A Proof of Theorem 1

We reiterate the setup and notation introduced in the paper here for ease of reference.

Notation $[N]$ denotes the set of natural number $\{1, 2, ..., N\}$. **Id** denotes the (vector-valued) identity function. We write two functions f, g agreeing for all points in set P as $f \equiv_P g$. Finally, we write the total derivative of a vector-valued function $\boldsymbol{f}$ by all its inputs $\boldsymbol{z}$ as $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}$, *i.e.* the Jacobian matrix with entries $\frac{\partial f_i}{\partial z_j}$.

Setup We are given two arbitrary distributions P, Q over latents $\boldsymbol{z} = (\boldsymbol{z}_1, ..., \boldsymbol{z}_K) \in \mathcal{Z}$. Each latent $\boldsymbol{z}_k$ describes one of the K *components* of the final data point $\boldsymbol{x}$ produced by the ground-truth data-generating process $\boldsymbol{f}$. A model $\hat{\boldsymbol{f}}$ is trained to fit the data-generating process on samples of P; the aim is to derive conditions on P and $\hat{\boldsymbol{f}}$ that are sufficient for $\hat{\boldsymbol{f}}$ to then also fit $\boldsymbol{f}$ on Q.

We assume that $\boldsymbol{f}, \hat{\boldsymbol{f}}$ are chosen such that we can find at least one *compositional representations* (Definition 1) for either function that shares a common *composition function* $\boldsymbol{C}$ and factorization of the latent space $\mathcal{Z}_1 \times \cdots \times \mathcal{Z}_K = \mathcal{Z}$.

Proof of Theorem 1. For $\hat{\boldsymbol{f}}$ to generalize to Q, we need to show fitting $\boldsymbol{f}$ on P implies also fitting it on Q, in other words

$$\boldsymbol{f} \underset{P}{\equiv} \hat{\boldsymbol{f}} \implies \boldsymbol{f} \underset{Q}{\equiv} \hat{\boldsymbol{f}} \tag{8}$$

Step 1. Since $\boldsymbol{C}$ is the same for both functions, we immediately get

$$\boldsymbol{\varphi} \underset{Q}{\equiv} \hat{\boldsymbol{\varphi}} \implies \boldsymbol{f} \underset{Q}{\equiv} \hat{\boldsymbol{f}}, \tag{9}$$

i.e. it suffices to show that the *component functions* generalize. Note, however, that since $\boldsymbol{C}$ is not generally assumed to be invertible, we do *not* directly get that agreement of $\boldsymbol{f}, \hat{\boldsymbol{f}}$ on P also implies agreement of their component functions $\boldsymbol{\varphi}, \hat{\boldsymbol{\varphi}}$ on P.

Step 2. We require P to have *compositional support* w.r.t. Q (Definition 2 and Assumption (A2)). The consequence of this assumption is that any point $\boldsymbol{q} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_K) \in Q$ can be constructed from components of the K *support points* $\boldsymbol{p}^k = \left(\boldsymbol{p}_1^k, ..., \boldsymbol{p}_K^k\right) \in P$ subject to $\boldsymbol{p}_k^k = \boldsymbol{q}_k$ as

$$\boldsymbol{q} = \left(\boldsymbol{p}_1^1, ..., \boldsymbol{p}_K^K\right). \tag{10}$$

A trivial consequence, then, is that points $\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}$ in *component space* corresponding to points in Q in latent space can always be mapped back to latents in P

$$\boldsymbol{\varphi}(\boldsymbol{q}) = \left(\boldsymbol{\varphi}_1(\boldsymbol{q}_1), ..., \boldsymbol{\varphi}_K(\boldsymbol{q}_K)\right) = \left(\boldsymbol{\varphi}_1\left(\boldsymbol{p}_1^{(1)}\right), ..., \boldsymbol{\varphi}_K\left(\boldsymbol{p}_K^{(K)}\right)\right) \tag{11}$$

because each *component function* $\boldsymbol{\varphi}_k$ only depends on the latents $\boldsymbol{z}_k$ of a single component. This is also the case for the component functions $\hat{\boldsymbol{\varphi}}$ of $\hat{\boldsymbol{f}}$ so that we get

$$\boldsymbol{\varphi} \underset{P}{\equiv} \hat{\boldsymbol{\varphi}} \implies \boldsymbol{\varphi} \underset{Q}{\equiv} \hat{\boldsymbol{\varphi}}. \tag{12}$$

Step 3. We now only need to show that $\boldsymbol{\varphi} \underset{P}{\equiv} \hat{\boldsymbol{\varphi}}$ follows from $\boldsymbol{f} \underset{P}{\equiv} \hat{\boldsymbol{f}}$. As noted above, this is not guaranteed to be the case, as $\boldsymbol{C}$ is not generally invertible (*e.g.* in the presence of occlusions). We, therefore, need to consider when a unique reconstruction of the component functions $\boldsymbol{\varphi}$ (and correspondingly $\hat{\boldsymbol{\varphi}}$) is possible, based on only the observations $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{z})$ on Q.

