Readings in Self-Supervised Learning

Emphasis on Computer Vision

Zachary R. McCaw

March 9, 2024

Papers

Note that some papers may not relate to self-supervised learning per se, but rather provide background or ideas that are utilized by self-supervised methods.

1.1 2015: Knowledge Distillation

In [27], the authors observe that soft or probabilistic targets, particularly those with higher entropy, convey more information per example than hard or discrete targets. Define the temperature-normalized softmax as:

$$p_i = \frac{\exp(z_i/\tau)}{\sum_{j=1}^{J}\exp(z_j/\tau)},$$

where z_i is the logit of node i in the output layer, and τ is the temperature hyper-parameter. The standard softmax is recovered by $\tau = 1$, while $\tau > 1$ results in a higher-entropy (equivalently, more uniform) distribution over classes. A teacher network is first trained with a standard softmax. During *distillation*, the temperature is increased such that the teacher produces a soft target for each example. The student network learns to match the output of the teacher, via cross entropy loss, training at the same elevated temperature. After training, the student network operates with $\tau = 1$. When labels are available, the loss may be updated to include the cross entropy between the student's prediction and the true class assignment.

1.2 2015: Adversarial Autoencoders

The X denote the model input and Z a corresponding latent representation. The adversarial autoencoder [34] learns an encoder $q(Z|X)$ such that the *aggregated posterior distribution* $q(Z) = \int q(Z|X)p(X)dX$ matches an arbitrary prior distribution $p(Z)$. Training involves a *reconstruction* step and a *regularization* step. In the reconstruction step, the encoder $q(Z|X)$ and decoder $p(X|Z)$ are trained to minimize the reconstruction error. In the regularization step, the discriminator is first trained to distinguish $Z_+ \sim p(Z)$ from $Z_- \sim q(Z|X)$ with $X \sim p(X)$, then the generator (here, the encoder) is updated to better confuse the discriminator.

1.3 2015: Deep Residual Learning

Sufficiently deep networks suffer a *degradation problem*: performance begins to decrease due to difficulty optimizing rather than over-fitting. In principle, deep layers (i.e. those further from the input) whose capacity is unneeded should learn an identify function. Thus, barring optimizing failures, adding additional layers to a network should never decrease performance. In practice, this is not the case. Deep residual learning [26] makes the identity mapping trivial to learn by adding the input x to a residual block (i.e. group of layers) to its output. Letting $h(x)$ denote the mapping the block would ideally perform, adding the input to the block's output allows the block instead to learn $f(x) = h(x) - x$, i.e. the residual. When the dimension of $h(x)$ differs from x, a linear projection W is applied to x for dimensional consistency: $y = f(x) + Wx$.

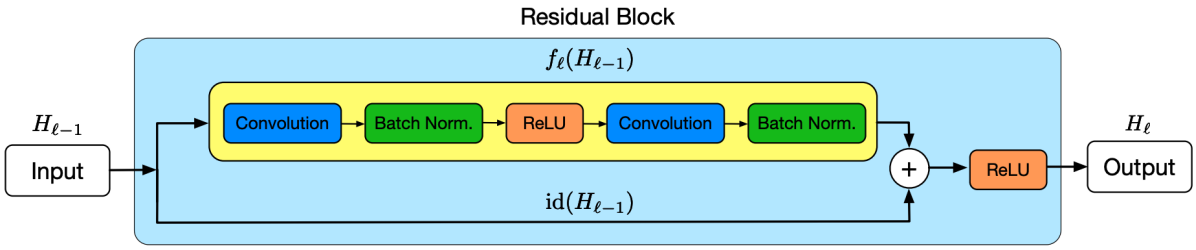


Fig. 1. A close look at the ℓ^{th} ResBlock in a ResNet.

Figure 1. **Source: Deep Networks with Stochastic Depth [28].**

1.4 2016: Stochastic Depth

Vanishing gradients refers to the loss of gradient information in deep networks during backpropagation due to repeated multiplication by small weights. Batch normalization and skip connection [26] are strategies for alleviating this problem. In *stochastic depth* [28], a substantial fraction of layers from a deep network are randomly dropped during training, independently for each mini-batch, resulting in a small expected depth. Similar to dropout, networks trained with stochastic depth can be interpreted as implicit ensembles of networks with different depths. For the lth layer, let B_l denote a Bernoulli random variable with probability p_l, redrawn for each mini-batch. With stochastic depth, the forward pass through the ResNet block in **Figure (1)** becomes:

$$H_l = \mathrm{ReLU}\big\{B_l f_l(H_{l-1}) + id(H_{l-1})\big\}.$$

The authors recommend a linearly decaying survival probability, such that layers closer to the input are retained with higher probability, $p_l = 1 - (l/L)(1 - p_L)$. Here L is the total number of layers, and p_L is the probability of retaining the last layer.

1.5 2016: Context Encoders

The context encoder [36] uses an unsupervised training framework where the model must learn to inpaint the contents of randomly ablated image regions. The context encoder model has an encoder-decoder architecture. The encoder maps from a partially obscured image to a latent embedding. The decoder must reconstruct the entire image, without missing regions, from the embedding. The objective function combines a reconstruction loss and an adversarial loss. The reconstruction loss is simply the L2 distance between the original and reconstructed images. For the adversarial set-up, the context encoder serves as the generator, while the discriminator must learn to distinguish the original and reconstructed images.

1.6 2017: Mask R-CNN

Object detection involves localizing an object within a bounding box. *Semantic segmentation* requires classifying each pixel into a fixed set of categories. *Instance segmentation* involves both detecting all objects present in an image, and segmenting each instance. Mask R-CNN extends Faster R-CNN to perform segmentation [25]. For each object, it predicts a bounding box, the object class, and a binary mask differentiating the object from background. Mask R-CNN first uses a *region proposal network* to identify candidate object bounding boxes. From each region of interest (i.e. area within a bounding box), a fixed-size feature map is extracted. These features become the input to a convolutional neural network (CNN) that performs both object classification and bounding box regression. In parallel to the classification and boxing branch, a fully convolutional network operates on each region of interest to output a binary segmentation mask. Importantly, and in contrast to previous work, the object classification task is decoupled from the segmentation task. An ablation analysis shows that attempting to perform segmentation and classification simultaneously, using a softmax activation, significantly reduces performance.

1.7 2017: Attention

The Transformer [40] is an architecture for processing sequential data that avoids recursion and convolution by making extensive use of attention. In *scaled dot-product attention*, the inputs are queries and keys of dimension d_k and values of dimension d_v. For a single query vector q, the dot product is taken with all keys (k_i), then scaled by $d_k^{-1/2}$, to form the attention score:

$$a_i = \frac{q^T k_i}{\sqrt{d_k}}$$

The attention scores (a_i) are coverted to a set of weights by applying the softmax:

$$w_i = \text{softmax}(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}}$$

The final output is an attention weighted sum of the values:

$$\text{Attention}(q, K, V) = \sum_i w_i v_i.$$

Attention can be calculated for n queries in parallel via:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V.$$

Here Q is an $n \times d_k$ matrix of queries, K is an $n \times d_k$ matrix of keys, V is a $n \times d_v$ matrix of values, and the softmax is applied to the $n \times n$ matrix $d_k^{-1/2} QK^T$ row-wise.

Multi-headed attention forms H different linear projections of the queries, keys, and values. Each set of projections it passed through an attention layer, and the output is described as an *attention head*:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$$

Note that the linear projections of the queries, keys, and values are indexed by the attention head. The H resulting heads are concatenated then linearly projected to the working dimension of the model d_m:

$$\text{MultiHead}(Q, K, V) = c(\text{head}_1, \ldots, \text{head}_H)W,$$

where W is $Hd_v \times d_m$.

In *self-attention*, the queries, keys, and values are all derived from the same source X:

$$Q = XW^Q, \qquad K = XW^K, \qquad V = XW^V.$$

1.8 2017: LARS Optimizer

The *learning rate scaling rule* suggests that if the batch size is increased by a factor of k while keeping the number of epochs fixed, then learning rate (LR) should likewise increase by a factor of k to compensate for performing fewer gradient steps. However, if the LR becomes excessive, training may diverge for increasing batch sizes. To counteract early instability, *warm-up* was proposed, in which the LR is gradually increased to its target maximum over several epochs. In [45], the authors observed that instability arises when the magnitude of an update ($\lambda||\nabla \ell(w)||$ in stochastic gradient descent) is large by comparison to the magnitude of the weights $||w||$. Moreover, the ratio of these quantities varies across layers. The *layer-wise adaptive rate scaling* (LARS) optimizer defines a local learning rate for each layer $\lambda_l = \eta \frac{||w_l||}{||\nabla \ell(w_l)||}$, where $\eta < 1$ is a trust coefficient. LARS is often utilized by contrastive learning methods that benefit from larger batch sizes.

1.9 2018: GPT

The Generative Pretrained Transformer (GPT) [37] is an auto-regressive transformer decoder [40] trained in a two-stage semi-supervised framework. First, the model is *pretrained* in an unsupervised (or self-supervised) manner to perform language modeling (i.e predicting the next token given the previous tokens in the context window) on a large corpus of text. Then, the model is fine-tuned in a supervised manner to perform specific tasks. During fine-tuning, the language modeling loss is included as an auxiliary task.

