A. Proof of Theorem 1

We recall the setting in Section 5, which compares the norm of 2-layer ReLU networks used to represent a family of piecewise constant functions directly versus in a composed manner. The input space $\mathcal{X} \subseteq \mathbb{R}$ is one-dimensional and the valid output space $\mathcal{V} \subset \mathcal{R}^k$ is a set of discrete points in a k-dimensional space. We first show a result for $k=1$, then extend to higher k.

Suppose that the input and ambient output space are 1-dimensional ($\mathcal{X} \subset \mathbb{R}, \mathcal{Y} = \mathbb{R}$) and we use a model $f_{\text{base}} : \mathcal{X} \to \mathcal{Y}$ from the family of bounded-norm two-layer ReLU neural networks $\mathcal{H}$. The valid output space $\mathcal{V}$ is a discrete set of points in $\mathbb{R}$, and the denoiser $\Pi : \mathbb{R} \to \mathcal{V}$ maps from real values to the nearest point in the valid set (breaking ties arbitrarily). We show that under certain conditions on the target function $f^\star : \mathcal{X} \to \mathcal{V}$, we can use a small-norm base predictor composed with a denoiser ($\Pi \circ f_{\text{base}}$) to represent $f^\star$, while directly representing $f^\star$ (without Π, as in standard fine-tuning) requires a large norm.

Target function family. The target function $f^\star$ is defined on a union of disjoint intervals $\mathcal{X} = \uplus_{i=0}^{N-1} [x_i^l, x_i^u]$ where the subscript index orders the intervals such that $x_{i+1}^l - x_i^u = \delta > 0$. Thus, there is a gap δ between any two intervals. We assume that all interval lengths $x_i^u - x_i^l$ are at least Δ_x. Since $\mathcal{V}$ is a discrete set of points, $f^\star$ is a piecewise constant function that takes a value $y_0^*, \ldots, y_{N-1}^* \in \mathcal{V}$ in each of the N intervals. Each interval has a distinct value, such that $y_i^* \neq y_{i+1}^*$ for any $0 \leq i \leq N-2$. We will slightly abuse notation throughout by referring to the index i of an interval as "interval i".

Hypothesis class and norm. Following Savarese et al. (2019), we define the hypothesis class $\mathcal{H}$ such that $f_\theta \in \mathcal{H}$ are

$$f_\theta(x) = \sum_{l=1}^{h} w_l^{(2)} \left[\langle w_l^{(1)}, x \rangle + b_l^{(1)} \right]_+ + b_l^{(2)} \tag{6}$$

over $x \in \mathbb{R}^d$, where we will take $d = 1$ throughout. Here, $w_l^{(1)} \in \mathbb{R}^d$ are rows of $W^{(1)} \in \mathbb{R}^{h \times d}$ and $b_l^{(1)}, b_l^{(2)}, w_l^{(2)} \in \mathbb{R}$ are elements of $b^{(1)}, b^{(2)}, w^{(2)} \in \mathbb{R}^h$ respectively. The parameter space for this hypothesis class is

$$\theta \in \Theta = \{(h, W^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}) : h \in \mathbb{N}, W^{(1)} \in \mathbb{R}^{h \times d}, w^{(2)}, b^{(1)}, b^{(2)} \in \mathbb{R}^h\}, \tag{7}$$

where the number of hidden units h can be unbounded. Note that since our function family is a piecewise function with a finite number of segments, a 2-layer ReLU network with a finite number of hidden units can exactly implement the function. Each network is associated with the squared Euclidean norm of the weights

$$C(\theta) = \frac{1}{2}(\|w^{(2)}\|_2^2 + \|W^{(1)}\|_F^2).$$

The norm associated with $f \in \mathcal{H}$ is the minimum norm required to implement a given f:

$$\|f\| = \inf_{\hat{\theta} \in \Theta} C(\hat{\theta}) \text{ s.t. } f_{\hat{\theta}} = f. \tag{8}$$

Our one-dimensional result relies on the result of Savarese et al. (2019) (Theorem 3.1) that for 2-layer ReLU networks, the norm can be rewritten as

$$\|f\| = \frac{1}{2} \max\left(\int_{-\infty}^{\infty} |f''(x)|^2 dx, |f'(-\infty) + f'(+\infty)| \right). \tag{9}$$

As a corollary, for one-dimensional functions, the minimum norm interpolant in $\mathcal{H}$ has equivalent norm to the norm of a linear spline interpolation of the points (Savarese et al., 2019).