As explained in the main paper, we can reason about how a change in the latents $\boldsymbol{z}_k$ of some slot affects the final output, which we can express through the chain rule as

$$\underbrace{\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}_k}(\boldsymbol{z})}_{N \times D} = \underbrace{\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}(\boldsymbol{\varphi}(\boldsymbol{z}))}_{N \times M} \underbrace{\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}(\boldsymbol{z}_k)}_{M \times D} \quad \forall k \in [K]. \tag{13}$$

Here, N is the dimension of the final output (*e.g.* $64 \times 64 \times 3$ for RGB images), M is the dimension of a component's representation $\tilde{\boldsymbol{x}}_k$ (*e.g.* also $64 \times 64 \times 3$ for RGB images), and D is the dimension of a component's latent description $\boldsymbol{z}_k$ (*e.g.* 5: x-position, y-position, shape, size, hue for sprites). Note that we can look at the derivative component-wise because each *component function* $\boldsymbol{\varphi}_k$ only depends on the latents $\boldsymbol{z}_k$ of its component. However, the *combination function* still depends on the (hidden) representation of all components, and therefore $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}$ is a function of all $\boldsymbol{\varphi}$ and the entire $\boldsymbol{z}$.

In equation 13, the left-hand side (LHS) $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}_k}$ can be computed from the training, as long as $\operatorname{supp} P$ is an open set. On the right-hand side (RHS), the functional form of $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}$ is known since $\boldsymbol{C}$ is given, but since $\boldsymbol{\varphi}(\boldsymbol{z})$ is still unknown, the exact entries of this Jacobian matrix are unknown. As such, equation 13 defines a system of partial differential equations (PDEs) for the set of component functions $\boldsymbol{\varphi}$ with independent variables $\boldsymbol{z}$.

Before we can attempt to solve this system of PDEs, we simplify it by isolating $\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}$. Since all terms are matrices, this is equivalent to solving a system of linear equations. For $N = M$, $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}$ is square, and we can solve by taking its inverse as long as the determinant is not zero. In the general case of $N \geq M$, however, we have to resort to the pseudoinverse to write

$$\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}^* = \left(\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}^\top \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k} \right)^{-1} \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}^\top \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}_k} \quad \forall k \in [K], \tag{14}$$

which gives all solutions $\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}^*$ if any exist. This system is overdetermined, and a (unique) solution exists if $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}$ has full (column) rank. In other words, to execute this simplification step on P, we require that for all $\boldsymbol{z} \in P$ the M column vectors of the form

$$\left(\frac{\partial C_1}{\partial \varphi_{km}}\big(\boldsymbol{\varphi}(\boldsymbol{z})\big), ..., \frac{\partial C_N}{\partial \varphi_{km}}\big(\boldsymbol{\varphi}(\boldsymbol{z})\big) \right)^\top \quad \forall m \in [M] \tag{15}$$

are linearly independent. Each entry of a column vector describes how all entries C_n of the final output (*e.g.* the pixels of the output image) change with a single entry φ_{km} of the intermediate representation of component k (*e.g.* a single pixel of the component-wise image). It is easy to see that if even a part of the intermediate representation is not reflected in the final output (*e.g.* in the presence of occlusions, when a single pixel of one component is occluded), the entire corresponding column is zero, and the matrix does not have full rank.

To circumvent this issue, we realize that the LHS of equation 14 only depends on the latents $\boldsymbol{z}_k$ of a single component. Hence, for a given latent $\boldsymbol{z}$ and a slot index k, the correct component function will have the same solution for all points in the set

$$P'(\boldsymbol{z}, k) = \{\boldsymbol{p} \in \operatorname{supp} P | \boldsymbol{p}_k = \boldsymbol{z}_k\}. \tag{16}$$