1.10 2018: UPerNet

[42] define the task of *unified perceptual parsing* as recognition of as many visual concepts as possible from an image. Visual concepts are organized into a hierarchy from scene-, object-, and object part-level, to object materials and textures. The unified perceptual parsing network (UPerNet) [42] is a multi-task model based on a feature pyramid network extractor. The evaluation metrics are pixel accuracy, the proportion of correctly classified pixels, and the mean intersection over union (mIoU) between predicted and ground-truth masks, averaged over classes. For some tasks, notably part segmentation, mIoU should include correct prediction of background. For object detection, multi-tasking does not improve, and slightly degrades, performance. Conversely, for material detection, multi-tasking improves performance, suggesting that knowing the object type is informative when predicting its material.

1.11 2018: BERT

Bidirectional Encoder Representations from Transformers (BERT) [17] uses both preceding and succeeding tokens (i.e. bidirectional context) to learn representations. BERT is based on the the transformer encoder architecture from [40] (this is in contrast to GPT [37], which uses the decoder architecture). Each input sequence starts with a learnable `[CLS]` token, and the final hidden state of this token serves as the input for classification tasks. BERT is pretrained on two tasks:

- *Masked language modeling*, in which $\sim 15\%$ of tokens are randomly selected for replacement. Among these, 80% are replaced with `[MASK]`, 10% are replaced with another token from the vocabulary, and 10% are left unchanged. The model is tasked with predicting the original token from context.
- Next sentence prediction, in which the model is presented 50% of the time with the actual next sentence, and 50% of the time with another random sentence from the corpus.

After pretraining, all parameters of the BERT model are fine-tuned for downstream tasks. The authors argue that large-scale self-supervised pretraining followed by fine-tuning reduces the need for heavily-engineered task-specific architectures.

1.12 2019: RoBERTa

The robustly optimized BERT approach (RoBERTa) [32] is a strategy for training BERT [17] models that attains significant performance improvements. BERT takes as input two token sequences delimited by special tokens:

$$\texttt{[CLS]}\ x_1, \ldots, x_N\ \texttt{[SEP]}\ y_1, \ldots, y_M\ \texttt{[EOS]}.$$

In the original BERT, the two sequences came from sentences sampled contiguously or from distinct documents with equal probability. RoBERTa packs the available input length with contiguous sentences and discards the next sentence prediction task, instead focusing on the masked language modeling task. Whereas the original BERT performed masking once during pre-processing (*static masking*), RoBERTa generates a new masking pattern each time a sequence is provided to the model (*dynamic masking*). Larger batch sizes are found to improve performance, even when the total computational budget for training is fixed. Moreover, evaluation performance continued to improve with additional training.

1.13 2019: ALBERT

ALBERT (A Lite BERT) [30] makes several modifications to BERT [17] that result in improved performance on natural language understanding benchmarks. These include:

- Decoupling the hidden layer size H from the token embedding size E, enabling $H \gg E$.
- Sharing parameters (both feed-forward and attention) across layers by default.
- Replacing the next-sentence prediction (NSP) task by the sentence order prediction (SOP) task.

In the SOP task, the model is presented with consecutive sentences (or more generally, segments) either in the original order (positive example) or in reversed order (negative example). For downstream tasks, SOP leads to improved performance whereas NSP does not. The authors conjecture that in NSP that model is more likely to learn topics prediction (i.e. whether the proposed next sentence is on the same topic as the last) as opposed to coherence (i.e. whether the proposed next sentence logically follows).

1.14 2019: MoCo

The Momentum Contrast [24] approach frames contrastive learning as dictionary lookup. A query encoder f_q encodes the query x_q as $q = f_q(x_q)$. A queue of encoded keys $(k_0, k_1, \dots)$ is maintained, where $k_j = f_k(x_j)$. The query and key are two different "views" of the same image, where a view is derived by cropping the original image and applying random augmentations. A *contrastive loss* is a function that is low when q is similar to its matching, or positive, key k_+ and dissimilar from all other keys. MoCo uses the loss:

$$\ell_q = -\ln \frac{\exp(q \cdot k_+/\tau)}{\sum_{j=0}^{K} \exp(q \cdot k_j/\tau)}, \tag{1}$$

where τ is a temperature hyper-parameter. The sum in the denominator includes 1 positive and K negative examples. Rather than learning separate parameters for the key encoder f_k, the parameters θ_k are an exponential moving average of the query parameters θ_q:

$$\theta_k \leftarrow m \cdot \theta_k + (1 - m) \cdot \theta_q.$$

Here $m \in [0, 1]$ is the momentum coefficient, with $m = 0.999$ by default.

1.15 2019: Noisy Student Training

In Noisy Student Training [43], a *teacher model* is first trained on labeled data, then used to generate *pseudo labels* for unlabeled data. A *student model*, with capacity at least that of the teacher, is trained on both genuine and pseudo labels. In addition, the student must learn in the presence of noise, such as random augmentations of the input and dropout. After training, the student becomes the teacher and the process is iterated. The authors observe that joint training on labeled and pseudo labeled data outperforms first pretraining on unlabeled data then fine-tuning on labeled data.

1.16 2020: SimCLR

SimCLR provides a simple framework for contrastive learning of visual representations [11] (**Figure 2**). A *data augmentation* module randomly transforms a given example x_i, producing two different views $\tilde{x}_i$ and $\tilde{x}_j$. A *base encoder* $f(\cdot)$, such as a ResNet [26], extracts a representation from each view, $h_i = f(\tilde{x}_i)$ and $h_j = f(\tilde{x}_j)$. A light, non-linear *projection head* $g(\cdot)$ maps the representations into the space where the loss is calculated, $z_i = g(h_i)$ and $z_j = g(h_j)$. Finally, a *contrastive loss* is calculated wherein the goal is to identify which pair of views originated from the same example. Specifically, the normalized temperature-scaled cross entropy loss takes the form:

$$\ell_{ij} = -\ln \frac{\exp\left\{\text{sim}(z_i, z_j)/\tau\right\}}{\sum_{k=1}^{2n} \mathbb{I}(k \neq i) \exp\left\{\text{sim}(z_i, z_k)/\tau\right\}}, \tag{2}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function, such as the cosine similarity, and τ is a temperature hyper-parameter.

Observations:

- Performance is significantly improved by including the projection head $g(\cdot)$ between the representation h and the loss calculation. This is likely because training encourages the input z to the loss calculation to be transformation invariant.
- Relative to the supervised setting, self-supervised learning benefits from larger models, larger batch sizes, and longer training.

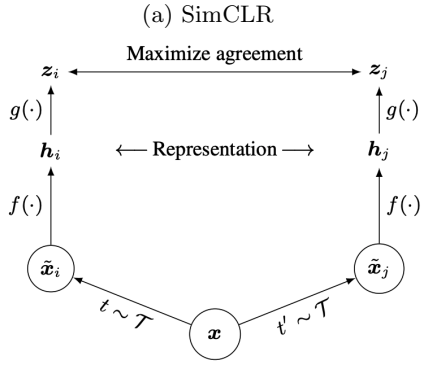


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

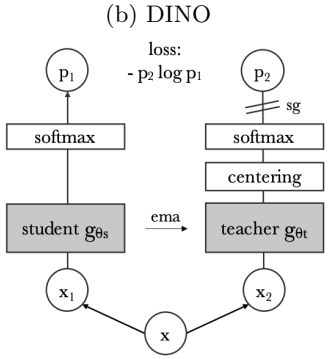


Figure 2: **Self-distillation with no labels.** We illustrate DINO in the case of one single pair of views (x_1, x_2) for simplicity. The model passes two different random transformations of an input image to the student and teacher networks. Both networks have the same architecture but different parameters. The output of the teacher network is centered with a mean computed over the batch. Each networks outputs a K dimensional feature that is normalized with a temperature softmax over the feature dimension. Their similarity is then measured with a cross-entropy loss. We apply a stop-gradient (sg) operator on the teacher to propagate gradients only through the student. The teacher parameters are updated with an exponential moving average (ema) of the student parameters.

Figure 2. **Sources: (a) A Simple Framework for Contrastive Learning of Visual Representations [11]. (b) Emerging Properties in Self-Supervised Vision Transformers [9].**

1.17 2020: Supervised Contrastive Learning

In contrastive self-supervised learning, a single positive pair (e.g. two views of the same image) is contrasted against all negative pairs, resulting in representations of examples belonging to the same class being pulled apart in embedding space. Supervised contrastive learning [29] generalizes the self-supervised case to allow for multiple positive examples, where the additional positives are views from other examples belonging to the same class. On each batch, data augmentation is applied to produce two views of a given example. Similar to SimCLR [11], each view is passed through an encoder network f and a projection network g. The output z of the projection network is unit-normalized, allowing the inner product to measure distances in projection space. The objective function is:

$$\ell = -\sum_{i=1}^{2n} \frac{1}{||\mathcal{P}(i)||} \sum_{p \in \mathcal{P}(i)} \ln \left\{ \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{n \in \mathcal{N}(i)} \exp(z_i \cdot z_n / \tau)} \right\}$$

The lead sum runs over all $2n$ views resulting from augmentation. For a given view i, $\mathcal{P}(i)$ is the set of all indices for positive examples: those views having the same label as i (but excluding i itself). Likewise, $\mathcal{N}(i)$ is the set of all indices for negative examples.

