A. Proof for Theorem 1

Algorithm 1 **Search for the minimal $\mathbf{c}$ and $\mathbf{s_m}$.** $\mathbf{c}$ and $\mathbf{s_m}$ discussed in text can be viewed as the concatenations of vectors in $\mathcal{C}$ and $\mathcal{S}_\mathbf{m}$. LocateParents$(\cdot)$ pins down the locations $\mathcal{Z}$'s parents in the graph. DirectedPaths$(\mathbf{d}, \mathcal{X}^c_\mathbf{m})$ returns the set of variables on the directed paths between $\mathbf{d}$ and $\mathcal{X}^c_\mathbf{m}$.

inputs: The hierarchical graph structure $\mathbf{G}$, and the partitioned observables $\mathcal{X}_\mathbf{m}$, $\mathcal{X}_{\mathbf{m}^c}$.
$\mathcal{C}, \mathcal{S}_\mathbf{m} \leftarrow \emptyset, \emptyset$.
Selection stage:
for $\mathbf{x} \in \mathcal{X}_\mathbf{m}$ **do**
  $\mathcal{Z} \leftarrow \{\mathbf{x}\}$.
  while $\mathcal{Z} \neq \emptyset$ **do**
    $\mathcal{Z}, \mathcal{E} \leftarrow$ LocateParents$(\mathcal{Z})$
    $\mathcal{S}_\mathbf{m} \leftarrow \mathcal{S}_\mathbf{m} \cup \mathcal{E}$
    for $\mathbf{p} \in \mathcal{Z}$ **do**
      if $\mathbf{p} \in$ Ancestors$(\mathbf{x}_{\mathbf{m}^c})$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{p}\}$
        $\mathcal{Z} \leftarrow \mathcal{Z} \setminus \{\mathbf{p}\}$
Pruning stage:
for $\mathbf{d} \in \mathcal{C}$ **do**
  for $\mathbf{d}' \in \mathcal{C} \setminus \{\mathbf{d}\}$ **do**
    if $\mathbf{d}' \in$ DirectedPaths$(\mathbf{d}, \mathcal{X}_{\mathbf{m}^c})$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\mathbf{d}\}$
return $\mathcal{C}, \mathcal{S}_\mathbf{m}$

Algorithm 2 Search for $\mathbf{s}_{\mathbf{m}^c}$ given $\mathcal{C}$. LocateParents$(\mathbf{p})$ pins down the locations $\mathbf{p}$'s parents in the graph.

inputs: The hierarchical graph structure $\mathbf{G}$, the partitioned observables $\mathcal{X}_\mathbf{m}$, $\mathcal{X}_{\mathbf{m}^c}$, and $\mathcal{C}$ returned by Algorithm 1.
$\mathcal{S}_{\mathbf{m}^c} \leftarrow \emptyset$.
for $\mathbf{x} \in \mathcal{X}_{\mathbf{m}^c}$ **do**
  $\mathcal{P}, \mathcal{P}' \leftarrow \{\mathbf{x}\}, \emptyset$.
  while $\mathcal{P} \neq \emptyset$ **do**
    for $\mathbf{p} \in \mathcal{P}$ **do**
      for $\mathbf{p}' \in$ LocateParents$(\mathbf{p})$ **do**
        if $\mathbf{p}'$ is exogenous **then**
          $\mathcal{S}_{\mathbf{m}^c} \leftarrow \mathcal{S}_{\mathbf{m}^c} \cup \{\mathbf{p}'\}$
        else if $\mathbf{p}' \in \mathcal{C}$ **then**
          $\mathcal{S}_{\mathbf{m}^c} \leftarrow \mathcal{S}_{\mathbf{m}^c} \cup ($LocateParents$(\mathbf{p}) \setminus \{\mathbf{p}'\})$
        else
          $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{\mathbf{p}'\}$
    $\mathcal{P} \leftarrow \mathcal{P}'$
return $\mathcal{C}, \mathcal{S}_\mathbf{m}$

Proof. We will directly show that Algorithm 1 returns the minimal set of variables that satisfy all conditions in Theorem 1, which implies its existence. We will then argue that such $\mathcal{C}$ is unique for a specific mask $\mathbf{m}$.