We can interpret these points as the intersection of P with a plane in latent space at $\boldsymbol{z}_k$ (*e.g.* all latent combinations in the training set in which one component is fixed in a specific configuration). We can then define a modified composition function $\tilde{\boldsymbol{C}}$ that takes $\boldsymbol{z}$ and a slot index k as input and produces a "superposition" of images corresponding to the latents in the subset as

$$\tilde{\boldsymbol{C}}\,(\boldsymbol{z}, k) = \sum_{\boldsymbol{p} \in P'(\boldsymbol{z}, k)} \boldsymbol{C}\big(\boldsymbol{\varphi}(\boldsymbol{p})\big). \tag{17}$$

Essentially, we are condensing the information from multiple points in the latent space into a single function. This enables us to write a modified version of equation 13 as

$$\sum_{\boldsymbol{p} \in P'(\boldsymbol{z}, k)} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}_k}(\boldsymbol{p}) = \sum_{\boldsymbol{p} \in P'(\boldsymbol{z}, k)} \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}\big(\boldsymbol{\varphi}(\boldsymbol{p})\big) \frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}(\boldsymbol{z}_k) = \frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{\varphi}_k}(\boldsymbol{z}, k) \frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}(\boldsymbol{z}_k) \quad \forall k \in [K] \tag{18}$$

Now we can solve for $\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}$ as in equation 14, but this time require only that $\frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{\varphi}_k}$ has full (column) rank for a unique solution to exist, *i.e.*

$$\operatorname{rank} \frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{\varphi}_k}(\boldsymbol{z}, k) = \sum_{\boldsymbol{p} \in P'(\boldsymbol{z}, k)} \frac{\partial \boldsymbol{C}}{\partial \boldsymbol{\varphi}_k}\big(\boldsymbol{\varphi}(\boldsymbol{p})\big) = M \quad \forall \boldsymbol{z} \in P \quad \forall k \in [K]. \tag{19}$$

In general, this condition is easier to fulfill since full rank is not required in any one point but over a set of points. For occlusions, for example, any pixel of one slot can be occluded in some points $\boldsymbol{p} \in P'$, as long as it is not occluded in all of them. We can interpret this procedure as "collecting sufficient information" such that an inversion of the generally non-invertible $\boldsymbol{C}$ becomes feasible locally.

The requirement that $\operatorname{supp} P$ has to be an open set, together with the full rank condition on the Jacobian of the composition function condensed over multiple points, $\tilde{\boldsymbol{C}}$, is termed *sufficient support* in the main paper (Definition 3 and Assumption (A3)). As explained here, this allows for the reconstruction of $\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}$ from the observations, *i.e.*

$$\boldsymbol{f} \underset{P}{\equiv} \hat{\boldsymbol{f}} \implies \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{z}} \underset{P}{\equiv} \frac{\partial \tilde{\boldsymbol{\varphi}}}{\partial \boldsymbol{z}}. \tag{20}$$

Step 4. The above step only gives us agreement of the *derivative* of the component functions, $\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}$, not agreement of the functions themselves. As explained above, the solution to the linear system of equations 14 constitutes a system of partial differential equations (PDEs) in the set of component functions $\boldsymbol{\varphi}$ with independent variables $\boldsymbol{z}$. We can see that this system has the form

$$\partial_i \boldsymbol{\varphi}(\boldsymbol{z}) = \boldsymbol{a}_i\big(\boldsymbol{z}, \boldsymbol{\varphi}(\boldsymbol{z})\big), \tag{21}$$

where $i \in [L] = [K \times D]$ is an index over the flattened dimensions K and D such that $\partial_i \boldsymbol{\varphi}$ denotes $\frac{\partial \boldsymbol{\varphi}}{\partial z_L}$ (which is essentially one column of $\frac{\partial \boldsymbol{\varphi}_k}{\partial \boldsymbol{z}_k}$ aggregated over all k) and $\boldsymbol{a}_i$ is the combination of corresponding terms from the LHS. If this system allows for more than one solution, we cannot uniquely reconstruct the component functions from their derivatives.

If we have access to some initial point, however, for which we know $\boldsymbol{\varphi}(\mathbf{0}) = \boldsymbol{\varphi}^0$, we can write

$$\begin{aligned} \boldsymbol{\varphi}(z_1, ..., z_L) - \boldsymbol{\varphi}^* = &\big(\varphi(z_1, ..., z_L) - \varphi(0, z_2, ..., z_L)\big) \\ &+ \big(\varphi(0, z_2, ..., z_L) - \varphi(0, 0, z_3, ..., z_L)\big) \\ &+ ... \\ &+ \big(\varphi(0, ..., 0, z_L) - \varphi(0, ..., 0)\big). \end{aligned} \tag{22}$$