A.1. Lower bound on $\|f_{\text{std}}\|$

We use the norm equivalence between 2-layer ReLU networks and linear splines in one dimension to compare the norm of functions from $\mathcal{H}$ that represent $f^\star$ with and without Π.

Lemma 1. *For piecewise constant target function $f^\star$ defined by endpoints at the points $(x_0^l, y_0^*), (x_0^u, y_0^*), \ldots, (x_{N-1}^l, y_{N-1}^*), (x_{N-1}^u, y_{N-1}^*)$, any $f_{std} \in \mathcal{H}$ has norm $\|f_{std}\| \geq \sum_{i=0}^{N-2} \frac{|y_{i+1}^* - y_i^*|}{\delta}$.*

Proof. By Theorem 3.3 in Savarese et al. (2019), the norm of a linear spline interpolation lower bounds $\|f_{\text{std}}\|$ for any $f_{\text{std}} \in \mathcal{H}$. We define a linear spline interpolation f of the $2N$ points as

$$f(x) = \begin{cases} y_i^* + \alpha_{2i}(x - x_i^l) & x_i^l \le x \le x_i^u, 0 \le i \le N-1 \\ y_i^* + \alpha_{2i+1}(x - x_i^u) & x_i^u \le x \le x_{i+1}^l, 0 \le i \le N-2 \end{cases} \tag{10}$$

where $\alpha_{2i+1} = \frac{y_{i+1}^* - y_i^*}{\delta}$ for $0 \le i \le N-2$ and $\alpha_{2i} = 0$ for $0 \le i \le N-1$. From Savarese et al. (2019), we have that $\|f\| = \max(\sum_{j=0}^{2N-3} |\alpha_{j+1} - \alpha_j|, |\alpha_0 + \alpha_{N-2}|)$, which we lower bound with the first term. There are $N-1$ nonzero slopes α_j and each one contributes to the norm twice. Thus $\|f_{\text{std}}\| \ge \|f\| = \frac{1}{2}(2\sum_{i=0}^{N-2} \frac{|y_{i+1}^* - y_i^*|}{\delta})$. □

A.2. Upper bound on $\|f_{\text{base}}^*\|$

We compute an upper bound on the norm of a min-norm base predictor f_{base}^* by construction. Consider learning a function $f_{\text{base}} \in \mathcal{H}$ where we make predictions as Π composed with f_{base}. For every value $y_i^* \in \mathcal{V}$, let $(y_i^l, y_i^u]$ be the interval of values in $\mathbb{R}$ closest to y_i^*. Thus, if $y \in (y_i^l, y_i^u]$ then Π maps y to y_i^*. Without loss of generality, we have assumed that Π breaks ties such that y_i^l does not map to y_i^*.

Adjacent intervals. Define interval i to be *adjacent* to $i+1$ if it satisfies either $y_i^u = y_{i+1}^l$ or $y_i^l = y^u i+1$, or equivalently, there is no target value $y_j^* \in \mathcal{V}$ in the interval (y_i^*, y_{i+1}^*). Considering the i-th pair of intervals to be $(i, i+1)$, let I be the index set of adjacent pairs of intervals in $f^\star$ and J the index set of non-adjacent pairs, where $|I| + |J| = N-1$. Assume $\min_i |y_{i+1}^* - y_i^*| \ge L, \max_i |y_{i+1}^* - y_i^*| \le U$ are the min and max separations between valid points.

