Supplementary material of: ViC-MAE: Self-Supervised Representation Learning from Images and Video with Contrastive Masked Autoencoders

Jefferson Hernandez[1], Ruben Villegas[2], Vicente Ordonez[1]
[1]Rice University, [2]Google DeepMind
{jefehern, vicenteor}@rice.edu, rubville@google.com
Code & Weights: https://github.com/jeffhernandez1995/ViC-MAE

A Implementation Details

ViC-MAE *architecture.* We will release code and model checkpoints, along with the specific training configurations. We followed previous training configurations that also worked well for our models [9, 13]. The pseudocode of ViC-MAE is also provided in Algorithm 1.

Algorithm 1: ViC-MAE PyTorch pseudocode.

```
# V[N, T, C, H, W] - minibatch (T=1 for images)
# tau: temperature coefficient
# clambda: contrastive coefficient

for V in loader:
    # Distant sampling
    f_i, f_j = random_sampling(V)
    # Patch embeddings and position encodings
    x_i = patch_embedd(f)
    x_i += pos_embedd
    # Mask out patches
    x_i, mask_i, ids_restore_i = random_masking(x)
    # Patchify, add pos_embed and mask out f_i ...
    # Forward frames on masked input
    x_i = frame_encoder(x_i) # [N, L_msk, D]
    x_j = frame_encoder(x_j) # [N, L_msk, D]
    # Pool features
    x_pool_i = pooling(x_i) # [N, D]
    x_pool_j = pooling(x_j) # [N, D]
    # Project and normalize
    p_i = l2_normalize(projector(x_pool_i), dim=1)
    z_j = l2_normalize(projector(x_pool_j), dim=1)
    # Predict pixels
    pred = decoder(x_i) # after adding mask tokens

    # compute pixel loss
    target = patchify(f_i)
    loss_pixel = (pred - target) ** 2
    loss_pixel = loss_pixel.mean(dim=-1) # [N, L]
    loss_pixel = (loss * mask).sum() / mask.sum()
    # compute contrastive loss
    loss_cons = ctr(p_i, z_j) + ctr(z_j, p_i)
    # compute final loss
    loss = loss_pixel + clambda * loss_cons


def ctr(p, z):
    # similarity matrix [N, N]
    sim = einsum('nl,nl->nn', p, z) * exp(tau)
    # compute info-nce loss
    labels = range(N) # positives are in diagonal
    loss = cross_entropy_loss(sim, labels)
    return 2 * loss
```

We follow the standard ViT architecture [8], which has a stack of Transformer blocks, each of which consists of a multi-head attention block and an Multi-Layer Perceptron (MLP) block, with Layer Normalization (LN). A linear projection layer is used after the encoder to match the width of the decoder. We use sine-cosine position embeddings for both the encoder and decoder. For the projection and target networks, we do average pooling on the encoder features and follow the architecture of Bardes *et al*. [2], which consists of a linear layer projecting the features up to twice the size of the encoder and two blocks of linear layers that

preserve the size of the features, followed by batch normalization and a ReLU non-linearity.

We extract features from the encoder output for fine-tuning and linear probing. We use the class token from the original ViT architecture, but notice that similar results are obtained without it (using average pooling).

Video Loading. In order to prevent video loading from being a bottleneck on performance due to time spent on video decoding, we leverage the `ffcv` library [15], which we modify to support videos as a list of images in the WebP format. This allows us to significantly surpass the default PyTorch data loaders which can only read data in a synchronous fashion, resulting in the process being blocked until video decoding is complete. The use of `ffcv` allows to perform training without the need of sample repetition as done in OmniMAE [10] and ST-MAE [9] at the cost of a significantly larger storage requirement. We will also release the code for `ffcv` to support videos.

Pre-training. The default settings can be found in Table 1. We do not perform any color augmentation, path dropping or gradient clipping. We initialize our transformer layer using xavier_uniform [11], as it is standard for Transformer architectures. We use the linear learning rate (lr) scaling rule so that lr = *base_ lr*×batchsize / 256 [12].

End-to-end finetuning. We follow common practice for end-to-end finetuning. Default settings can be found in Table 2. Similar to previous work, we use layer-wise lr decay [13].

Linear evaluation. We follow previous work for linear evaluation results [9,13]. As previous work has found we do not use common regularization techniques such as mixup, cutmix, and drop path, and likewise, we set the weight decay to zero [5]. We add an extra batch normalization layer without the affine transformation after the encoder features. Default settings can be found in Table 3.