Condition 1: We first discuss the invertibility of $g_{\mathbf{xm}}$. We note that due to the invertibility assumption of the generating process, each backtrack step in Algorithm 1 is invertible (lossless). Thus, it is obvious that the *before* the pruning stage, the mapping between $(\mathcal{C}, \mathcal{S}_\mathbf{m})$ and $\mathcal{X}_\mathbf{m}$ is invertible, as the information of $\mathcal{X}_\mathbf{m}$ is either stored in either $\mathcal{C}$ or $\mathcal{S}_\mathbf{m}$. We now show that the pruning stage does not break this invertibility. To see this, we note that for each $\mathbf{c}$ that is removed in the pruning stage, there exists $\mathbf{c}' \in \mathcal{C}$ on the directed path from $\mathbf{c}$ to $\mathcal{X}_{\mathbf{m}^c}$ (per Algorithm 1). Therefore, $\mathbf{c}$ is a parent/ancestor of $\mathbf{c}'$ and can thus be retrieved by backtracking from $\mathbf{c}'$ thanks to the invertibility of the generating process. Therefore, the mapping between $(\mathcal{C}, \mathcal{S}_\mathbf{m})$ and $\mathcal{X}_\mathbf{m}$ is invertible.

We now address the invertibility of $g_{\mathbf{x}_{\mathbf{m}^c}}$, i.e., the mapping between $(\mathcal{C}, \mathcal{S}_{\mathbf{m}^c})$ and $\mathcal{X}_{\mathbf{m}^c}$. We observe that a similar argument applies: Algorithm 2 dictates that the latent variables from the backtracking from $\mathcal{X}_{\mathbf{m}^c}$ are either stored in either $\mathcal{C}$ or $\mathcal{S}_{\mathbf{m}^c}$. It follows that invertibility of $g_{\mathbf{xm}^c}$.

Condition 2: We show that $(\mathbf{c}, \mathbf{s_m}, \mathbf{s}_{\mathbf{m}^c})$ returned by Algorithm 1 and Algorithm 2 satisfies Condition 2 by contradiction. We suppose that $\mathbf{s_m} \not\perp\!\!\!\perp (\mathbf{c}, \mathbf{s}_{\mathbf{m}^c})$. Then it implied that $\exists \mathbf{d} \in (\mathbf{c}, \mathbf{s}_{\mathbf{m}^c})$, $\exists \boldsymbol{\varepsilon} \in \mathbf{s_m}$, such that $\mathbf{d} \in$ Descendants$(\boldsymbol{\varepsilon})$. More precisely, it followed that there was a directed path that started from $\boldsymbol{\varepsilon}$ and ended at $\mathbf{d}$, and a child of $\boldsymbol{\varepsilon}$, denoted as $\boldsymbol{\delta}$, was located on this path. If $\mathbf{d} \notin$ Descendants$(\boldsymbol{\varepsilon})$, there would be no directed paths from $\boldsymbol{\varepsilon}$ to $\mathbf{d}$ and thus at least one V-structure would sit on each path between $\boldsymbol{\varepsilon}$ and $\mathbf{d}$ that blocked the path. According to Algorithm 1, as $\boldsymbol{\varepsilon} \in \mathbf{s_m}$, it implied that $\boldsymbol{\delta} \notin \mathbf{c}$ and $\boldsymbol{\delta} \notin$ Ancestors$(\mathbf{x_m}) \cap$ Ancestors$(\mathbf{x}_{\mathbf{m}^c})$.