Lemma 2. *The norm of the minimum-norm base predictor $\|f_{base}^*\|$ is upper bounded as*

$$\|f_{base}^*\| \le \max\left(\frac{|J|U}{\delta} + \frac{|I|U}{\Delta_x}, \frac{U}{\Delta_x}\right). \tag{11}$$

Proof. We give an explicit construction $\hat{f}$ in the univariate setting where the norm of $\hat{f}$ upper bounds $\|f_{\text{base}}^*\|$. We define the construction $\hat{f}$ via a set of points $(x_0, y_0), \ldots, (x_{2N-1}, y_{2N-1})$ to linearly interpolate. For interval $1 \le i \le N-2$, we have two different cases describing the interval's relation with its previous and next intervals:

1. Same direction: if $y_{i-1}^* < y_i^* < y_{i+1}^*$, set $(x_{2i}, y_{2i}) = (x_{2i}^l, y_{2i}^l)$ and $(x_{2i+1}, y_{2i+1}) = (x_i^u, y_i^u)$. If $y_{i-1}^* > y_i^* > y_{i+1}^*$, set $(x_{2i}, y_{2i}) = (x_{2i}^l, y_{2i}^u)$ and $(x_{2i+1}, y_{2i+1}) = (x_i^u, y_i^l)$.
2. Change direction: if $y_{i-1}^* < y_i^* > y_{i+1}^*$, set $(x_{2i}, y_{2i}) = (x_{2i}^l, y_{2i}^l + \epsilon)$ and $(x_{2i+1}, y_{2i+1}) = (x_i^u, y_i^l + \epsilon)$. If $y_{i-1}^* > y_i^* < y_{i+1}^*$, set $(x_{2i}, y_{2i}) = (x_{2i}^l, y_{2i}^u)$ and $(x_{2i+1}, y_{2i+1}) = (x_i^u, y_i^u)$.

We will choose $\epsilon > 0$ to be a small, arbitrary value. For the beginning and end intervals $i \in \{0, N-1\}$, we choose $(x_0, y_0), (x_{2N-2}, y_{2N-2})$ to minimize the norm of the linear spline interpolation given the other points.

We change the construction for adjacent intervals as follows:

1. Adjacent to previous interval ($i > 0$): If interval $i-1$ is adjacent to i, we change the construction such that $x_{2i} = x_i^l - \delta/2$.
2. Adjacent to next interval ($i < N-1$): If interval i is adjacent to $i+1$, then $x_{2i+1} = x_i^u + \delta/2$ (unless case 3 occurs). If $0 < i < N-1$ and $y_{i-1}^* < y_i^* > y_{i+1}^*$, then we also set $y_{2i+1} = y_i^l$ (instead of $y_i^l + \epsilon$).
3. Adjacent to both previous and next intervals ($0 < i < N-1$): If $(i-1, i), (i, i+1)$ are adjacent and $y_{i-1}^* < y_i^* > y_{i+1}^*$, set $x_{2i+1} = (x_i^u - x_i^l)/2$ and $y_{2i+1} = y_i^l + \epsilon$.

The number of non-adjacent pairs of intervals in $f^\star$ determines the complexity gap between $\|f_{\text{base}}^*\|$ and $\|f_{\text{std}}\|$.

Let $\hat{f}$ be the linear spline interpolation of the points $(x_0, y_0), \ldots, (x_{2N-1}, y_{2N-1})$ as above, where $\Pi \circ \hat{f}(x) = f^\star(x)$ for $x \in \mathcal{X}$ by construction. As a feasible solution, $\|\hat{f}\| \ge \|f_{\text{base}}^*\|$. We distinguish between interval segments with endpoints $(x_{2i}, y_{2i}), (x_{2i+1}, y_{2i+1})$ and interconnecting segments with endpoints $(x_{2i+1}, y_{2i+1}), (x_{2(i+1)}, y_{2(i+1)})$, for $0 \le i \le N-2$. For any i, let $\hat{\alpha}_{2i}$ be the slope of the interval segment and $\hat{\alpha}_{2i+1}$ be the slope of the interconnecting segment. For some

interconnecting segments in the construction, the segment is of length zero. For these interconnecting segments, we define $\hat{\alpha}_{2i+1} = \hat{\alpha}_{2i}$ which does not affect the norm calculation. The norm of the construction is

$$\|\hat{f}\| = \frac{1}{2}\max(\sum_{i=0}^{N-2} |\hat{\alpha}_{2i+1} - \hat{\alpha}_{2i}| + |\hat{\alpha}_{2(i+1)} - \hat{\alpha}_{2i+1}|, |\hat{\alpha}_0 + \hat{\alpha}_{N-2}|).$$