Table 1: Pre-training setting.

config	value
optimizer	AdamW [17]
base learning rate	1.5e-4
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.95 [3]
batch size	4096
learning rate schedule	cosine decay [16]
warmup epochs [12]	40
epochs	800
augmentation	hflip, crop [0.5, 1]
contrastive loss weight λ	0.025
contrastive loss schedule	0 until epoch 200 then 0.025

Table 2: End-to-end fine-tuning setting.

config	ViT/B	ViT/L
optimizer	AdamW	
optimizer momentum	β_1, β_2=0.9, 0.999	
base learning rate		
IN1K	3e-3	0.5e-3
P65	2e-3	
K400	1.6e-3	4.8e-3
SSv2	1e-3	
weight decay	0.05	
learning rate schedule	cosine decay	
warmup epochs	5	
layer-wise lr decay [1, 6]	0.65	0.75
batch size	1024	768
training epochs		
IN1K	100	50
P65	60	50
K400	150	100
SSv2	40	
augmentation	RandAug (9, 0.5) [7]	
label smoothing [18]	0.1	
mixup [19]	0.8	
drop path [14]	0.1	0.2

B Combining MAE with Negative-Free Methods.

We tried to combine MAE with instance discrimination learning methods by using the [CLS] token of the transformer as a global video feature representation. This representation allows us to use any instance discrimination learning loss without modifications to the underlying ViT transformer encoder. This combination works as follows: Sample two images from a video or an image and its transformed version I_i, I_j and perform patch-level masking. The two inputs are processed by the ViT model f_θ producing token representations $f_\theta(I_i) = \{x_i^{\text{CLS}}, x_i^1, x_i^2, \cdots, x_i^L\}$, where L is the sequence length of the transformer model. This is divided into two disjoint sets. The set $\{x_i^1, x_i^2, \cdots, x_i^L\}$ represents the local features of the input i and are used for masked image modeling following Eq. 1. Then, the x_i^{CLS} token can be used as a global representation with a contrastive loss.

We experiment with this approach using the SimSiam loss [4] and the VicReg loss [2]. We review here these methods and how to combine them with MAEs, but the reader is referred to the original works for a more in-depth explanation of these methods [2, 4].

SimSiam. A combination of SimSiam and MAE, which we refer to as *MAE + SimSiam* uses the x_i^{CLS} token which represents the global video representation as follows: We pass x_i^{CLS} to a projector network $\mathcal{P}$ to obtain $p_i \triangleq \mathcal{P}(x_i^{\text{CLS}})/\|\mathcal{P}(x_i^{\text{CLS}})\|_2$.

Table 3: Linear evaluation setting.

config	value
optimizer	SGD
base learning rate	0.1
weight decay	0
optimizer momentum	0.9
batch size	4096
learning rate schedule	cosine decay
warmup epochs	10
training epochs	90
augmentation	RandomResizedCrop

Table 4: Combining MAE and contrastive methods is not trivial. Linear evaluation on the ImageNet-1K dataset using types of contrastive learning. We use the [CLS] token as the global video representation and apply common contrastive methods, but these do not result on the best performance, which is obtained with our method.

Method	**ImageNet-1K**	
	Top-1	Top-5
MAE [13] + SiamSiam [4]	58.58	82.88
MAE [13] + VicReg [2]	63.86	84.07
ViC-MAE (ours)	**67.66**	**86.22**

A similar procedure is followed for input j, but the global representation is not passed to the projector network $\mathcal{P}$ in order to obtain $z_j \triangleq x_i^{\text{CLS}}/\|x_i^{\text{CLS}}\|_2$. The SimSiam objective is then applied as follows:

$$\mathcal{L}_{p_i,z_j}^{\text{SimSiam}} = \|p_i - z_j\|_2^2 = 2(1 - p_i \cdot z_j). \tag{1}$$

VicReg. A combination of VicReg and MAE, which we refer to as *MAE + VicReg* uses the x_i^{CLS} token which represents the global video representation as follows: We pass it to a projector network $\mathcal{P}$ to obtain $p_i \triangleq \mathcal{P}(x_i^{\text{CLS}})/\|\mathcal{P}(x_i^{\text{CLS}})\|_2$, we repeat this procedure for input j using the target network $\mathcal{T}$ to obtain $z_i \triangleq \mathcal{T}(x_i^{\text{CLS}})/\|\mathcal{T}(x_i^{\text{CLS}})\|_2$. The loss is calculated at the embedding level on p_i and z_j. The inputs are processed in batches, let us denote $P = [p^1, \cdots, p^n]$ and $Z = [z^1, \cdots, z^n]$, where each p^m and z^m are the global representation of video m after the projector network and target network respectively in a batch of size n vectors of dimension d. Let us denote by p_l the vector composed of each value at dimension l in all vectors in P. The variance loss of VicReg is then calculated as follows:

$$v(P) = \frac{1}{d}\sum_{l=1}^{d} \max(0, \gamma - S(p_i, \epsilon)), \tag{2}$$

Table 5: Semi-supervised evaluation on ImageNet. We performed end-to-end finetuning using the settings in 2, but disable RandAug and MixUp for this experiment.