1.18 2020: GPT-3

[7] introduces GPT-3 and develops the idea of *in-context learning*, where the model is provided with K demonstrations of a task at inference time. No updates of the model's weights are performed with these demonstrations; rather, they are provided as context or conditioning. The number of demonstrations K is limited by the size of the model's context window. Zero-shot, one-shot, and few-shot learning are distinguished by whether $K = 0$, $K = 1$, or $K \geq 2$ demonstrations are provided. For some tasks (e.g. general language understanding via SuperGLUE), GPT-3 with few-shot learning achieves performance superior to the fine-tuned state-of-the-art (SOTA), while for other tasks (e.g. reading comprehension via RACE) fine-tuning remains superior.

1.19 2020: SimCLRv2

SimCLR v2 [12] is a semi-supervised learning framework with 3 main steps. First, self-supervised pre-training on unlabeled data is conducted as in v1 [11], but with higher-capacity networks. Second, the model is fine-tuned for a specific task on labeled data. Because a deeper *projection head* is used in v2, fine-tuning is applied starting from a middle layer of $g(\cdot)$ rather than the output of the base encoder $f(\cdot)$. Finally, *knowledge distillation* [27] is performed on the unlabeled data, in which a *student network* is trained to predict the outputs of the fine-tuned *teacher network*. A key observation is that using higher capacity networks for the base encoder leads to better task-specific performance after fine-tuning.

1.20 2020: BYOL

The Bootstrap Your Own Latent (BOYL) [22] framework utilizes two networks, the *online* network and the *target* network. The online network has parameters θ and consists of an encoder $f_\theta(\cdot)$, a projector $g_\theta(\cdot)$, and a predictor $q_\theta(\cdot)$. The target network is similar to the online network, but with two important differences:

1. Reminiscent of MoCo [24], the parameters ξ of the target network are an exponential moving average of the parameters from the online network, $\xi \leftarrow m \cdot \xi + (1 - m) \cdot \theta$, $m \in [0, 1]$.
2. A predictor is not included in the target network.

Starting from an input image x, two random augmentation are applied to produce a pair of views, v and v'. The first view is passed through the online network to produce the representation $y = f_\theta(v)$, the projection $z = g_\theta(y)$, and the prediction $q = q_\theta(z)$. Likewise, the second view is passed through the target network to produce the representation $y' = f_\xi(v')$ and projection $z' = g_\xi(y')$. The prediction q output by the online network is tasked with approximating the projection z' output by the target network. The half-loss is the mean squared error between the normalized prediction $\bar{q}$ and the normalized target representation $\bar{z}'$, $\ell(v, v') = ||\bar{q} - \bar{z}'||_2^2$. As the half-loss is asymmetric, the final loss is the obtained by adding this loss to the half-loss $\ell(v', v)$ obtained by swapping v and v'.

1.21 2020: SwAV

SwAV learns representations by swapping assignments between multiple views of the same image [8]. Given representations $z_1 = f_\theta(x_1)$ and $z_2 = f_\theta(x_2)$ from two different views of the same image x, codes q_1 and q_2 are assigned based on similarity to a learnable set of K *prototype vectors* $\{c_1, \ldots, c_K\}$. The loss is based on the "swapped" prediction problem, where the code assigned to view 2 (i.e. q_2) should be similar to the representation assigned to image 1 (i.e. z_1), and vice versa:

$$L(z_1, z_2) = \ell(z_1, q_2) + \ell(z_2, q_1).$$

The loss is the cross entropy with temperature τ:

$$\ell(z, q) = -\sum_{k=1}^{K} q^{(k)} \ln p^{(k)}, \qquad p^{(k)} = \frac{\exp(z^T c_k/\tau)}{\sum_{k'=1}^{K} \exp(z^T c_{k'}/\tau)}.$$

For a given batch, the codes are computed in such a way (via the Sinkhorn-Knopp algorithm) that examples within a batch are equally partitioned by the prototypes. SwAV also introduces the *multi-crop* strategy, which forms views by taking two standard resolution crops and multiple low-resolution crops.

1.22 2020: W-MSE

[20] proposes the whitening mean square error (W-MSE) loss, an alternative to contrastive loss that does not require negative examples. Given an input image x, $k \geq 2$ positive examples are generated via augmentation. The images are embedded using a base *encoder* f (e.g. ResNet), then mapped into a lower dimensional space via a non-linear *projection head* g. To calculate the loss, the initial outputs $v = g \circ f(x)$ are centered by subtracting the batch mean μ_V, then scaled to have identity covariance using L^{-1}, where LL^T is the Cholesky decomposition of the batch covariance matrix Σ_V:

$$z = L^{-1}(v - \mu_V).$$

For a batch with N images and k positive pairs per image, there are $N\binom{k}{2}$ total positive pairs. The loss is calculated as the sum over these pairs of a distance metric $d(\cdot, \cdot)$ between the representations:

$$\ell_{\text{W-MSE}} = \frac{1}{N\binom{k}{2}} \sum_{ij} d(z_i, z_j).$$

If the whitened representations z are further normalized to have unit length, which is equivalent to projecting them onto the unit hypersphere, then taking $d(\cdot, \cdot)$ as the MSE is equivalent to using the cosine similarity.

1.23 2020: iGPT

The image generative pretrained transformer (iGPT) uses a sequence transformer architecture that predicts pixels instead of tokens [10]. The *autoregressive* objective decomposes the likelihood of a pixel sequence as:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_{i-1}, \ldots, x_1; \theta)$$

for some linear ordering of the pixels in an image, and minimizes $\ell_{\text{AR}} = -\mathbb{E}_X\{\ln p(X)\}$. For comparison, a BERT [17] objective, based on the idea of masked language modeling, is:

$$\ell_{\text{BERT}} = -\mathbb{E}_X \mathbb{E}_M \sum_{i=1}^{n} \ln p(x_i | x_{[1,\ldots,n]\setminus M}),$$

where $M \subset [1, \ldots, n]$ is a subset of pixels randomly selected to be masked, each independently with probability 0.15. The iGPT model utilizes a transformer decoder architecture [40]. For the autoregressive objective, causal attention is applied such that the present pixel can only attend to preceding pixels. For the BERT objective, attention is unmasked. In order to make the context length manageable, images are down-sampled to a low working resolution (e.g. 32×32 to 64×64) in a constrained color space.

iGPT with the autoregressive objective is competitive with other self-supervised models on both the linear probing and fine-tuning evaluations, but requires significantly more parameters than contrastive learning methods (notably, SimCLR [11]). iGPT with the BERT objective under-performs on linear probing, but excels at fine-tuning. Two interesting observations are that:

- For linear probing, embeddings from an intermediate layer, rather than the penultimate layer, typically achieved better performance.
- When fine-tuning, optimizing the sum of the pre-training objetive and the classification objective led to better performance than optimizing the classification objective alone.

1.24 2020: Vision Transformers

The vision transformer (ViT) [18] (**Figure 3**) models an image as a sequence of patches, regarding patches similarly to tokens in NLP tasks. Each patch is flattened and linearly embedded. A learnable `[class]` token is prepended to the patch sequence, whose state is output by the transformer encoder. A learnable 1D position embedding is added to each patch embedding. The transformer encoder applies self-attention to the patch embeddings to capture relationships between patches. The ViT is pretrained to perform image classification on a large data set, then fine-tuned for subsequent tasks.

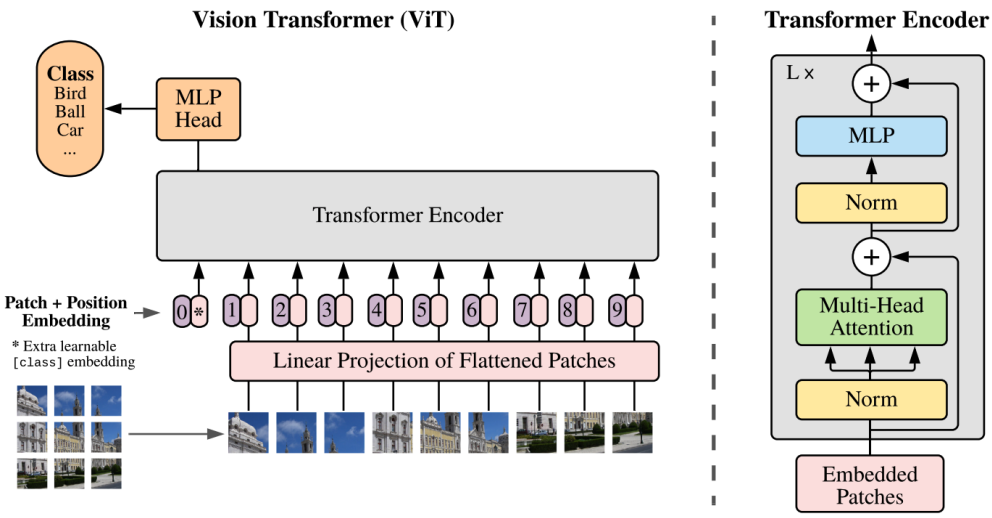


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Figure 3. **Source: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [18].**

1.25 2020: SimSiam

Siamese networks are weight-sharing networks applied to different inputs (or different views of the same input), and are useful for comparing or contrasting examples [14]. SimCLR [11], SwAV [8], and BYOL [22] all use a Siamese architecture, and employ different strategies to prevent collapse of the representations to a trivial (i.e. constant) solution. SimSiam [14] (**Figure 4**) uses an asymmetric Siamese architecture, with an encoder f followed by a predictor h on one branch, and the same encoder followed by a stop-gradient on the other branch. The objective is to minimize the symmetrized negative cosine similarity between $p_1 = h \circ f(x_1)$ and $z_2 = f(x_2)$. The authors observe that the use of the stop-gradient on one branch is essential to avoiding collapse. Although features such as the predictor network, batch normalization, and similarity function affect performance, they do not prevent collapse. Lastly, the authors relate SimSiam to existing architectures. It is comparable SimCLR without the dissimilarity component of the loss; to SwAV without online clustering; and to BOYL without momentum encoding for the target branch.

