corruption by masking pixels (Vincent et al., 2010; Pathak et al., 2016; Chen et al., 2020a), removing color channels (Zhang et al., 2016), shuffling image patches (Noroozi & Favaro, 2016), denoising pixel-level noise (Atito et al., 2021) and so on.

Table 4: Object detection and instance segmentation on COCO. Mask R-CNN is adopted and trained with the $1\times$ schedule. All the results are based on the same implementation for object detection and instance segmentation. #Epochs refers to the number of pretraining epochs on ImageNet-1K. *: use multi-crop augmentation (See Table 1).

Method	#Epochs	Supervised	Self-supervised	Object detection			Instance segmentation		
				AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Methods using ViT-S:									
DeiT	300	√	×	43.1	65.2	46.6	38.4	61.8	40.6
MoCo v3*	300	×	√	43.3	64.9	46.8	38.8	61.6	41.1
BEiT	300	×	√	35.6	56.7	38.3	32.6	53.3	34.2
CAE	300	×	√	43.8	64.5	47.1	39.0	61.3	41.7
Methods using ViT-B:									
DeiT	300	√	×	46.9	68.9	51.0	41.5	65.5	44.4
MoCo v3*	300	×	√	45.5	67.1	49.4	40.5	63.7	43.4
DINO*	400	×	√	46.8	68.6	50.9	41.5	65.3	44.5
BEiT	300	×	√	39.5	60.6	43.0	35.9	57.7	38.5
BEiT	800	×	√	42.1	63.3	46.0	37.8	60.1	40.6
MAE	300	×	√	45.4	66.4	49.6	40.6	63.4	43.7
MAE	1600	×	√	48.4	69.4	53.1	42.6	66.1	45.9
CAE	300	×	√	48.0	68.7	52.7	42.3	65.6	45.4
CAE	800	×	√	**49.2**	70.2	53.8	**43.3**	67.1	46.6

Contrastive learning. In computer vision, contrastive learning has been popular for self-supervised representation learning (Chen et al., 2020b; He et al., 2020; Tian et al., 2020; Chen et al., 2021; Grill et al., 2020; Caron et al., 2021; Chen & He, 2021; Caron et al., 2020; Wu et al., 2018). The basic idea is to maximize the similarity between the views augmented from the same image and minimize the similarity between the views augmented from different images. Random cropping is an important augmentation scheme, and thus typical contrastive learning methods (e.g., MoCo v3) tend to learn knowledge mainly from the center regions of the original images. Some dense variants (Wang et al., 2021; Xie et al., 2021a) eliminate the tendency in a limited degree by considering an extra contrastive loss with dense patches,

Masked image modeling. Motivated by BERT for masked language modeling (Devlin et al., 2019), the method studied in (Dosovitskiy et al., 2021) and BEiT (Bao et al., 2021) use the ViT structure to solve the masked image modeling task, e.g., predicting the pixels or the discrete tokens. But they do not have explicitly an encoder or a decoder and the ViT structure is essentially a mixture of encoder and decoder, limiting the representation learning quality.

More about recent MIM methods: There are several concurrently-developed methods, such as Masked Autoencoder (MAE) (He et al., 2021), SplitMask (El-Nouby et al., 2021), Masked Feature Prediction (MaskFeat) (Wei et al., 2021), Simple MIM (SimMIM) (Xie et al., 2021b), Perceptual Codebook for BEiT (PeCo) (Dong et al., 2021). Complementary to our study MaskFeat and PeCo improve the pretraining quality by studying the prediction targets. MAE and SplitMask are closely related to BEiT, and can be viewed as a modification of BEiT. They prepend an extra ViT structure that only receives visible patches as a so-called encoder, and then feed the encoded representations and the mask tokens of masked patches into a lightweight ViT structure for prediction. See the computational graph in Appendix C.