We first investigate the case where $\mathbf{d} \in \mathbf{c}$, i.e., $\mathbf{s_m} \not\perp\!\!\!\perp \mathbf{c}$. The fact that $\mathbf{d} \in \mathbf{c}$ implied that $\mathbf{d} \in$ Ancestors$(\mathbf{x_m}) \cap$ Ancestors$(\mathbf{x}_{\mathbf{m}^c})$ which further implied that $\boldsymbol{\delta} \in$ Ancestors$(\mathbf{x_m}) \cap$ Ancestors$(\mathbf{x}_{\mathbf{m}^c})$ as $\boldsymbol{\delta}$ was an ancestor of $\mathbf{d}$. Therefore, we have arrived at a contraction to the observation that $\boldsymbol{\delta} \in$ Ancestors$(\mathbf{x_m}) \cap$ Ancestors$(\mathbf{x}_{\mathbf{m}^c})$.

We now discuss the scenario where $\mathbf{d} \in \mathbf{s}_{\mathbf{m}^c}$. By design, Algorithm 2 ensures that $\mathbf{s}_{\mathbf{m}^c}$ contains two types of latent variables, exogenous variables and a spouse of latent variables in $\mathbf{c}$. As $\mathbf{s_m}$ consists solely of exogenous variables and exogenous variables are independent mutually, it could only be the case that $\mathbf{d}$ was a spouse of a latent variable in $\mathbf{c}$. By Algorithm 1, there would be a directed path from $\boldsymbol{\delta}$ to $\mathbf{x_m}$. Also, Algorithm 2 ensured that $\mathbf{d}$ lied on a path directed to $\mathbf{x}_{\mathbf{m}^c}$. As there existed a directed path from $\boldsymbol{\delta}$ to $\mathbf{d}$, there must exist a directed path from $\boldsymbol{\delta}$ to $\mathbf{x}_{\mathbf{m}^c}$. Therefore, $\boldsymbol{\delta} \in$ Ancestors$(\mathbf{x_m}) \cap$ Ancestors$(\mathbf{x}_{\mathbf{m}^c})$ which contradicts the fact established above.

Therefore, these contradiction implies that $\mathbf{s_m} \perp\!\!\!\perp (\mathbf{c}, \mathbf{s}_{\mathbf{m}^c})$.

So far, we have shown that Algorithm 1 and Algorithm 2 yield $(\mathbf{c}, \mathbf{s_m}, \mathbf{s}_{\mathbf{m}^c})$ that fulfills the conditions of Figure 3. In the following, we show that $(\mathbf{c}, \mathbf{s_m})$ is the minimal solution and it unique.

Uniqueness and minimality of $(\mathbf{c}, \mathbf{s_m})$: We now reason about that given the mask and the hierarchical structure, $(\mathbf{c}, \mathbf{s_m})$ returned by Algorithm 1 is the set of minimal dimensionality that can fulfill the conditions and such a minimal set is unique.

By construction, Algorithm 1 ensures that for each $\mathbf{c} \in \mathcal{C}$ there exists a (undirected) path that is made up of a directed path from $\mathbf{c}$ to the masked variable $\mathbf{x_m}$ and a directed path from from $\mathbf{c}$ to the unmasked variable $\mathbf{x}_{\mathbf{m}^c}$ and no other $\mathbf{c}' \in \mathcal{C}$ sits on this entire undirected path. To see this, there must exist a directed path from $\mathbf{c}$ to $\mathbf{x_m}$ without any other $\mathbf{c}' \in \mathcal{C}$ on it, otherwise $\mathbf{c}$ would not be placed in $\mathcal{C}$ in Algorithm 1. In addition, the pruning stage of Algorithm 1 mandates that there must exist $\mathbf{x}_{\mathbf{m}^c}$ such that the path from $\mathbf{c}$ to $\mathbf{x}_{\mathbf{m}^c}$ does not contain other $\mathbf{c}' \in \mathcal{C}$. We note that $\mathbf{c}$ chosen by Algorithm 1 is the variable with the smallest possible dimension to block such a path, as it resides on the highest level compared to other variables on the path and the variable dimension increases monotonically along directed paths.