1.26 2020: DeiT

Data-efficient image Transforms (DeiT) [39] have a similar architecture to the ViT [18], with one addition: a learnable distillation token, analogous to the class token, is appended after the patch sequence (by contrast, the class token is appended before the patch sequence). When training the DeiT (i.e. the student), the availability of a strong teacher is assumed. The loss consists of two parts: the final state of the class token is used to predict the true label, and the final state of the distillation token is used to predict a label provided by a teacher network. Soft distillation, as in [27], where the target is a probability distribution, is distinguished from hard distillation, where the target is a discrete label, namely the argmax of the probability distribution. The authors observe that hard distillation outperforms soft distillation, and that convnets make better teachers than transformers, perhaps because they transfer inductive biases.

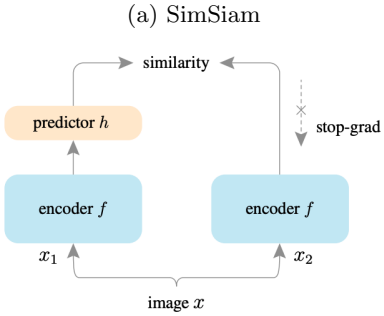


Figure 1. **SimSiam architecture**. Two augmented views of one image are processed by the same encoder network f (a backbone plus a projection MLP). Then a prediction MLP h is applied on one side, and a stop-gradient operation is applied on the other side. The model maximizes the similarity between both sides. It uses neither negative pairs nor a momentum encoder.

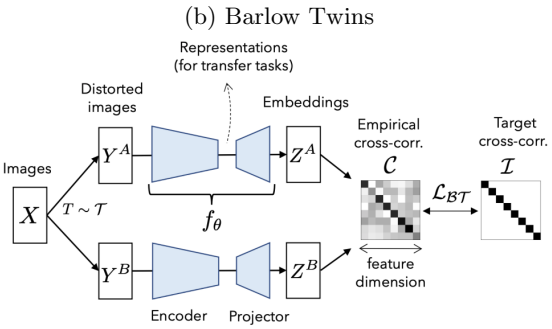


Figure 1. BARLOW TWINS's objective function measures the cross-correlation matrix between the embeddings of two identical networks fed with distorted versions of a batch of samples, and tries to make this matrix close to the identity. This causes the embedding vectors of distorted versions of a sample to be similar, while minimizing the redundancy between the components of these vectors. BARLOW TWINS is competitive with state-of-the-art methods for self-supervised learning while being conceptually simpler, naturally avoiding trivial constant (i.e. collapsed) embeddings, and being robust to the training batch size.

Figure 4. **Sources: (a) Exploring Simple Siamese Representation Learning [14]. (b) Barlow Twins: Self-Supervised Learning via Redundancy Reduction [46].**

1.27 2021: SEER

Most self-supervised learning algorithms are developed and evaluated in the context of the highly curated ImageNet data set. Goyal *et al* [21] investigate whether competitive, high-capacity models can be trained on collections of random (i.e. uncurated) internet images. When fine-tuned and evaluated on ImageNet, models pretrained on internet images do not achieve the performance of models pretrained on ImageNet. However, when evaluated on other data sets (e.g. iNaturalist, Places), models pretrained on internet images achieve superior performance.

1.28 2021: Barlow Twins

Self-supervised learning aims to learn useful representations of the input data without requiring labels. Although often approached by maximizing the similarity between representations of distorted views of the same image, safeguards are needed to prevent the trivial solution of a constant representation [46]. In *Barlow twins* (**Figure 4**), the loss is based on the cross-correlation of representations (Z_A and Z_B) from identical networks applied to two different views of the same image:

$$\ell = \sum_{k=1}^{K}(1 - C_{kk})^2 + \lambda \sum_{k=1}^{K}\sum_{j \neq k} C_{jk}^2$$

where $K = \dim(Z)$ is the embedding dimension, and C_{jk} is the correlation between the jth element of Z_A and the kth element of Z_B calculated across the batch. Intuitively, the loss aims to make the cross-correlation between the representations close to the identity matrix. This *information bottleneck* loss aims to conserve as much information about the example as possible while being minimally informative about the specific distortions applied during training. ResNet [26] is used for the encoder network, followed by a projector network that transforms the output of the encoder network into the space where the loss is calculated.

Methods such as BYOL [22] and SimSiam [14] introduce asymmetry between the paried embeddings to prevent collapse. For example, BYOL applies a predictor network to the online branch, and uses momentum

updates for the parameters of the target branch. Barlow Twins neither requires, nor benefits from, this type of symmetry breaking.

1.29 2021: Swin Transformer

The shifted windows (Swin) transformer [33] builds on ViT [18] through the use of hierarchical feature maps, local attention, and staggered windows in consecutive layers. The image is first split into disjoint patches, where each patch constitutes a visual token whose feature vector is the concatenation of the pixel intensities. Swin transformer blocks, which maintain the number of tokens, are interleaved with merging layers, which concatenate then linearly project the feature vectors of tokens within a group. The standard transformer block computes global self-attention, which is quadratic in the number of tokens. The Swin transformer block uses local self-attention, which is linear in the number of tokens for a fixed window size. Local attention limits the ability of the model to learn long-range dependencies. The provide cross-window attention, the windows of successive Swin transformer blocks are shifted. Lastly, the authors observe a benefit from including a relative position bias in the self-attention calculation.

1.30 2021: DINO

In knowledge distillation with no labels (DINO) [9] (**Figure 2**), a student network is trained to predict the output of a teacher network. The student network $g_{\theta_s}(\cdot)$ and teacher network $g_{\theta_t}(\cdot)$ have the same structure, but the parameters of the teacher network are updated via an exponential moving average $\theta_t \leftarrow m \cdot \theta_t + (1-m) \cdot \theta_s$, $m \in [0, 1]$ (see MoCo [24]). The network's output $g_\theta(x)$ is converted to a probability distribution over K dimensions using the temperature-normalized softmax:

$$P(x) = \frac{\exp\{g_\theta(x)/\tau\}}{\sum_{k=1}^{K} \exp\{g_\theta(x)/\tau\}}.$$

The parameters θ_s of the student network are updated to minimize the cross-entropy between the student and teacher networks:

$$\theta_s \leftarrow \arg\min_{\theta_s} H\{P_t(x), P_s(x)\}.$$

Here $H(a, b) = -a \ln b$. Starting from an input image x, the loss is further specialized by taking two *global views*, x_{g1} and x_{g2}, which are passed only through the teacher, and multiple *local views*, x_{lk}, which are only passed through the student (see multi-crop from SwAV [8]). The final objective is:

$$\ell = \sum_{x_g \in \mathcal{X}_g} \sum_{x_l \in \mathcal{X}_l} H\{P_t(x_g), P_s(x_l)\}.$$

The motivation for this loss is to achieve "local-to-global" correspondence. To avoid collapse, a combination of centering (i.e. updating the mean via an exponential moving average across batches) and sharpening (i.e. applying the softmax with temperature $\tau < 1$) is applied to the teacher network. Similar to SimCLR [11], the network $g_\theta = h_\theta \circ f_\theta$ is composed of a base encoder f_θ (e.g. ViT [18] or ResNet [26]) followed by a projection head h_θ, here a 3-layer MLP.

1.31 2021: MoCo v3

MoCo v3 [15] is an incremental improvement on the momentum contrast framework [24]. Two views x_1 and x_2 of an input image x are generated with random augmentations. Each view is encoded with a query encoder f_q and a key encoder f_k:

$$q_1, q_2 = f_q(x_1), f_q(x_2) \qquad\qquad k_1, k_2 = f_k(x_1), f_k(x_2)$$

The query and key networks are asymmetric. The query network f_q consists of a base encoder, a projection head, and an extra prediction head [22]. The key network f_k consists of the base encoder and projection head only. The parameters of f_q are updated during backpropagation, while the parameters of f_k are updated as

an exponential moving average of f_q. The loss function in (1) is retained, but the memory queue from [24] is discarded in favor of contrasting against the other examples in the same batch. The loss is symmetrized as:

$$\ell = \ell(q_1, k_2) + \ell(q_2, k_1).$$

Observations:

- Freezing the parameters of the ViT's patch projection layer during training (i.e. fixing a random patch projection) improved stability and final accuracy.
- A ViT with a sine-cosine positional embedding outperformed a model with a learned positional embedding. Ablating the positional embedding entirely only decreased ImageNet accuracy 1.6% suggesting that the ViT is not making full use of positional information.
- A model in which the `[class]` token was replaced by global average pooling achieved similar accuracy, suggesting the `[class]` token is non-essential to the ViT.
- Using momentum ($m = 0.99$) contributed a 2.2% increase in accuracy. Using a prediction head in f_q contributed 1% to accuracy, as did replacing replacing layer norm by batch norm in the ViT.