Notice that both differences in the first term involve an interconnecting segment.

We first bound the first term in the norm. Suppose $(i,i+1) \in J$ is an non-adjacent pair. The contribution to the norm is

$$\begin{aligned} |\hat{\alpha}_{2i+1} - \hat{\alpha}_{2i}| + |\hat{\alpha}_{2(i+1)} - \hat{\alpha}_{2i+1}| &\leq 2|\hat{\alpha}_{2i+1}| \\ &\leq 2\frac{|y^*_{i+1} - y^*_i|}{\delta} \leq \frac{2U}{\delta} \end{aligned}$$

where in the first inequality, we note that the worst-case difference in slopes in our construction is when $\hat{\alpha}_{2i} = 0$ and $\hat{\alpha}_{2(i+1)} = 0$. The second step follows from $|\hat{\alpha}_{2i+1}| \leq \frac{\min(|y^u_i - y^l_{i+1}|, |y^l_i - y^u_{i+1}|) + \epsilon}{\delta}$ which is upper bounded by the second inequality for small enough ϵ. For purposes of the bound, we let $y^u_j = y^l_j + U$ for $j = \text{argmax}_i y^*_i$ and $y^l_k = y^u_k - U$ for $k = \text{argmin}_i y^*_i$. We can do this since the construction always ''changes direction" with slope going to 0 as $\epsilon \to 0$ for the extremal-valued intervals.

Suppose $(i,i+1) \in I$ is an adjacent pair. Let A be the event that $0 < i < N-1$ and $y^*_{i-1} < y^*_i > y^*_{i+1}$. If not A, the contribution to the norm is $|\hat{\alpha}_{2(i+1)} - \hat{\alpha}_{2i}|$ since the interconnecting segment has length zero and $\hat{\alpha}_{2i+1} = \hat{\alpha}_{2i}$. In this case, the contribution to the norm is

$$\begin{aligned} |\hat{\alpha}_{2(i+1)} - \hat{\alpha}_{2i}| &\leq |\hat{\alpha}_{2(i+1)}| + |\hat{\alpha}_{2i}| \\ &\leq \frac{|y^u_{i+1} - y^l_{i+1}|}{|x^u - x^l|} + \frac{|y^u_i - y^l_i|}{|x^u - x^l|} \leq 2\frac{U}{\Delta_x}. \end{aligned}$$

If A, we have $|\hat{\alpha}_{2i+1} - \hat{\alpha}_{2i}| \leq 2\epsilon/(\Delta_x/2)$ from the special case. Thus the contribution to the norm is

$$\begin{aligned} |\hat{\alpha}_{2i+1} - \hat{\alpha}_{2i}| + |\hat{\alpha}_{2(i+1)} - \hat{\alpha}_{2i+1}| &\leq \frac{4\epsilon}{\Delta_x} + |\hat{\alpha}_{2(i+1)}| + |\hat{\alpha}_{2i+1}| \\ &\leq \frac{4\epsilon}{\Delta_x} + \frac{|y^u_{i+1} - y^l_{i+1}|}{|x^u - x^l|} + \frac{\epsilon}{|x^u - x^l|/2} \\ &\leq \frac{U + 6\epsilon}{\Delta_x} \leq 2\frac{U}{\Delta_x} \end{aligned}$$

for small enough ϵ.