Therefore, the choice of the each $\mathbf{c}$ is minimal and such a choice is unique. As $\mathcal{S}_\mathbf{m}$ is the set of exogenous variables necessary for $\mathcal{C}$ to restore $\mathcal{X}_\mathbf{m}$, the selection of $\mathcal{S}_\mathbf{m}$ is also unique. Hence, we conclude that the $(\mathcal{C}, \mathcal{S}_\mathbf{m})$ returned by Algorithm 1 is the minimal choice and is unique.

□

B. Identifiability proof

Theorem 3. *The generating process (Figure 3) is defined as follows:*

$$[\mathbf{v}_1, \mathbf{v}_2] = g(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2), \tag{3}$$

$$\mathbf{v}_1 = g_1(\mathbf{c}, \mathbf{s}_1), \tag{4}$$

$$\mathbf{v}_2 = g_2(\mathbf{c}, \mathbf{s}_2), \tag{5}$$

where $\mathbf{c} \in \mathcal{C} \subset \mathbb{R}^{d_c}$, $\mathbf{s}_1 \in \mathcal{S} \subset \mathbb{R}^{d_{s_1}}$, *and* $\mathbf{s}_2 \in \mathcal{S}_2 \subset \mathbb{R}^{d_{s_2}}$. *Both* g_1 *and* g_2 *are smooth and have non-singular Jacobian matrices almost anywhere, and* g *is invertible.*

If $\hat{g}_1 : \mathcal{Z} \to \mathcal{V}_1$ *and* $\hat{g}_2 : \mathcal{Z} \to \mathcal{V}_2$ *assume the generating process of the true model* (g_1, g_2) *and match the joint distribution* $p_{\mathbf{v}_1,\mathbf{v}_2}$, *then there is a one-to-one mapping between the estimate* $\hat{\mathbf{c}}$ *and the ground truth* $\mathbf{c}$ *over* $\mathcal{C} \times \mathcal{S} \times \mathcal{S}$, *that is,* $\mathbf{c}$ *is block-identifiable.*

Proof. For $(\mathbf{v}_1, \mathbf{v}_2) \sim p_{\mathbf{v}_1,\mathbf{v}_2}$, because of the matched joint distribution, we have the following relations between the true variables $(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2)$ and the estimated ones $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2)$:

$$\mathbf{v}_1 = g_1(\mathbf{c}, \mathbf{s}_1) = \hat{g}_1(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1), \tag{6}$$

$$\mathbf{v}_2 = g_2(\mathbf{c}, \mathbf{s}_2) = \hat{g}_2(\hat{\mathbf{c}}, \hat{\mathbf{s}}_2), \tag{7}$$

$$(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2) = \hat{g}^{-1}(\mathbf{v}_1, \mathbf{v}_2) = \hat{g}^{-1}(g(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2)) := h(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2), \tag{8}$$

where we define the smooth and invertible function $h := \hat{g}^{-1} \circ g$ that transforms the true variables $(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2)$ to estimates $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2)$.

Plugging Equation 8 into Equation 6 yields the following:

$$g_1(\mathbf{c}, \mathbf{s}_1) = \hat{g}_1(h_{c,s_1}(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2)).$$

For $i \in \{1, \dots, d_{v_1}\}$ and ($j \in \{1, \dots, d_{s_2}\}$), taking partial derivative of the i-th dimension of both sides w.r.t. $s_{2,j}$:

$$0 = \frac{\partial g_{1,i}(\mathbf{c}, \mathbf{s}_1)}{\partial s_{2,j}} = \frac{\partial \hat{g}_{1,i}(h_{c,s_1}(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2))}{\partial s_{2,j}}.$$

The equation equals to zero because there is no $s_{2,j}$ in the left-hand side of the equation. Expanding the derivative on the right-hand side gives:

$$\sum_{k \in \{1, \dots, d_c + d_{s_1}\}} \frac{\partial \hat{g}_{1,i}}{\partial h_{(c,s_1),k}} \cdot \frac{\partial h_{(c,s1),k}}{\partial s_{2,j}}(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2) = 0 \tag{9}$$