1.32 2021: VICReg

The Variance-Invariance-Covariance regularization (VICReg) scheme [5] is based on a joint embedding architecture (Figure 7) together with a loss composed of three components:

- **Invariance term**: the MSE between two embeddings of the same image, which should be minimized. For N examples, the invariance term is calculated between embeddings z_i and $\tilde{z}_i$ from two views of the same image:

$$\ell_I = \frac{1}{N} \sum_{i=1}^{N} ||z_i - \tilde{z}_i||_2^2.$$

- **Variance term**: a hinge loss which maintains the standard deviation of each embedding dimension above a threshold, encouraging different representations across examples. Specifically:

$$\ell_V = \frac{1}{J} \sum_{j=1}^{J} \max\{0, \gamma - S_\epsilon(z_j)\}$$

  where J is the dimension of the embedding, γ is the target standard deviation, and $S_\epsilon(x) = \sqrt{\mathbb{V}(x) + \epsilon}$. Use of the standard deviation rather than the variance is intentional, and helps to avoid collapse.

- **Covariance term**: a term that attracts the covariances between embedding dimensions towards zero, preventing informational redundancy. Following Barlow Twins [46], the covariance term is:

$$\ell_C = \frac{1}{J} \sum_{j=1}^{J} \sum_{k \neq j} C_{jk}^2,$$

  where C_{jk} denotes the covariance between embedding dimensions j and k.

- The overall loss function is:

$$\ell = \alpha_1 \ell_I(z, \tilde{z}) + \alpha_2 \big\{ \ell_V(z) + \ell_V(\tilde{z}) \big\} + \alpha_3 \big\{ \ell_C(z) + \ell_C(\tilde{z}) \big\}.$$

  Here the αs are weights controlling the relative importance of each term. Note that the variance and covariance losses are applied to each branch separately, while the invariance loss is calculated between the branches.

A symmetrical Siamese architecture with shared weights is adopted, although this is not a requirement. Each branch consists of an *encoder* (e.g. ResNet50), which produces representations, followed by an *expander*, which maps to the space where the loss is calculated. The expander is an MLP which maps to a space $4\times$ the dimension of the encoder (i.e. 2048). For multi-modal tasks, the branches can have different architectures, and the per-branch losses can receive different weights.

1.33 2021: BEiT

Bidirection encoder representation from image transformers (BEiT) [4] proposes *masked image modeling* (MIM), by analogy to the masked language modeling proposed in BERT [17]. An input image is first partitioned into patches and flattened into vectors $(x_i, \ldots, x_N)$, then linearly projected to obtain patch embeddings $(Ex_1, \ldots, Ex_N)$. A learnable start token $e_{[S]}$ is prepended to the input sequence, and a learnable 1D position embedding E_{pos} is added, forming the input to the transformer blocks:

$$H_0 = (e_{[S]}, Ex_1, \ldots, Ex_N) + E_{\text{pos}}$$

The output $H_L = (h_{[S]}, h_1, \ldots, h_N)$ of the transformer blocks serves as the encoding of the image patches. Targets for the pretext task are obtained by passing the image through an *image tokenizer*: a discrete variational autoencoder (dVAE) [38] (the DALL-E tokenizer). The dVAE maps the input image x to a set $z = (z_1, \ldots, z_N)$ of discrete tokens, drawn from a finite vocabulary, then learns to reconstruct x from the visual tokens z. Importantly, there is one token per input patch. To perform MIM, a fraction (40%) of the input embeddings are replaced by a learnable mask embedding $e_{[M]}$, for example:

$$H_0^M = (e_{[S]}, Ex_1, e_{[M]}, Ex_3, e_{[M]}, Ex_5) + E_{\text{pos}}.$$

Rather than choosing patches completely at random, masking is applied to contiguous blocks. The objective is to minimize the evidence lower bound:

$$\sum_{(x_i, x_i^*)} \ln p(x_i|x_i^*) \geq \sum_{(x_i, x_i^*)} \mathbb{E}_{z_i \sim q_\phi(z_i|x_i)}\{\ln p_\psi(x_i|z_i)\} - \mathbb{KL}\{q_\phi(z|x_i), p_\theta(z_i|x_i^*)\}$$

Here x_i^* denotes a masked patch and x_i the corresponding original patch; $q_\phi(z_i|x_i)$ is the encoder of the dVAE, and $p_\psi(x_i|z_i)$ the decoder; finally, $p_\theta(z_i|x_i^*)$ is the model which recovers the visual tokens from the masked image. The model is pre-trained on the MIM task. For down-stream tasks, a task layer is appended after the final transformer layer, and the model is fine-tuned. Key components of the BEiT design are the combination of discrete visual tokens and blockwise masking.

1.34 2021: EsViT

Efficient self-supervised Vision Transformers (EsViT) [31] extends the shifted window (Swin) transformer [33] to the self-supervised setting. Training is similar to DINO [9]. Specifically, a student is trained to match the output of a teacher, and the parameters of the teacher are updated as an exponential moving average of the student. Each network is composed of a base encoder f and a pair of projection heads, h_V for the view-level loss, and h_R for the region-level loss. The projection heads are MLPs with a softmax over the last layer, such that the output is a probability distribution. As in [33], an image is broken into a sequence of patches, which are hierarchically featurized by passage through f. For a given view v, the output of the base encoder $f(v)$ is the feature sequence $(z_1, \ldots, z_T)$, where z_i is the representation for the ith region. The *view-level loss* is calculated from the average feature vector for a view $\bar{z} = \text{mean}\{f(v)\}$:

$$\ell_V = -\frac{1}{|\mathcal{P}|} \sum_{(s,t)\in\mathcal{P}} h_V(\bar{z}_t; \theta_t) \ln h_V(\bar{z}_s; \theta_s).$$

Here $\mathcal{P}$ denotes the set of pairs of student and teacher views, $\bar{z}_s$ denote the average feature vector for the student's view, and $\bar{z}_t$ for the teacher's view. The *region-level loss* operates on individual feature vectors from $(z_1, \ldots, z_T)$:

$$\ell_R = \frac{1}{|\mathcal{P}|} \sum_{(s,t)\in\mathcal{P}} H_R(s,t), \qquad H_R(s,t) = -\frac{1}{T} \sum_{i=1}^{T} h_R(z_{j*}; \theta_t) \ln h_R(z_i; \theta_s),$$

Here $j^* = j^*(i)$ is the index of the teacher feature vector that is most similar to the ith student feature vector. The overall loss $\ell = \ell_V + \ell_R$.

1.35 2021: Masked Autoencoders

- **Topics**: Computer vision.

The masked autoencoder (MAE) [23] strategy divides an image into patches, then randomly samples a subset of patches without replacement for encoding. Masking a high percentage ($\sim 75\%$) is necessary to create a challenging pretraining task. The encoder is a ViT that only operates on the visible or unmasked patches. The decoder accepts both the visible patch encodings and mask tokens, where the mask tokens are shared, learnable vectors. The decoder is only used during pretraining for the image reconstruction task. The loss function is mean squared error in pixel space, calculated only over the masked patches.

1.36 2021: iBOT

Image BERT pretraining with online tokenizer (iBOT) [47] (**Figure 5**) builds on BEiT [4] by learning the tokenizer in parallel with the encoder. iBOT uses a student-teacher setup, where the teacher plays the role of the online tokenizer. Two views, u and v are obtained from an input image x by applying random augmentations. Each view is partitioned into a sequence of patches: $u = (u_i)$ and $v = (v_i)$. Blockwise masking is applied to u and v, replacing some patches with learnable mask tokens. Let u^* and v^* denote the masked sequences. The student network $P_S = h_S \circ f_S$ is composed of a base learner f_S and a patch projector h_S, and the teacher has an analogous architecture. The parameters of the teacher network are an exponential moving average of the parameters of the student network. The output of each is a probability distribution over K dimensions. The masked image modeling (MIM) component of the loss is:

$$\ell_{\text{MIM}} = -\sum_i m_i P_T(u_i)' \ln P_S(u_i^*) - \sum_j m_j P_T(v_j)' \ln P_S(v_j^*).$$

Here the sum runs over the patches within an image, and only the patches that are masked to the student contribute to the MIM loss. Note that the MIM loss compares the student's distribution, given the masked version of a view, with the teacher's distribution, given the unmasked version of the same view. The knowledge distillation component of the objective is:

$$\ell_{\text{KD}} = -P_T(u)' \ln P_S(v^*) - P_T(v)' \ln P_S(u^*).$$

Note that for the KD loss, the student is given the masked version of one view while the teacher is given the unmasked version of the other view.

1.37 2021: SimMIM

In simple masked image modeling (SimMIM), a section of the input image is masked, the masked image is passed through an encoder (e.g. ViT or Swin), then a linear layer predicts the pixel values in the masked region using L_1 regression loss [44]. Each masked patch is represented by a learnable token of the same dimension as the other patches after embedding. To account for down-sampling, a 1×1 convolution is applied to the output of the encoder, reshaping it to have the same dimension as the number of pixels in the masked region (e.g. $32 \times 32 \times 3$ for a color image).

Observations:

- Randomly masking outperforms contiguous, block-wise masking, although performance depends on the mask size and the proportion of patches that are masked. Comparing computer vision with NLP, the optimal masking ratio is higher in vision.
- Using a lightweight (i.e. linear) prediction head leads to performance as good or better than deeper prediction heads on the fine-tuning evaluation. However, the resulting representations are not competative for linear probing.