The prepended architecture in MAE and SplitMask is only for image understanding (e.g., encoding the image patches), but the lightweight ViT decoder (e.g., containing 8 transformer blocks in MAE) might also have a partial role for representation extraction besides the decoding role. Differently, our approach aims to decouple the two roles. This is empirically verified by the superiority of our CAE over MAE as given in Tables 3 and 4.

The very recent approach data2vec (Baevski et al., 2022) is similar to our approach in making predictions in the latent representation space from the visible patches to the masked patches. Similar to BEiT, data2vec couples the prediction and representation extraction processes together and potentially benefits from our approach, separating the two processes.

6. Conclusion

There are two core points for our CAE architecture design: (i) decouple the encoding (image understanding) and decoding (pretext task completion, making predictions for masked patches) roles and (ii) make predictions in the latent semantic representation space from visible patches to masked

patches. We will study our CAE for the NLP and speech tasks.

We give some analysis about contrastive pretraining and masked image modeling for self-supervised representation learning, as well as supervised pretraining. We speculated that due to strong dependence on random cropping augmentation, typical contrastive learning methods (e.g., MoCo and SimCLR) tend to learn semantics mainly from center patches of the original images and little from non-center patches[3].

Supervised pretraining is similar and also learns the information from the center patches in ImageNet-1K as the instances of the 1000 classes mainly lie in the center. This explains why contrastive pretraining and supervised pretraining perform similarly for downstream tasks.

In contrast, masked image modeling methods care about all the patches, having the potential to learn more information and accordingly perform better for downstream tasks.

Acknowledgements

We would like to acknowledge Hangbo Bao, Xinlei Chen, Li Dong, Qi Han, Zhuowen Tu, Saining Xie, and Furu Wei for the helpful discussions.

References

Asano, Y. M., Rupprecht, C., and Vedaldi, A. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Atito, S., Awais, M., and Kittler, J. Sit: Self-supervised vision transformer. *arXiv preprint arXiv:2104.03602*, 2021.

Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., and Auli, M. data2vec: A general framework for self-supervised learning in speech, visionand languags. *Technical report*, 2022.

Bao, H., Dong, L., and Wei, F. BEiT: BERT pre-training of image transformers. *arXiv:2106.08254*, 2021.

Bardes, A., Ponce, J., and LeCun, Y. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Cai, Z. and Vasconcelos, N. Cascade r-cnn: High quality object detection and instance segmentation. *TPAMI*, 43: 1483–1498, 2021.

[3]It is interesting to study how contrastive learning performs if the objects appear at any position in an image other than at the center for ImageNet-1K.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *ECCV*, pp. 132–149, 2018.

Caron, M., Bojanowski, P., Mairal, J., and Joulin, A. Unsupervised pre-training of image features on non-curated data. In *ICCV*, pp. 2959–2968, 2019.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. Generative pretraining from pixels. In *ICML*, pp. 1691–1703. PMLR, 2020a.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020b.

Chen, X. and He, K. Exploring simple siamese representation learning. In *CVPR*, pp. 15750–15758, 2021.

Chen, X., Xie, S., and He, K. An empirical study of training self-supervised vision transformers. *CoRR*, abs/2104.02057, 2021. URL https://arxiv.org/abs/2104.02057.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. Ieee, 2009.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *NAACL-HLT*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

Dong, X., Bao, J., Zhang, T., Chen, D., Zhang, W., Yuan, L., Chen, D., Wen, F., and Yu, N. Peco: Perceptual codebook for bert pre-training of vision transformers. *arXiv preprint arXiv:2111.12710*, 2021.

Dosovitskiy, A., Springenberg, J. T., Riedmiller, M., and Brox, T. Discriminative unsupervised feature learning with convolutional neural networks. *NeurIPS*, 27:766–774, 2014.

Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M., and Brox, T. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *TPAMI*, 38(9):1734–1747, 2015.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. OpenReview.net, 2021.

El-Nouby, A., Izacard, G., Touvron, H., Laptev, I., Jegou, H., and Grave, E. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

Ermolov, A., Siarohin, A., Sangineto, E., and Sebe, N. Whitening for self-supervised representation learning. In *ICML*, pp. 3015–3024. PMLR, 2021.