For $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1) \in \mathcal{C} \times \mathcal{S} \setminus \mathcal{E}_1$ where $\mathcal{E}_1$ denotes some subset with zero measure, there are at least $d_c + d_{s_1}$ values of i for which vectors $[\frac{\partial \hat{g}_{1,i}}{\partial h_{(c,s_1),1}}(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1), \dots, \frac{\partial \hat{g}_{1,i}}{\partial h_{(c,s_1),d_c+d_{s_1}}}(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1)]$ are linearly independent, which is equivalent to the non-singular Jacobian matrix condition. Therefore, the $(d_c + d_{s_1}) \times (d_c + d_{s_1})$ linear system is invertible and the solution states that:

$$\frac{\partial h_{(c,s_1),k}}{\partial s_{2,j}}(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2) = 0,$$

for any $k \in \{1, \dots, d_c + d_{s_1}\}$, $j \in \{1, \dots, d_{s_2}\}$, and $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1) \in \mathcal{C} \times \mathcal{S} \setminus \mathcal{E}_1$. Therefore, we have shown that h_{c,s_1}, i.e. $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1)$, does not depend on $\mathbf{s}_2$.

Applying the same reasoning to h_{c,s_2}, we can obtain that h_{c,s_2}, i.e. $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_2)$ does not depend on $\mathbf{s}_1$ on $\mathcal{C} \times \mathcal{S}$.

Thus, for $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2) \in \mathcal{C} \times \mathcal{S} \times \mathcal{S}$, we can observe that $\hat{\mathbf{c}}$ does not depend on $\mathbf{s}_1$ and $\mathbf{s}_2$, that is, $\hat{\mathbf{c}} = h_c(\mathbf{c})$.

Notice that in all procedures above, the roles of the true quantities $(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2, g, g_1, g_2)$ and the estimated quantities $(\hat{\mathbf{c}}, \hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \hat{g}, \hat{g}_1, \hat{g}_2)$ are totally symmetric. Therefore, we can switch the two sets of quantities and derive the relation: for $(\mathbf{c}, \mathbf{s}_1, \mathbf{s}_2) \in (\mathcal{C} \times \mathcal{S} \times \mathcal{S})$, $\mathbf{c}$ does not depend on $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$, that is, $\mathbf{c} = h'_c(\hat{\mathbf{c}})$.

In sum, we have shown that on $(\mathcal{C} \times \mathcal{S} \times \mathcal{S})$, there is a one-to-one mapping between $\mathbf{c}$ and $\hat{\mathbf{c}}$.

□

Proof. We invoke Theorem 3 and establish the connection between the MAE training and the estimation model in Theorem 3. In particular, we show that under Assumption 2, any solution produced by the MAE objective satisfies the conditions in Theorem 3 and consequently is equipped with the identifiability guarantee.

We make the following assignments from the MAE configuration to the estimation models in Theorem 3:

- $\mathbf{v}_1 \leftarrow \mathbf{x_m}$;
- $\mathbf{v}_2 \leftarrow \mathbf{x}_{\mathbf{m}^c}$;
- $\hat{g}_1 \leftarrow D_\mathbf{m}(\cdot, \hat{\mathbf{s}}_\mathbf{m})$;
- $\hat{g}_2 \leftarrow \tilde{g}_{\mathbf{m}^c}$, where $E_{\mathbf{m}^c}(\cdot) = [\tilde{g}^{-1}_{\mathbf{m}^c}(\cdot)]_{1:d_c}$.

We can observe that the minimizer of MAE satisfies the conditions specified in Theorem 3. This is because for the optimizer $E_{\mathbf{m}^c}$ of the MAE objective, we can always construct a $\tilde{g}_{\mathbf{m}^c}$, which, together with $D_\mathbf{m}$, matches the joint distribution $p_{\mathbf{x}_m,\mathbf{x}_{\mathbf{m}^c}}$, as stipulated in Theorem 3. Thus, as shown in Theorem 3, there exists a one-to-one mapping between the MAE estimate $\hat{\mathbf{c}} := E_{\mathbf{m}^c}(\mathbf{x}_{\mathbf{m}^c})$ and the true variable $\mathbf{c}$, which concludes our proof. □

C. Experimental Setup

In this section, we provide the details of the experimental setups for our empirical results. Checkpoints and some codes are in https://github.com/martinmamql/mae_understand.