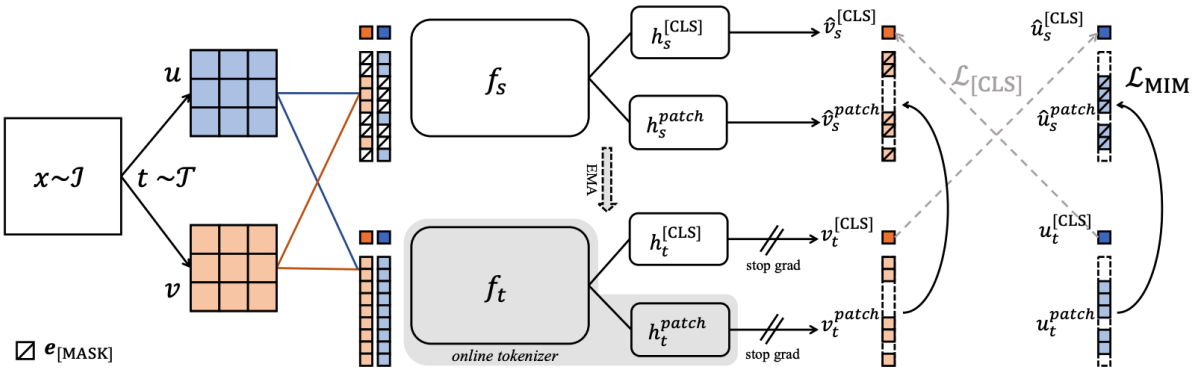


Figure 3: **Overview of iBOT framework, performing masked image modeling with an *online tokenizer*.** Given two views $\boldsymbol{u}$ and $\boldsymbol{v}$ of an image $\boldsymbol{x}$, each view is passed through a teacher network $h_t \circ f_t$ and a student network $h_s \circ f_s$. iBOT minimizes two losses. The first loss $\mathcal{L}_{\texttt{[CLS]}}$ is self-distillation between cross-view `[CLS]` tokens. The second loss $\mathcal{L}_{\text{MIM}}$ is self-distillation between in-view patch tokens, with some tokens masked and replaced by $\boldsymbol{e}_{\texttt{[MASK]}}$ for the student network. The objective is to reconstruct the masked tokens with the teacher networks' outputs as supervision.

Figure 5. **Source: iBOT: Image BERT Pre-Training with Online Tokenizer [47].**

1.38 2021: MaskFeat

Masked feature (MaskFeat) prediction [41] proposes to directly predict features of masked image tokens. Possible targets include pixel intensities, histograms of oriented gradients, tokens from a discrete VAE [38], and embeddings for supervised or self-supervised models (e.g. ResNet50 or ViT-B). The input image is divided into patches, following BEiT [4], blocks of patches are masked by a learnable token, a positional embedding is added, the patch embeddings are passed through a transformer, output tokens for masked patches are linearly projected, adjusting their dimension depending on the target features, and the loss against the targets (typically L2) is calculated. Among featurizations that do not require pre-training, the histogram of oriented gradients provides the best targets. Among featurizations that did require pre-training, ViT-B trained with DINO [9] provided the best targets. When the target features are extracted from a pre-trained model, this model may be viewed as a teacher. Targets extracted from supervised models under-performed.

1.39 2021: RCDM

The Representation Conditional Diffusion Model (RCDM) [6] provides a means to investigate the latent space learned by a self-supervised model. A diffusion model with a UNet architecture is trained to predict the noise which should be removed from an image in order to reverse a progressive corruption process. This UNet is directly conditioned on a representation h of the input image. Once trained, the model can be used to sample the latent space while conditioning on the representation of an input image. Many self-supervised models, including SimCLR and DINO, consist of a *base encoder* followed by a *projector*. By visualizing each latent space, the authors demonstrate that most of the invariances promoted by the augmentations applied during training affect only the final representation (i.e. that obtained following the projector) rather than the base representation. That the base representation retains more information about the initial image explains why it achieves better performance when evaluated by linear probing.

1.40 2021: SplitMask

In SplitMask [19], an image is augmented then split into 16×16 patches. The patches are partitioned into disjoint sets A and B, where patches in B are masked from the perspective of A and vice versa. Each set of patches, (x_a) and (x_b), is processed independently by a shared ViT to produce encodings (z_a) and (z_b). The encodings then pass through a shared shallow ViT to return decodings (y_a) and (y_b). A masked image

modeling (MIM) loss is calculated between the encodings and decodings, i.e. $(x_a) \leftrightarrow (y_a)$ and $(x_b) \leftrightarrow (y_b)$. For example, the MIM task may be to classify each patch into a discrete finite visual vocabular, as in BEiT [4]. In addition, the decodings are mean pooled to produce global representations y_A and y_B. An InfoNCE loss is calculated between these representations:

$$\ell(y_A; y_B) = \frac{\exp(y'_A y_B/\tau)}{\sum_{y \in \{y_B\} \cup N} \exp(y'_A y/\tau)},$$

where N is the set of negatives, composed of the representations from other images in the batch.

Observations:

- Models based on autoencoding a masked image fall within the denoising autoencoder framework, where the masking operation corresponds to adding noise. A key contribution of [19] is demonstrating that denoising autoencoders are data-efficient, and thus capable to learning representations using data sets orders of magnitude smaller than ImageNet.
- An experiment is run in which the DALL-E tokenzier [38] utilized by BEiT is replaced with random projection tokenizer. Let V denote the visual vocabulary size and x a vectorial patch embedding. For $k \in \{1, \ldots, V\}$ let $\boldsymbol{u}_k$ denote a unit vector of the same dimension as v with components sampled uniformly at random. The random projection tokenier is:

  $$k = \arg \max_{k \in \{1,\ldots,V\}} x' u_k.$$

  Performance using the random projection tokenizer is equivalent to that using the DALL-E tokenizer, at a fraction of the computational cost.

1.41 2022: data2vec

data2vec is a general framework for self-supervised learning intended for use with language, speech, and vision [3]. The input is first encoded as a sequence of tokens (e.g. patches in the case of an image). For the student's view, a subset of tokens are randomly replaced with a learnable `[MASK]` token. The student is tasked with predicting the output of a teacher, which receives an unmasked input sequence. Specifically, the student must predict a *contextualized latent representation*: the mean of the top K blocks of the teacher network for the masked tokens. The teacher has the same architecture as the student, and its weights are an exponential moving average of the student's. For computer vision, a ViT architecture is adopted. The objective is the Huber loss between the student and teacher representations. Augmentations are applied, but the student and teacher both see the same modified image. For image classification, the mean-pooling is applied to the last transformer block, which becomes the input to a softmax classification layer.

Observations:

- Ablation studies demonstrate that averaging $K > 1$ layers leads to better performance than predicting only the last layer for all modalities.
- Due to self-attention, the targets take into account broader context. This distinguishes data2vec from methods that learn to reconstruct a local part of the input image.
- Representation collapse, in which all inputs are mapped to similar representations, can be avoided by preventing the teacher weights from changing too quickly.

1.42 2022: Context Autoencoder

The context autoencoder (CAE) utilizes an encoder-regressor-decoder architecture [13]. An input image x is randomly split into two sets of patches: visible x_v and masked x_m. The *encoder* f (i.e. a ViT), maps the visible patches to embeddings $z_v = f(x_v)$. A *regressor* h predicts the embeddings of the masked patches

$z_m = f(x_m)$ from those for the visible patches z_v, conditioned on the positions of the masked patches. This step aligns $h \circ f(z_v)$ with $f(x_m)$. The *decoder* attempts to predict targets y_m for the masked patches from the embeddings predicted by the regressor $\hat{z}_m = h(z_v)$. Both the original masked patches x_m and tokenizations of the masked patches, obtained from the dVAE used in BEiT [4], are considered as targets.

1.43 2022: Masked Siamese Networks

In Masked Siamese Networks (MSNs) [1] (**Figure 6**), random data augmentation is applied to an input image to generate the target view and $M \geq 1$ anchor views. The views are partitioned into patches, and random plus focal masking is applied to the anchor views. An anchor encoder and a target encoder are trained to map from patches to embeddings; the parameters of the target encoder are an exponential moving average of the parameters of the anchor encoder. For MSNs, the encoder is a ViT, and the output is the final representation of the `[CLS]` token. To train the encoder, a distribution of each view is calculated over a set of $K > 1$ learnable prototype vectors. The model is penalized for differences between the target distribution and the anchor distributions. The overall loss is:

$$\ell = \frac{1}{MN} \sum_{i=1}^{N} \sum_{m=1}^{M} H(p_i^+, p_{im}) - \lambda H(\bar{p}).$$

Here N is the number of examples, M is the number of anchors per image, p_i^+ is the distribution over prototypes for the target view, p_{im} is the distribution over prototypes for anchor view m, λ is a penalty parameter, and $\bar{p}$ is the mean distribution of the anchor views:

$$\bar{p} = \frac{1}{MN} \sum_{i=1}^{N} \sum_{m=1}^{M} p_{im}.$$

The penalty term, i.e. $\lambda H(\bar{p})$, is referred to as the *mean entropy maximization regularizer*, and encourages the model to utilize the full set of prototype vectors. MSNs can be viewed as a generalization of DINO [9]. Unlike similar masking-based methods (e.g. SplitMask [19], data2vec [3]), MSNs do not utilize a learnable masking token.