Gallinari, P., Lecun, Y., Thiria, S., and Soulie, F. F. Mémoires associatives distribuées: une comparaison (distributed associative memories: a comparison). In *Proceedings of COGNITIVA 87, Paris, La Villette, May 1987*. Cesta-Afcet, 1987.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. Learning representations by predicting bags of visual words. In *CVPR*, pp. 6928–6938, 2020a.

Gidaris, S., Bursuc, A., Puy, G., Komodakis, N., Cord, M., and Pérez, P. Online bag-of-visual-words generation for unsupervised representation learning. *arXiv preprint arXiv:2012.11552*, 2020b.

Goyal, P., Caron, M., Lefaudeux, B., Xu, M., Wang, P., Pai, V., Singh, M., Liptchinsky, V., Misra, I., Joulin, A., and Bojanowski, P. Self-supervised pretraining of visual features in the wild. *arXiv: Computer Vision and Pattern Recognition*, 2021.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *ICCV*, pp. 2961–2969, 2017.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9726–9735. Computer Vision Foundation / IEEE, 2020.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

Henaff, O. Data-efficient image recognition with contrastive predictive coding. In *ICML*, pp. 4182–4192. PMLR, 2020.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313 (5786):504–507, 2006.

Hinton, G. E. and Zemel, R. S. Autoencoders, minimum description length, and helmholtz free energy. *NeurIPS*, 6:3–10, 1994.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *ECCV*, pp. 646–661. Springer, 2016.

Huang, J., Dong, Q., Gong, S., and Zhu, X. Unsupervised deep learning by neighbourhood discovery. In *ICML*, pp. 2849–2858. PMLR, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

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

LeCun, Y. *Mod'eles connexionistes de l'apprentissage*. PhD thesis, Universit´e de Paris VI, 1987.

Li, J., Zhou, P., Xiong, C., and Hoi, S. C. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *ECCV*, pp. 740–755. Springer, 2014.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, pp. 69–84. Springer, 2016.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. In *CVPR*, pp. 2536–2544, 2016.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In Meila, M. and Zhang, T. (eds.), *ICML*, volume 139, pp. 8821–8831. PMLR, 2021.

Ranzato, M., Poultney, C., Chopra, S., LeCun, Y., et al. Efficient learning of sparse representations with an energy-based model. *NeurIPS*, 19:1137, 2007.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning. *arXiv: Computer Vision and Pattern Recognition*, 2020.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv: Learning*, 2018.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

Wang, X., Zhang, R., Shen, C., Kong, T., and Li, L. Dense contrastive learning for self-supervised visual pretraining. In *CVPR*, pp. 3024–3033, 2021.

Wei, C., Fan, H., Xie, S., Wu, C.-Y., Yuille, A., and Feichtenhofer, C. Masked feature prediction for self-supervised visual pre-training. *arXiv preprint arXiv:2112.09133*, 2021.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pp. 3733–3742, 2018.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., and Sun, J. Unified perceptual parsing for scene understanding. In *ECCV*, pp. 418–434, 2018.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *ICML*, pp. 478–487. PMLR, 2016.

Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., and Hu, H. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *CVPR*, pp. 16684–16693, 2021a.

Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., Dai, Q., and Hu, H. Simmim: A simple framework for masked image modeling. *arXiv preprint arXiv:2111.09886*, 2021b.

Yang, J., Parikh, D., and Batra, D. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pp. 5147–5156, 2016.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv: Computer Vision and Pattern Recognition*, 2019.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2017.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *ECCV*, pp. 649–666. Springer, 2016.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. Scene parsing through ade20k dataset. In *CVPR*, pp. 633–641, 2017.

Zhuang, C., Zhai, A. L., and Yamins, D. Local aggregation for unsupervised learning of visual embeddings. In *ICCV*, pp. 6002–6012, 2019.