For the second term in the norm, we bound

$$|\hat{\alpha}_0 + \hat{\alpha}_{N-2}| \leq |\hat{\alpha}_0| + |\hat{\alpha}_{N-2}| \leq \frac{2U}{\Delta_x} \tag{12}$$

for small enough ϵ. Putting cases together and using $\|f^*_{\text{base}}\| \leq \|\hat{f}\|$ gives the result. □

A.3. Univariate result

Lemma 3. *Let $f^\star$ be a piecewise constant function defined on $\mathcal{X} = \uplus_{i=1}^n [x^l_i, x^u_i]$ with values in a discrete set $\mathcal{V} \subset \mathbb{R}$, and let $\delta > 0$ be the gap between any two intervals in $\mathcal{X}$. Let f_{std} be a 2-layer bounded-norm ReLU network that implements $f^\star$ and $f^*_{base} = \min_{f \in \mathcal{H}} \{\|f\| : \Pi \circ f(x) = f^\star(x), x \in \mathcal{X}\}$. Then*

$$\frac{\|f_{std}\|}{\|f^*_{base}\|} = \Omega\left(\frac{NL}{U(|J| + \delta\frac{|I|}{\Delta_x})}\right). \tag{13}$$

Proof. Using Lemma (1), we have $\|f_{\text{std}}\| \geq \frac{NL}{\delta}$. Taking the (inverse) ratio with the upper bound in Lemma (2) (considering the second term of the maximum as a constant) implies that the (inverse) ratio between norms is

$$\frac{\|f^*_{\text{base}}\|}{\|f_{\text{std}}\|} < \frac{U}{NL}\left(|J| + \delta\frac{|I|}{\Delta_x}\right).$$

Taking the inverse gives the result. □

A.4. Multivariate outputs

We extend Lemma 3 to the multivariate output case, considering functions from $\mathbb{R} \to \mathbb{R}^k$. Here, the denoiser is defined as $\Pi(y) \in \operatorname{argmin}_{v \in \mathcal{V}} \|v - y\|_2$. Similar to the univariate case, we will construct a lower bound on the complexity of f_{std} and upper bound on f^*_{base}. For the lower bound, extending to the multivariate case is straightforward: to fit the multivariate output, the model must also fit each individual output coordinate, and thus the maximum over the individual multivariate lower bounds will hold. For the upper bound, we consider a 2-layer ReLU neural network family where the j-th output of $f_\theta(x)$ is $f_{\theta_j}(x)$, where each f_{θ_j} is a univariate 2-layer ReLU network, and measure its complexity. In our special case, we show the coordinate-wise and multivariate projections are the same, allowing us to leverage the univariate bound.

For each univariate network, the first layer weights are shared but each output j has a different second layer weight. We denote the second layer weights for the j-th network as $w^{(2,j)} \in \mathbb{R}^h$. The norm of the multivariate output model f_θ is defined via

$$C(\theta) = \frac{1}{2}(\|W^{(1)}\|_F^2 + \sum_{j=1}^{k} \|w^{(2,j)}\|_2^2). \tag{14}$$

Again, the norm associated with $f \in \mathcal{H}$ is the minimum norm required to implement a given f:

$$\|f\| = \inf_{\hat{\theta} \in \Theta} C(\hat{\theta}) \text{ s.t. } f_{\hat{\theta}} = f. \tag{15}$$

Here, we define bounds for terms analogous to those in the univariate case. We let J_j and I_j be the index set of non-adjacent and adjacent pairs of intervals respectively in the j-th output coordinate. Let $y^*_{i,j}$ be the j-th output coordinate of the i-th valid point. For the j-th output coordinate, let $L_j = \min_i |y^*_{i,j} - y^*_{i+1,j}|$ and $U_j = \max_i |y^*_{i,j} - y^*_{i+1,j}|$ be the min and max separation between valid points. Let Δ_x be the length of the smallest interval in $\mathcal{X}$. Given these definitions, we show a similar gap in the multivariate output case.