1.44 2023: I-JEPA

Representations learned by predicting masked content require less prior knowledge than methods based on view-invariance (e.g. contrastive learning), but the resulting representations are at a lower semantic level, and under-perform representations learned via view-invariance on linear-probing. Consequently, representations learned by masking typically require end-to-end fine-tuning.

Self-supervised learning architectures fit within the framework of *energy-based models* [2] (see Figure 7). A *joint embedding* architecture aims to output similar embeddings (S_X, S_Y) for compatible views, but requires a mechanism to prevent representation collapse. A *generative* architecture learns to reconstruct the target Y from an encoded input S_X and an auxiliary input Z. For example, in masked learning, S_X is the embedding of the masked image, and Z is often a learnable masking token. Finally, the *joint embedding predictive* architecture is similar to the generative architecture, however the loss is calculated in embedding space rather than input space.

In Image-based Joint-Embedding Predictive Architecture (I-JEPA) [2], the image is split into a sequence of N non-overlapping patches, then passed through a target encoder to obtain representations $s_Y = (s_{Y1}, \ldots, s_{YN})$. M blocks of contiguous patches, possibly overlapping, serve as the targets. Independently, a *context block* is sampled, overlap with the target blocks is removed, then the context is encoded into representations $s_X = (s_{X1}, \ldots, s_{XJ})$. A *predictor* takes as input the representations s_X of the context encoder and a learnable mask token for patch in the target blocks. The task is to predict representations at patches within the target blocks. Let S_{Yj} denote the representation at patch j within block i, and $\hat{S}_{Yj}$ the predicted

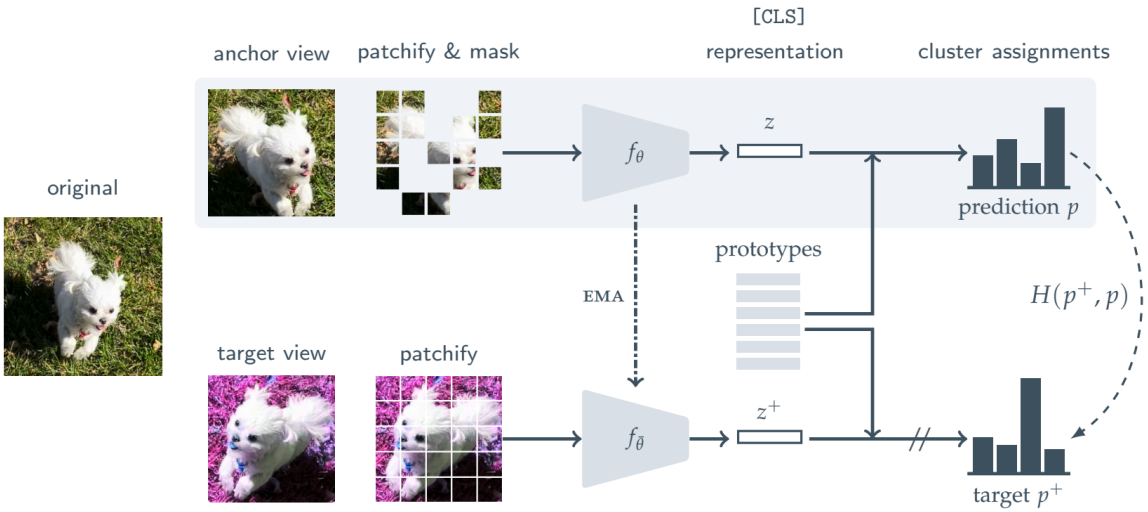


Figure 3: **Masked Siamese Networks.** First use random data augmentations to generate two views of an image, referred to as the anchor view and the target view. Subsequently, a random mask is applied to the anchor view, while the target view is left unchanged. The objective is then to assign the representation of the masked anchor view to the same clusters as the representation of the unmasked target view. A standard cross-entropy loss is used as the criterion to optimize.

Figure 6. **Source: Masked Siamese Networks for Label-Efficient Learning [1].**

representation based on the context representations s_X at a corresponding mask token m_j. Then loss is:

$$\ell = \frac{1}{M} \sum_{i=1}^{M} \sum_{j \in B_i} D\big(\hat{s}_{Yj}, s_{Yj}\big),$$

where D denotes a distance function. The context encoder, target encoder, and predicted all have ViT architectures.

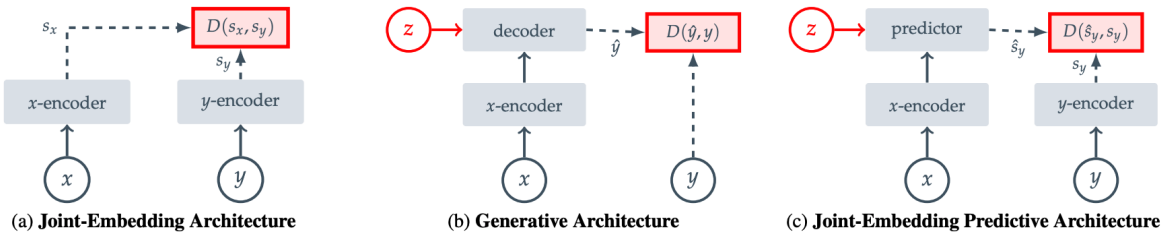
Observations:

- Compared to data2vec [3], MAE [23], and CAE [13], I-JEPA improves performance on linear-probing and fine-tuning evaluations while requiring less computation.
- Computing the loss in pixel space, rather than representation space, significantly decreases performance. Moreover, performance is sensitive to the masking strategy, with multi-block masking achieving the best performance.

1.45 2023: DINO v2

As in the original DINO [9], the image-level objective of DINO v2 [35] consists of training a student network to match the output of a teacher network, which is itself an exponential moving average of the student. Both student and teacher have a ViT architecture. The final state of the `[CLS]` is passed through an MLP projection head to generate a vector of *prototype scores*. This vector is then mapped to a probability distribution via the softmax with Sinkhorn-Knopp (SK) centering. The DINO term of the loss is the cross entropy between the student and teacher distributions:

$$\ell_{\text{DINO}} = -\sum p_t \ln p_s,$$



(a) **Joint-Embedding Architecture** (b) **Generative Architecture** (c) **Joint-Embedding Predictive Architecture**

Figure 2. Common architectures for self-supervised learning, in which the system learns to capture the relationships between its inputs. The objective is to assign a high energy (large scaler value) to incompatible inputs, and to assign a low energy (low scaler value) to compatible inputs. **(a)** Joint-Embedding Architectures learn to output similar embeddings for compatible inputs x, y and dissimilar embeddings for incompatible inputs. **(b)** Generative Architectures learn to directly reconstruct a signal y from a compatible signal x, using a decoder network that is conditioned on additional (possibly latent) variables z to facilitate reconstruction. **(c)** Joint-Embedding Predictive Architectures learn to predict the embeddings of a signal y from a compatible signal x, using a predictor network that is conditioned on additional (possibly latent) variables z to facilitate prediction.

Figure 7. **Source: Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture** [2].

where p_s denotes the student's distribution and p_t the teacher's. An iBOT [47] loss is calculated by randomly masking some of the input patches to the student but not the teacher. For each masked patch, indexed by i, a cross entropy loss is calculated between the student's and the teacher's representations, after applying a softmax with SK centering:

$$\ell_{\text{iBOT}} = -\sum_i p_{ti} \ln p_{si}.$$

Additionally, DINO v2 adds Kozachenko-Leonenko (KoLeo) regularization:

$$\ell_{\text{KoLeo}} = -\frac{1}{n}\sum_{i=1}^{n} \ln(\delta_{ni}), \qquad \delta_{ni} = \min_{j \neq i} ||x_i - x_j||.$$

Observations:

- iBOT suggested sharing parameters between the projection heads used during calculation of ℓ_{DINO} and ℓ_{iBOT}, but DINO v2 finds better performance using different parameters.
- Training a larger model (ViT-g) then distilling to a smaller model (ViT-L) outperforms training the smaller model from scratch.
- Training at high resolution improves performance at the cost of significantly more compute. As a compromise, a fixed amount of training at high resolution is added to the end of pre-training.
- When PCA was performed on the patch features extracted by DINO v2, the first PC separated foreground from background, and subsequent PCs correspond to parts of objects.

1.46 2023: Registers

The goal of self-supervised learning is to learn a genetic feature extractor which can be used for multiple downstream tasks. Darcet *et al* [16] observe that the attention maps from most ViTs, although not the original DINO, contain a small number outlier tokens with high norms scattered throughout the background. The authors hypothesize that the model learns to recognize patches containing little information, and repurposes the corresponding tokens to aggregate global image information while discarding local information. To remedy this behavior, learnable *register* tokens `[reg]` are added after the patch embedding layer, similarly to the `[CLS]` token. At the output, the final states of these tokens are discarded. The optimal number of registers depends on the downstream task; adding the first provides the most benefit, and adding subsequent registers may or may not provide benefit.

Notes

2.1 Evaluation Protocols

Common protocols for evaluating self-supervised learning methods include the following.