A. Training Details

Pretraining. The settings are almost the same as BEiT (Bao et al., 2021). We use AdamW (Loshchilov & Hutter, 2017) for optimization and train the CAE for 300 (800) epochs with the batch size being 2048. We set the learning rate as 1.5e-3, with cosine learning rate decay and a 10-epoch warmup, and set the weight decay as 0.05. We do not employ drop depth (Huang et al., 2016) and dropout.

Fine-tuning on ImageNet. We follow the fine-tuning protocol in BEiT to use layer-wise learning rate decay, weight decay and AdamW. The batch size is 4096, the warmup epoch is 10 and the weight decay is 0.05. For ViT-S, we train 200 epochs with learning rate 1.6e-2 and layer-wise decay rate 0.75. For ViT-B, we train 100 epochs with learning rate 8e-3 and layer-wise decay rate 0.65.

Linear probing. We use the LARS (Karpathy et al., 2014) optimizer with momentum 0.9. The model is trained for 90 epochs. The batch size is 16384, the warmup epoch is 10 and the learning rate is 6.4. We adopt an extra BatchNorm layer (Ioffe & Szegedy, 2015) without affine transformation (`affine=False`) before the linear classifier. We do not use mixup (Zhang et al., 2017), cutmix (Yun et al., 2019), drop path (Huang et al., 2016), or color jittering, and we set weight decay as zero.

Attentive probing. The parameters of the encoder are fixed during attentive probing. A cross-attention module, a BatchNorm layer (`affine=False`), and a linear classifier are appended after the encoder. The extra class token representation in cross-attention is learned as model parameters. The keys and the values are the patch representations output from the encoder. There is no MLP or skip connection operation in the extra cross-attention module. We use the SGD optimizer with momentum 0.9 and train the model for 90 epochs. The batch size is 8192, the warmup epoch is 10 and the learning rate is 0.4. Same as linear probing, we do not use mixup, cutmix, drop path, or color jittering, and we set weight decay as zero.

Object detection and instance segmentation on COCO. We utilize multi-scale training and resize the image with the size of the short side between 480 and 800 and the longe side no larger than 1333. The batch size is 32, the learning rate is 3e-4, and the layer-wise decay rate is 0.75. We train the network with the $1\times$ schedule: 12 epochs with the learning rate decayed by $10\times$ at epochs 9 and 11. We do not use multi-scale testing. The Mask R-CNN implementation follows MMDetection (Chen et al., 2019).

Semantic segmentation on ADE20K. We use AdamW as the optimizer. The batch size is 16 and the layer-wise decay rate is 0.65. The input resolution is 512×512. We search from two learning rates, 3e-4 and 4e-4, for all the results in Table 3. We conduct fine-tuning for 160K steps. We do not use multi-scale testing.

Hyperparameter choice. There is a tradeoff variable λ in the loss function given in Equation 1. We did not do an extensive study and only tried two choices, $\lambda = 1$ and $\lambda = 2$. The choice $\lambda = 1$ works also well, slightly worse than $\lambda = 2$. The depths for latent contextual regressor and the decoder are chosen as 4 and 4, which shows better performance than smaller depths. We did not study larger depths that potentially lead to better performance.

B. Details for t-SNE Visualization in Figure 5

We adopt t-SNE (Van der Maaten & Hinton, 2008) to visualize the high-dimensional patch representations output from our CAE encoder on ADE20K (Zhou et al., 2017). ADE20K has a total of 150 categories. For each patch in the image, we set its label to be the category that more than half of the pixels belong to. We collect up to 1000 patches for each category from sampled 500 images.

C. Computational Graph for MAE and SplitMask

We provide the computational graph for Masked autoencoder (He et al., 2021) and SplitMask (El-Nouby et al., 2021) (one stream) in Figure 8. Compared to our CAE, the main issue is that the so-called decoder $\mathcal{R}$ might have also the encoding role, i.e., learning semantic representations of the visible patches.