C.1. Masked Autoencoder

Masked Autoencoder (MAE) is an auto-encoding approach based on Vision Transformers (ViT) [17]. It consists of five steps: masking, encoding, unmasking, decoding, and reconstruction. First, an image is divided into non-overlapping patches. Then MAE samples a subset of patches and discards the remaining patches. MAE uses a hyper-parameter, masking ratio, to determine the percentage of patches to discard. For instance, if the masking ratio is 75%, $\frac{3}{4}$ of the patches in an image will be discarded, and only $\frac{1}{4}$ of the patches will be fed into the encoder. The sampling of patches follows a uniform distribution. Next, a ViT encoder first embeds patches using a linear projection with positional embeddings and then uses the processed embeddings to feed into transformer blocks. For decoding, MAE first re-arranges the encoded embeddings from the visible patches according to their corresponding positions in the original image and then uses a shared learned mask token to fill in the patches that are masked. Essentially, this means the input of the decoder is a combination of encoded visible patches and the mask tokens, where the positions of the mask tokens are the masked patches in the original image. The decoder is another lightweight ViT, and it processes the decoder input through transformer blocks. Lastly, the last layer of the decoder linearly projects output patches to pixels, and the pixel output is reshaped to form a reconstruction of the original image. The objective function is the mean squared error between the reconstruction and the original image. MAE has thrived because of its simple design and strong empirical performance.

In the main text, inspired by a follow-up work of MAE [28], we study the effect of masking by decoupling the patch size for masking images and the patch size hyperparameter in the ViT. Particularly, in the main text, we only vary the masking patch size and fix the ViT patch size at 8. Nevertheless, the original MAE [22] does not decouple the two patch sizes. Therefore, for the reference of readers, in Appendix, we provide some analysis and results produced based on the patch size design from the original MAE [22], where the masking patch size and the ViT patch size are equal, and we study three patch sizes: $\{8, 16, 32\}$. The experimental setup in [28] and the setup in [22] are interchangeable except for whether the patch size for the Vision Transformer varies.

C.2. Pretraining and Linear Probing

For pretraining MAE under different masking ratios or patch sizes, we leverage the Tensor Processing Unit (TPU) from Google Cloud. We train separate MAE models for each (masking ratio, patch size) pair, and each pretrained MAE corresponds to a unique masking ratio and patch size. We train all MAEs for 800 epochs. Training time varies, with the shortest (patch size $= 32$) taking 18 hours on a TPU v3-128 Pod, and the longest (patch size $= 8$) taking 40 hours on a TPU v3-128 pod. The architecture follows the exact implementation from the original MAE paper [22], without any hyper-parameter tuning except masking ratio and patch size, which we study in this paper. Details of augmentation, initialization, and base learning rate scaling can be found in the Appendix section of [22], all of which we follow.

After pretraining, we also follow the original MAE work to use linear probing to evaluate the representation quality. After pretraining, we remove the projection layers and add a supervised learning classifier on frozen features of MAE encoders. The decoders are discarded during linear probing. Other details of linear probing can be found in the Appendix section of [22]. We use the same hyper-parameters of linear probing as in [22].

C.3. Reconstructing high-level or low-level representations

To perform reconstruction, we use both the encoder and the decoder from the pretrained MAEs. All samples from ImageNet-1K are passed through the encoder *without* any masking and the decoder reconstructs images in the original input space. Since no masking is applied, no masking token is applied to the input of the decoder. We use the reconstructed images and the original images to perform evaluations of four metrics: SSIM, FSIM, MSE, and PSNR. No training is performed, and the weights of the encoder and the decoder are all frozen.