Theorem 1. *Let the valid output space $\mathcal{V}$ be a set of N discrete points $y^*_1, \ldots, y^*_N$ in $\mathcal{V} = \mathbb{R}^k$. Let $f^\star$ be a piecewise constant function defined on $\mathcal{X} = \uplus_{i=1}^n [x^l_i, x^u_i]$ with values in the discrete set $\mathcal{V}$, and let $\delta > 0$ be the gap between any two intervals in $\mathcal{X}$. Let $f_{std} : \mathbb{R} \to \mathbb{R}^k$ be a multivariate bounded-norm 2-layer ReLU network in $\mathcal{H}$ that implements $f^\star$ and $f^*_{base} = \min_{f \in \mathcal{H}} \{\|f\| : \Pi \circ f(x) = f^\star(x), x \in \mathcal{X}\}$. Then*

$$\frac{\|f_{std}\|}{\|f^*_{base}\|} = \Omega\left(\frac{N \max_j L_j}{\sum_{j=1}^k U_j \left(|J_j| + \delta \frac{|I_j|}{\Delta_x}\right)}\right). \tag{16}$$

Proof. For the lower bound on $\|f_{\text{std}}\|$, note that f_{std} must fit every coordinate of the output (a univariate regression problem). Thus, we can lower bound by the norms on any output coordinate, $\|f_{\text{std}}\| \geq \|f_{\text{std},j}\|$ for any j, where $f_{\text{std},j}$ is the univariate 2-layer ReLU network for the j-th output coordinate. In particular, we can bound by the maximum of the norms, $\|f_{\text{std}}\| \geq \max_j \|f_{\text{std},j}\|$.

For the upper bound on $\|f^*_{\text{base}}\|$, we construct a multivariate network f_θ. For the j-th output coordinate f_{θ_j} of the construction, we ensure that $\Pi_j \circ f_{\theta_j} = f^\star_j$, where the projection for the j-th coordinate is $\Pi_j(y) = (\arg\min_{v \in \mathcal{V}} |v_j - y|)_j$. While generally the coordinate-wise and multivariate projections are not equal, they are equal here since the coordinate-wise projection is in $\mathcal{V}$: by definition, $f^\star_j(x) = (\arg\min_{v \in \mathcal{V}} |v_j - f_{\theta_j}(x)|)_j$ for all j, and thus for any valid $v \in \mathcal{V}$, $\sum_j (f^*_j(x) - f_{\theta_j}(x))^2 \leq \sum_j (v_j - f_{\theta_j}(x))^2$ so that the coordinate-wise projection and the multivariate projections are the same. We assume for convenience that ties in the multivariate projection and the coordinate-wise projection are broken in the same way in these cases. Thus, for the upper bound we focus on constructing f_θ where the coordinate-wise projection represents $f^\star$.

We take f_{α_j} to be an independent univariate 2-layer ReLU network that fits the j-th coordinate of $f^\star$ and has the same norm the corresponding univariate construction from Lemma 3 that fits the j-th output coordinate. Each α_j consists of h_j hidden units with first layer weights $W^{(1)}_{\alpha_j}, b^{(1)}_{\alpha_j}, w^{(2)}_{\alpha_j}, b^{(2)}_{\alpha_j} \in \mathbb{R}^{h_j}$. We construct f_{θ_j}, the network that computes the j-th output of our construction f_θ, by concatenating the first layer weights of each f_{α_j} to define the shared first layer weights

$$W^{(1)} = [W^{(1)}_{\alpha_1}; \ldots; W^{(1)}_{\alpha_k}] \in \mathbb{R}^h$$
$$b^{(1)} = [b^{(1)}_{\alpha_1}; \ldots; b^{(1)}_{\alpha_k}] \in \mathbb{R}^h$$

	Test MSE
Baseline	0.193
Composed (Emboss)	**0.187**
Composed (Contrast)	**0.172**
Composed (Gaussian blur)	**0.171**

Figure 5: Test MSE on font image generation for a variety of noising functions.