D. More Results for Concurrently-Developed MIM Methods

Table 5 reports the results of semantic segmentation on ADE20K. The segmentation results are from the corresponding paper. We also report the results of object detection and instance segmentation for MAE under the Cascaded Mask R-CNN

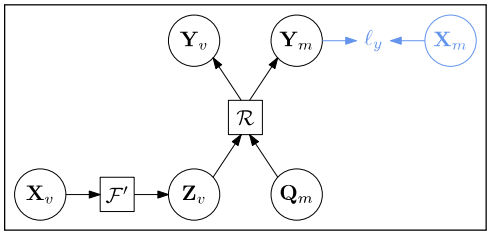

Figure 8: The computational graph for MAE (He et al., 2021) and the one stream in SplitMask (El-Nouby et al., 2021). The two functions, $\mathcal{F}'$ and $\mathcal{R}$, are both based on self-attention. $\mathcal{F}'$ (called encoder in MAE) only processes the visible patches $\mathbf{X}_v$, and $\mathcal{R}$ (called decoder in MAE) processes both the latent representations $\mathbf{Z}_v$ of the visible patches and the mask queries ($\mathbf{Q}_m$) and updates them simultaneously.

framework (Cai & Vasconcelos, 2021). This is different from Table 4 that is based on Mask R-CNN.

Table 5: The results of concurrently-developed MIM methods for semantic segmentation on ADE20K, and object detection and instance segmentation on COCO with the Cascaded Mask-RCNN framework. The segmentation results of other methods are from the corresponding paper, and all the detection results are from our implementation.

Method	Backbone	Dataset	#Epochs	ADE	COCO	
				mIoU	AP^b	AP^m
SplitMask (El-Nouby et al., 2021)	ViT-B	ADE20K	21000	45.7	-	-
MAE (He et al., 2021)	ViT-B	ImageNet-1K	1600	48.1	51.3	44.3
PeCo (Dong et al., 2021)	ViT-B	ImageNet-1K	300	46.7	-	-
CAE	ViT-B	ImageNet-1K	300	47.7	51.0	44.0
CAE	ViT-B	ImageNet-1K	800	48.8	52.3	45.0

E. Image Reconstruction without the Alignment Module

We visualize the reconstructed image for a modification of CAE by dropping the alignment module during pretraining. The visualizations in Figure 9 show that the reconstructed images are noisy and meaningless. The reason is that the input and output representations of latent contextual regressor are not in the same space. As a comparison, the reconstructed images in Figure 4 in the main paper are much better, indicating the importance of the alignment module.

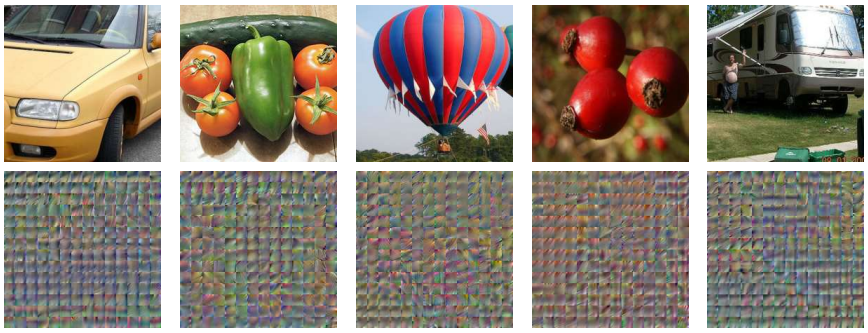

Figure 9: Reconstructed images for the variant of our CAE (the alignment module dropped). Compared to Figure 4 in the main paper, the reconstructed images (bottom row) without alignment are noisy and meaningless. This provides another evidence for that the alignment module is necessary to make the input and output representations of latent contextual regressor be in the same space.

F. More about Attention Maps of MoCo v3 and Our CAE

We show the attention maps of MoCo v3 and our CAE for random crops in Figure 10. This further verifies that MoCo v3 (contrastive pretraining) pretrained on ImageNet-1K tends to attend to the object region, corresponding to the center region of the original image as shown in Figure 6.