In Fig. 9, we show the reconstruction analysis using the original patch size design in MAE. Similar to the result in the main text, higher patch sizes produce image reconstructions capturing high-level similarities better, while low patch sizes have reconstructions better on low-level metrics.

C.4. Attention Analysis

We follow the attention heatmap visualization in DINO [10], where the chosen token is the `[CLS]` token or an object-related token. We visualize the self-attention module from the last block of MAE encoder ViT. Brighter colors suggest larger attention weights. For easier visualization, attentions that are below a threshold of activation scores are not shown. We use the same threshold as [10]. For the self-attention visualization on the `[CLS]` token, we use an average of all heads in the last layer of the encoder ViT. For the self-attention visualization of the object-related token, we use the first head of the last layer of the encoder ViT, because using the average attention over all heads will result in a heatmap with much higher overall attention scores across pixels, making the visualization hard to interpret.

C.5. Linear separability

To illustrate the linear separability of different MAEs under varied masking ratios or patch sizes, we sample ten random classes from ImageNet, and then use each MAE encoder to process images in the 10 classes to produce embeddings. We then project embeddings of all samples using PCA to a 50-dimension space before t-SNE, as recommended by [56]. For t-SNE, we use a perplexity of 20.

In Fig. 10, we show the t-SNE plot using the original patch size design in MAE. Similar to the main text, embeddings are more separated in patch sizes 16 and 32 than 8, but differently, there are no significant differences between 16 and 32. Larger patch sizes generate more linearly separable embeddings in this case, although the separability seems indistinguishable for sizes 16 and 32.

For the robustness evaluation, we evaluate different variants of ImageNet validation datasets: ImageNet-v2 (INV2) [52], ImageNet-Adversarial (IN-A) [25], ImageNet-Rendition [4], and ImageNet-Sketch (IN-S) [59]. We also include another object classification dataset, ObjectNet (OJN) [4]. ImageNet-v2 contains three new test sets with 10,000 new images each, sampled a

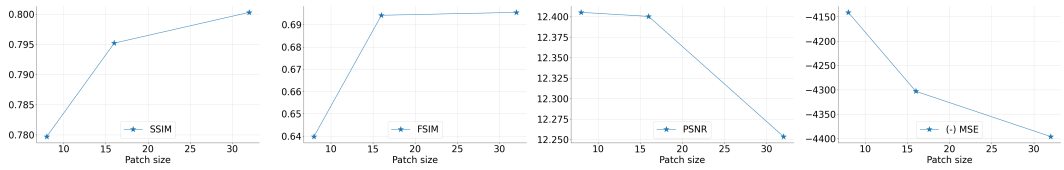

Figure 9. **Reconstruction evaluation** using the validation set without masking, based on two structural-level similarity metrics (SSIM and FSIM) and two pixel-level metrics (PSNR and MSE). We plot negative MSE for easier visualization. Higher SSIM and FSIM indicate high-level information is better captured, while higher PSNR and negative MSE indicates better low-level reconstruction. Here the patch size refers to the patch size in the original MAE, where the masking patch size and the patch size of ViT are equal.

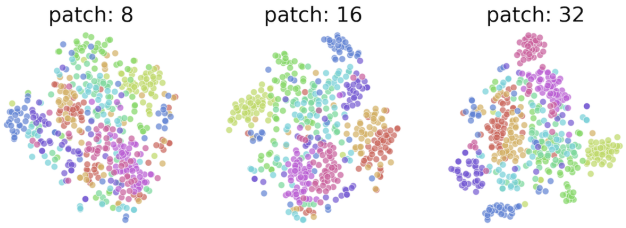

Figure 10. T-SNE embeddings of different MAE models under varied masking ratios and patch sizes. We fix the masking ratio at 0.75 to change patch sizes. Each color represents one ImageNet class. The patch size refers to the patch size in the original MAE, where the masking patch size and the patch size of ViT are equal.

decade after the collection of the original ImageNet dataset, and is independent of existing models to prevent overfitting. ImageNet-Adversarial consists of natural images with adversarial filtration, meaning samples that can be classified with spurious cues are removed. Examples in ImageNet-A are harder to classify correctly and can cause mistakes across various models. ImageNet-Rendition contains renditions of ImageNet classes, such as art, cartoons, graffiti, and paintings. These examples share the same high-level object labels as ImageNet examples but differ in style and texture. ImageNet-Sketch contains black and white images of ImageNet classes, also differing in color and texture compared to original ImageNet samples. ObjectNet is a set of images captured at unusual poses in cluttered, natural scenes, which can severely degrade recognition performance.