Percentage of data	**5%**	**10%**	**25%**	**50%**	**75%**	**100%**
MAE [13] + SimSiam [4]	7.15	23.41	39.73	54.94	62.88	67.44
MAE [13] + VicReg [2]	47.48	56.63	66.62	73.00	75.29	77.41
ViC-MAE (ours)	**50.25**	**58.22**	**67.65**	**73.97**	**75.80**	**77.89**

where $S(z, \epsilon) = \sqrt{\text{Var}(z) + \epsilon}$ and γ is a constant target value for the standard deviation, fixed to 1. The covariance loss of VicReg can be calculated as:

$$c(P) = \frac{1}{d} \sum_{l \neq k}^{d} [\text{Cov}(p^m)]_{l,k}^2, \tag{3}$$

where $\text{Cov}(p^m) = \frac{1}{N-1} \sum_m (p^m - \bar{p})(p^m - \bar{p})^T$. The final VicReg loss over the batch is defined as:

$$\mathcal{L}_{p_i, z_j}^{\text{VicReg}} = \frac{\lambda}{n} \|p_i - z_j\|_2^2 + \mu \left[v(P) + v(Z) \right] + \nu \left[c(P) + c(Z) \right]. \tag{4}$$

We perform experiments using these two combinations of MAE and contrastive losses as baseline comparisons for our method but found them to be underperforming with only contrastive or only masked methods. In other words, it is not trivial to adapt constrastive learning methods to be used in combination with masked autoencoders. See Table 4 for more details. For the contrastive learning part we experiment with two alternatives.

- *MAE + {SimSiam or VicReg}*. The predictor consists of the backbone network f_θ and a projector followed by a predictor as in Bardes *et al*. [2]. The target encoder consists of the backbone f_θ and the projector, which are shared between the two encoders.
- ViC-MAE. The predictor and the target networks share the same architecture consisting of the backbone network f_θ and a projector following Bardes *et al*. [2].

When using the MAE + {SimSiam or VicReg} combinations, we use the [CLS] token from the ViT architecture which is typically used to capture a global feature from the transformer network and is used to fine-tune the network for downstream tasks such as classification.

Combining MAE with negative-free representation learning is non trivial, and we set to test these by comparing our model with MAE models with alternative negative-free learning objectives Siamsiam [4] and VicReg [2]. We present our results using linear evaluation on Table 4. We use the [CLS] token as the global video representation for contrastive pre-training for 400 epochs. We can notice that competing methods underperform compared to our model which uses pooling of the local features by an absolute margin of $> 3\%$ over the *MAE + VicReg* model.

Semi-supervised evaluation on ImageNet. We also test ViC-MAE against negative-free representation learning methods on the problem of Semi-Supervised evaluation on the ImageNet dataset. The setting consists on training on a subset of the training data and testing on the whole validation data. We chose subsets of size 5%, 10%, 25%, 50%, 75% and 100% of the whole training set of ImageNet. We compare our model against MAE [13] + SimSiam [4], and MAE [13] + VicReg [2]. Results are shown on Table 5, and show the supperiority of ViC-MAE over simple combinations of contrastive learning and masked image modeling.

C Extra Ablations

Temporal representation learning. To investigate the temporal representation learned by our model, we conducted an ablation study on finetuned video models using the K400 dataset, examining performance through standard evaluation, frame shuffling, and frame repetition strategies. We evaluated and averaged 16 random permutations and exhaustive single-frame repetitions. Our findings reveal a decrease in accuracy from the original 87.8% to 78.8% with shuffled frames and 60% with repeated frames, underscoring our model's ability to implicitly learn temporal representations despite not being explicitly designed for temporal modeling. These results demonstrate the model's effectiveness in finetuning for temporal representation learning, highlighting its capacity to capture diverse temporal features.

References

1. Bao, H., Dong, L., Piao, S., Wei, F.: Beit: Bert pre-training of image transformers. In: International Conference on Learning Representations (2021)
2. Bardes, A., Ponce, J., Lecun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In: ICLR 2022-International Conference on Learning Representations (2022)
3. Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International conference on machine learning. pp. 1691–1703. PMLR (2020)
4. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021)
5. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9640–9649 (2021)
6. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: Electra: Pre-training text encoders as discriminators rather than generators. In: International Conference on Learning Representations (2019)
7. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. pp. 702–703 (2020)

8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
9. Feichtenhofer, C., Fan, H., Li, Y., He, K.: Masked autoencoders as spatiotemporal learners. Neural Information Processing Systems (NeurIPS) (2022)
10. Girdhar, R., El-Nouby, A., Singh, M., Alwala, K.V., Joulin, A., Misra, I.: Omnimae: Single model masked pretraining on images and videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10406–10417 (2023)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256. JMLR Workshop and Conference Proceedings (2010)
12. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)
13. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16000–16009 (2022)
14. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. pp. 646–661. Springer (2016)
15. Leclerc, G., Ilyas, A., Engstrom, L., Park, S.M., Salman, H., Madry, A.: FFCV: Accelerating training by removing data bottlenecks. In: Computer Vision and Pattern Recognition (CVPR) (2023), `https://github.com/libffcv/ffcv/`. commit 45f1274
16. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (2016)
17. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2018)
18. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
19. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018)