- **k-nearest neighbors**: The pre-trained model is frozen, and representation vectors, typically the output of the base encoder f, are generated for all examples in the training data set. For a new example x in the evaluation data set, a represent $z = f(x)$ is calculated from the pre-trained model. The label of x is determined by majority vote among the top k (e.g. 20) nearest neighbors of z from the training data.
- **Linear probing**: The pre-trained model f is frozen and used to generate representations for all examples in the evaluation data set. Then, a simple linear model is trained to perform a down-stream task, such as classification, given the representations.
- **Semi-supervised learning**: Within a large labeled data set, typically ImageNet, the base encoder f is first pre-trained via self-supervised learning. Then, using a subset of the labeled data, typically 1% or 10%, the base model is fine-tuned to perform the evaluation task (e.g. classification).
- **Transfer learning**: The base encoder f is pre-trained via self-supervised learning in a large data set. The model is then transferred to another data set and adapted to perform the task of interest, such as object detection or semantic segmentation. The evaluation may be performed with the base encoder frozen, as in linear probing, or unfrozen, as in fine-tuning.
- **Object detection**: The base encoder f is pre-trained via self-supervised learning, integrated into a Mask R-CNN model [25], then fine-tuned and evaluated on COCO. The evaluation metric is the Average Precision (AP) of the object bounding box.
- **Semantic segmentation**: The base encoder f is pre-trained via self-supervised learning, integrated into a UPerNet model [42], then fine-tuned and evaluated on ADE20K. The evaluation metric is the mean Intersection over Union (mIoU), averaged over semantic categories.

References

[1] M Assran, M Caron, I Misra, et al. Masked siamese networks for label-efficient learning. *arXiv*, 4 2022. https://arxiv.org/abs/2204.07141.

[2] M Assran, Q Duval, I Misra, et al. Self-supervised learning from images with a joint-embedding predictive architecture. *arXiv*, 1 2023. https://arxiv.org/abs/2301.08243.

[3] A Baevski, WN Hsu, Q Xu, et al. data2vec: A general framework for self-supervised learning in speech, vision and language. *arXiv*, 2 2022. https://arxiv.org/abs/2202.03555.

[4] H Bao, L Dong, and F Wei. Beit: Bert pre-training of image transformers. *arXiv*, 6 2021. https://arxiv.org/abs/2106.08254.

[5] A Bardes, J Ponce, and Y LeCun. Variance-invariance-covariance regularization for selfsupervised learning. *arXiv*, 5 2021. https://arxiv.org/abs/2105.04906.

[6] F Bordes, R Balestriero, and P Vincent. High fidelity visualization of what your self-supervised representation knows about. *arXiv*, 12 2021. https://arxiv.org/abs/2112.09164.

[7] TB Brown, B Mann, N Ryder, et al. Language models are few-shot learners. *arXiv*, 5 2020. https://arxiv.org/abs/2005.14165.

[8] M Caron, I Misra, J Mairal, et al. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv*, 6 2020. https://arxiv.org/abs/2006.09882.

[9] M Caron, H Touvron, I Misra, et al. Emerging properties in self-supervised vision transformers. *arXiv*, 4 2021. https://arxiv.org/abs/2104.14294.

[10] M Chen, A Radford, R Child, et al. Generative pretraining from pixels. 7 2020. https://paperswithcode.com/paper/generative-pretraining-from-pixels.

[11] T Chen, S Kornblith, M Norouzi, and G Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, 2 2020. https://arxiv.org/abs/2002.05709.

[12] T Chen, S Kornblith, K Swersky, M Norouzi, and G Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv*, 6 2020. https://arxiv.org/abs/2002.05709.

[13] X Chen, M Ding, X Wang, et al. Context autoencoder for self-supervised representation learning. *arXiv*, 2 2022. https://arxiv.org/abs/2202.03026.

[14] X Chen and K He. Exploring simple siamese representation learning. *arXiv*, 11 2020. https://arxiv.org/abs/2011.10566.

[15] X Chen, S Xie, and K He. An empirical study of training self-supervised vision transformers. *arXiv*, 4 2021. https://arxiv.org/abs/2104.02057.

[16] T Darcet, M Oquab, J Mairal, and P Bojanowski. Vision transformers need registers. *arXiv*, 9 2023. https://arxiv.org/abs/2309.16588.

[17] J Devlin, MW Chang, K Lee, and K Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 10 2018. https://arxiv.org/abs/1810.04805.

[18] A Dosovitskiy, L Beyer, A Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 10 2020. https://arxiv.org/abs/2010.11929.

[19] A El-Nouby, G Izacard, H Touvron, et al. Are large-scale datasets necessary for self-supervised pre-training? *arXiv*, 12 2021. https://arxiv.org/abs/2112.10740.

[20] A Ermolov, A Siarohin, E Sangineto, and N Sebe. Whitening for self-supervised representation learning. *arXiv*, 7 2020. https://arxiv.org/abs/2007.06346.

[21] M Goyal, P Caron, B Lefaudeux, et al. Self-supervised pretraining of visual features in the wild. *arXiv*, 3 2021. https://arxiv.org/abs/2103.01988.

[22] JB Grill, F Strub, F Altche, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv*, 6 2020. https://arxiv.org/abs/2006.07733.

[23] K He, X Chen, S Xie, et al. Masked autoencoders are scalable vision learners. *arXiv*, 11 2021. https://arxiv.org/abs/2111.06377.

[24] K He, H Fan, Y Wu, S Xie, and R Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv*, 11 2019. https://arxiv.org/abs/1911.05722.

[25] K He, G Gkioxari, P Dollar, and R Girshick. Mask r-cnn. *arXiv*, 3 2017. https://arxiv.org/abs/1703.06870.

[26] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. *arXiv*, 12 2015. https://arxiv.org/abs/1512.03385.

[27] G Hinton, O Vinyals, and J Dean. Distilling the knowledge in a neural network. *arXiv*, 3 2015. https://arxiv.org/abs/1503.02531.

[28] G Huang, Y Sun, Z Liu, D l Sedra, and K Weinberger. Deep networks with stochastic depth. *arXiv*, 3 2016. https://arxiv.org/abs/1603.09382.

[29] P Khosla, P Teterwak, C Wang, et al. Supervised contrastive learning. *arXiv*, 4 2020. https://arxiv.org/abs/2004.11362.

[30] Z Lan, M Chan, S Goodman, et al. Albert: A lite bert for self-supervised learning of language representations. *arXiv*, 9 2019. https://arxiv.org/abs/1909.11942.

[31] C Li, J Yang, P Zhang, et al. Efficient self-supervised vision transformers for representation learning. *arXiv*, 6 2021. https://arxiv.org/abs/2106.09785.

[32] Y Liu, M Ott, N Goyal, et al. Roberta: A robustly optimized bert pretraining approach. *arXiv*, 6 2019. https://arxiv.org/abs/1907.11692.

[33] Z Liu, Y Lin, Y Cao, et al. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv*, 3 2021. https://arxiv.org/abs/2103.14030.

[34] A Makhzani, J Shlens, N Jaitly, I Goodfellow, and B Frey. Adversarial autoencoders. *arXiv*, 11 2015. https://arxiv.org/abs/1511.05644.

[35] M Oquab, T Darcet, T Moutakanni, et al. Dinov2: Learning robust visual features without supervision. *arXiv*, 4 2023. https://arxiv.org/abs/2304.07193.

[36] D Pathak, P Krahenbuhl, J Donahue, T Darrell, and AA Efros. Context encoders: Feature learning by inpainting. *arXiv*, 4 2016. https://arxiv.org/abs/1604.07379.

[37] A Radford, K Narasimhan, T Salimans, and I Sutskever. Improving language understanding by generative pre-training. 6 2018. https://paperswithcode.com/paper/improving-language-understanding-by.

[38] A Ramesh, M Pavlov, G Goh, et al. Zero-shot text-to-image generation. *arXiv*, 2 2021. https://arxiv.org/abs/2102.12092.

[39] H Touvron, M Cord, M Douze, et al. Training data-efficient image transformers & distillation through attention. *arXiv*, 12 2020. https://arxiv.org/abs/2012.12877.

[40] A Vaswani, N Shazeer, N Parmar, et al. Attention is all you need. *arXiv*, 6 2017. https://arxiv.org/abs/1706.03762.

[41] C Wei, H Fan, S Xie, et al. Masked feature prediction for self-supervised visual pre-training. *arXiv*, 12 2021. https://arxiv.org/abs/2112.09133.

[42] T Xiao, Y Liu, B Zhou, Y Jiang, and J Sun. Unified perceptual parsing for scene understanding. *arXiv*, 7 2018. https://arxiv.org/abs/1807.10221.

[43] Q Xie, MT Luong, E Hovy, et al. Self-training with noisy student improves imagenet classification. *arXiv*, 11 2019. https://arxiv.org/pdf/1911.04252.pdf.

[44] Z Xie, Z Zhang, Y Cao, et al. Simmim: A simple framework for masked image modeling. *arXiv*, 11 2021. https://arxiv.org/abs/2111.09886.

[45] Y You, I Gitman, and B Ginsburg. Large batch training of convolutional networks. *arXiv*, 8 2017. https://arxiv.org/abs/1708.03888.

[46] J Zbontar, L Jing, I Misra, Y LeCun, and S Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv*, 3 2021. https://arxiv.org/abs/2103.03230.

[47] J Zhou, C Wei, H Wang, et al. ibot: Image bert pre-training with online tokenizer. *arXiv*, 11 2021. https://arxiv.org/abs/2111.07832.