In each line of this equation, only a single $z_i =: t$ is changing; all other $z_1, ..., z_L$ are fixed. Any solution of 22, therefore, also has to solve the L ordinary differential equations (ODEs) of the form

$$\partial_t \boldsymbol{\varphi}(z_1, ..., z_{i-1}, t, z_{i+1}, ..., z_L) = \boldsymbol{a}_i\big(z_1, ..., z_{i-1}, t, z_{i+1}, ..., z_L, \boldsymbol{\varphi}(z_1, ..., z_{i-1}, t, z_{i+1}, ..., z_L)\big), \tag{23}$$

which have a unique solution if $\boldsymbol{a}_i$ is Lipschitz in $\boldsymbol{\varphi}$ and continuous in z_i, as guaranteed by (A1). Therefore, 22 has at most one solution. This reference point does not have to be in $\boldsymbol{z} = \mathbf{0}$, as a simple coordinate transform will yield the same result for any point in P. It is therefore sufficient that there exists *some* point $\boldsymbol{p}^0 \in P$ for which $\boldsymbol{\varphi}(\boldsymbol{p}^0) = \hat{\boldsymbol{\varphi}}(\boldsymbol{p}^0)$ to obtain the same unique solution for $\boldsymbol{\varphi}$ and $\hat{\boldsymbol{\varphi}}$, which is exactly what (A4) states. Overall, this means that agreement of the derivatives of the component functions also implies agreement of the component functions themselves, *i.e.*

$$\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{z}} \underset{P}{\equiv} \frac{\partial \tilde{\boldsymbol{\varphi}}}{\partial \boldsymbol{z}} \implies \boldsymbol{\varphi} \underset{P}{\equiv} \hat{\boldsymbol{\varphi}} \tag{24}$$

Step 5. Finally, we can conclude the model $\hat{\boldsymbol{f}}$ fitting the ground-truth generating process $\boldsymbol{f}$ on the training distribution P, through 20, 24, 12, 9, implies the model generalizing to Q as well. In other words, equation 8 holds.

□

B Details about the compositional functions

As explained in equation 7 in section 4, the composition function is implemented as a soft pixel-wise addition in most experiments. The use of the sigmoid function $\sigma(\cdot)$ in the composition

$$\boldsymbol{x} = \sigma(\tilde{\boldsymbol{x}}_1) \cdot \tilde{\boldsymbol{x}}_1 + \sigma(-\tilde{\boldsymbol{x}}_1) \cdot \tilde{\boldsymbol{x}}_2 \tag{25}$$

was necessary for training stability. With this formulation, sprites can also overlap somewhat transparently, which is not desired and leads to small reconstruction artifacts for some specific samples. Implementing the composition with a step function as

$$\boldsymbol{x} = \mathrm{step}(\tilde{\boldsymbol{x}}_1) \cdot \tilde{\boldsymbol{x}}_1 + \mathrm{step}(-\tilde{\boldsymbol{x}}_1) \cdot \tilde{\boldsymbol{x}}_2 \tag{26}$$

instead would be more faithful to the ground-truth data-generating process, but is hard to train with gradient descent.

Note that both formulations could easily be extended to more than one sprite by simply repeating the composition operation with any additional sprite.

In section 4, we also looked at a model that implements the composition through alpha compositing instead (see also Table 1, **#11**). Here, each component's intermediate representation is an RGBa image. The components are then overlaid on an opaque black background using the composition function

$$x_\alpha = x_{1,\alpha} + (1 - x_{1,\alpha}) \cdot x_{2,\alpha} \tag{27}$$

$$x_{\mathrm{RGB}} = x_{1,\alpha} \cdot x_{1,\mathrm{RGB}} + (1 - x_{1,\alpha}) \cdot \frac{x_{2,\alpha}}{x_\alpha} \cdot x_{2,\mathrm{RGB}}. \tag{28}$$

While this yields a compositional function, the sufficient support condition (Definition 3) is generally not fulfilled on the sprites data. The reason is that in fully transparent pixels ($\alpha = 0$), changing the RGB value is not reflected in the output. Conversely, if a pixel is black, changing its alpha value will not affect how it is blended over a black background. As a result, most columns in the Jacobian $\frac{\partial C}{\partial \boldsymbol{\varphi}_k}$ (see also equation 15) will be zero. Since the intermediate representations of each sprite will contain a lot of black or transparent pixels (the entire background), the rank of the Jacobian here will be low. In this case, the workaround from equation 17 does not help since the low rank is not a result of another component in the foreground but of the specific parameterization of each component itself.

As stated in the main paper, the fact that this parameterization still produces good results and generalizes well is an indicator that there might be another proof strategy or workaround that avoids this specific issue.