Note that for evaluating these datasets, no training is performed; we use the MAE encoders *after* linear probings, therefore the checkpoints that are pretrained and linear-probed on ImageNet, and evaluate the checkpoints on these *validation* datasets without any parameter updates.

In Table 4, we show the robustness analysis using the original patch size design in MAE. A moderate patch size 16 yields the best robustness evaluation on IN-v2, OJN, IN-R, and IN-S. If we follow the original MAE and do not decouple masking patch size and ViT patch size, a medium patch size has stronger robustness performances than extreme patch sizes.

C.6. Shape bias

The cue-conflict dataset was introduced by [19] to evaluate how much deep learning models rely on shape information for prediction, which reflects the model's robustness to spurious correlation

mask ratio	patch size	IN1K	IN-v2	OJN	IN-R	IN-A	IN-S
0.75	8	62.57	49.17	13.44	19.42	3.73	10.73
0.75	16	67.41	54.23	18.24	25.20	3.76	15.51
0.75	32	55.51	42.35	13.46	18.70	1.89	9.48

Table 4. **Accuracy (%) of linear probing and robustness evaluation** on ImageNet variants and ObjectNet. We linear probe MAE via supervised training on IN1K, and then perform inference on IN1K as well as other evaluation sets. We fix the masking ratio at 0.75 to change patch sizes. The patch size refers to the patch size in the original MAE, where the masking patch size and the patch size of ViT are equal.

mask ratio	patch size	AP^{box}	AP^{mask}
0.75	8	34.21	32.28
0.75	16	33.77	32.04
0.75	32	32.39	30.54

Table 5. **COCO object detection and segmentation** using a ViT Mask R-CNN baseline. We fix the masking ratio at 0.75 to change patch sizes. The patch size refers to the patch size in the original MAE, where the masking patch size and the patch size of ViT are equal.

like textures. This dataset consists of 1280 images that are synthesized from 160 images of objects and 48 images of textures. The shape accuracy is measured by the fraction of images that are predicted correctly by their shape. We directly run the pretrained MAE models with linear probes trained on ImageNet-1K on the cue-conflict dataset to examine the representation resulting from MAE pretraining, without any adaptation to the test dataset.

C.7. Transfer learning

We use the pretrained MAE ViT encoder as an FPN [42] backbone in Mask-RCNN [24], following [22]. To do so, [22] uses a stack of pretrained transformer blocks in MAE to produce feature maps at a single scale; for instance, patch size 16 will produce stride 16 features. Then the features are equally divided, and upsampling or downsampling is applied to create features at different scales. Lastly, the FPN is built on multi-scale features. Below we include the transfer learning results of different patch sizes on COCO object detection and segmentation [43]. Because different patch sizes in ViT will influence the scale of feature maps in the FPN, we enforce the same combinations of multi-scale features: i.e., stride 4, 8, 16, and 32.

From Table 5, we show the transfer learning results of MAE under different patch sizes. Patch size 8 performs the best, and patch size 16 is better than 32. The reason for the better perfor-

mance at patch size 8 may be due to a smaller batch size used, compared to patch size 16 and 32 (we can only fit batch size 1 for patch size 8 due to the increased number of tokens to process because of a smaller patch size.) We use the same batch size for 32 and 16, and the comparison between the two supports our claim: an extreme masking scheme can hurt the model's capacity to capture high-level information, or in this case, the semantic understanding of the scene.