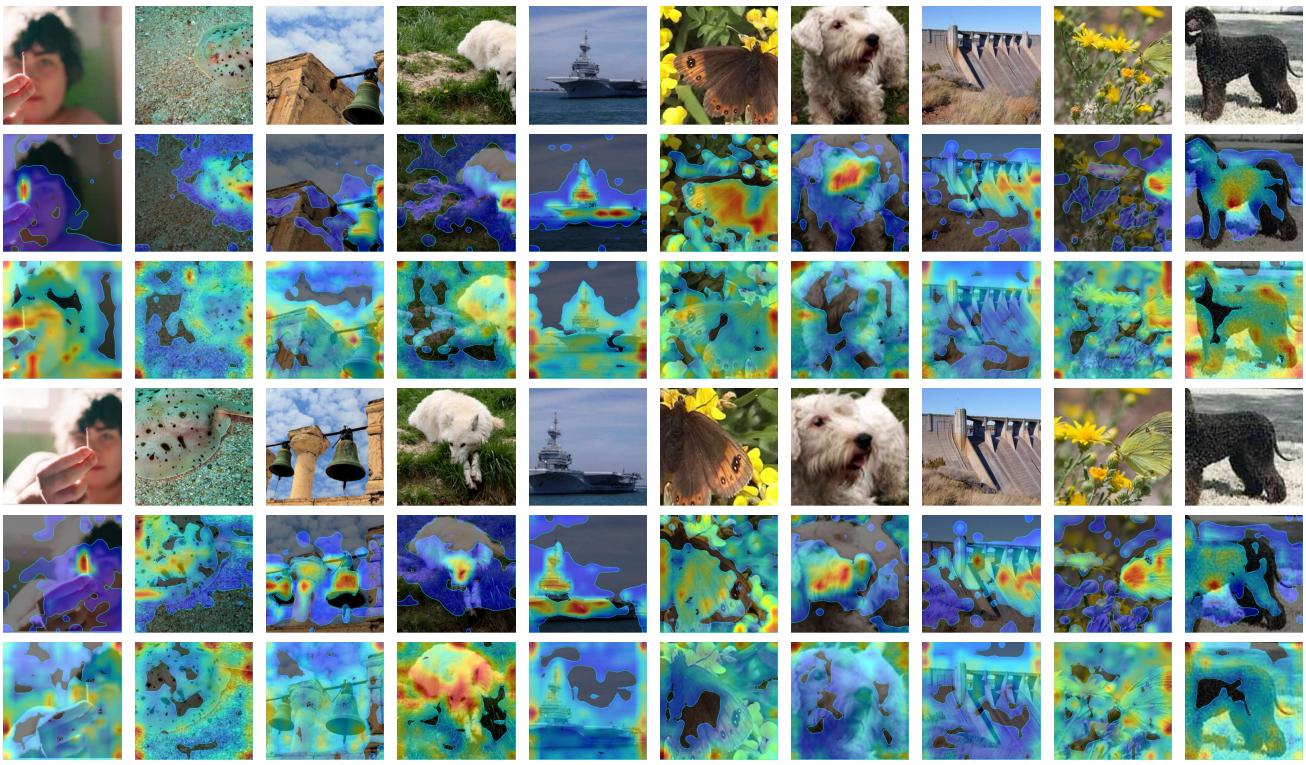

Figure 10: The attention maps over two sets of randomly cropped images (the 1st the 4th rows) for MoCo v3 (the 2nd the 5th rows) and our CAE (the 3rd the 6th rows) pretrained on ImageNet-1K. MoCo v3 tends to focus mainly on the object region and little on other regions. Our CAE tends to consider almost all the patches. The attention maps over the original images are shown in Figure 6.

G. Possible Extensions of Our CAE

Our CAE can benefit from various schemes and tricks exploited in contrastive learning. For example, we can exploit EMA (exponential moving average), used in MoCo, BYOL, and so on. We may replace the encoder for masked patches with the EMA encoder, possibly removing the necessity of the decoder for target prediction.