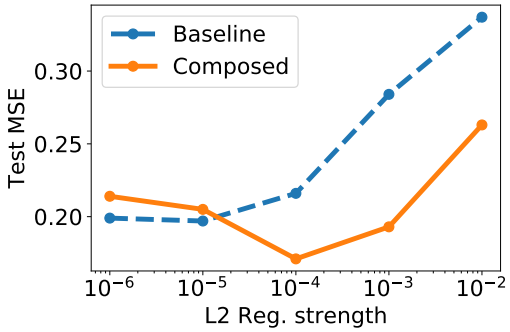


Figure 6: Varying L2 regularization strength (1e-6 to 1e-2) for direct and composed predictors. Increasing the regularization strength improves the composed predictor while hurting the direct predictor.

where $h = \sum_{j=1}^{k} h_j$. The second layer weights extend the corresponding second layer weights $w_{\alpha_j}^{(2)}$ with zeros for the newly introduced hidden units:

$$w^{(2,j)} = [\mathbf{0}^{h_{j-}}; w_{\alpha_j}^{(2)}; \mathbf{0}^{h_{j+}}]$$
$$b^{(2,j)} = [\mathbf{0}^{h_{j-}}; b_{\alpha_j}^{(2)}; \mathbf{0}^{h_{j+}}]$$

where $h_{j-} = \sum_{r=1}^{j-1} h_r$ and $h_{j+} = \sum_{r=j+1}^{k} h_r$. We define the j-th output of our construction $f_\theta(x)$ to be $f_{\theta_j}(x)$. The norm of this concatenated network is $\|f_\theta\| = \sum_{j=1}^{k} \|f_{\theta_j}\|$. We bound $\|f^*_{\text{base}}\| \leq \|f_\theta\|$.

Then using Lemma 1 and Lemma 2 on each output coordinate, we have $\|f_{\text{std}}\| \geq \frac{NL_j}{\delta}$ and $\|f^*_{\text{base}}\| \leq \|f_\theta\| \leq \sum_{j=1}^{k} (\frac{|J_j|U_j}{\delta} + \frac{|I_j|U_j}{\Delta_x})$. Taking the ratio gives the result. □

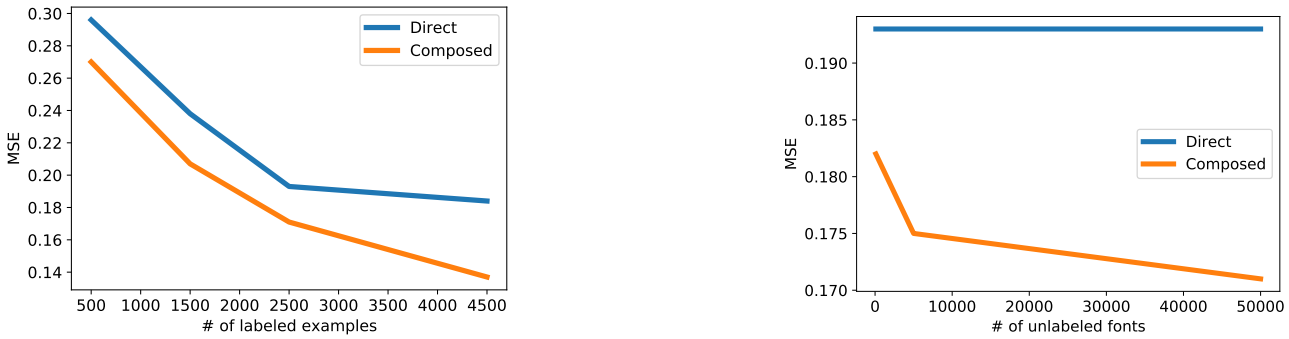
B. Font image generation

We use a dataset of 56k fonts scraped from the internet (Bernhardsson, 2016). Out of the 6200 labeled examples (62 characters $\times$ 100 fonts), we split randomly into 2500 training examples, 100 validation examples, and 3600 test examples. The training set contains 25 examples on average from each font, and the model must generate new characters from those fonts at test time. We have additional unlabeled images for ∼50k other fonts to train the denoiser. After learning the denoiser, we train the base model composed with the denoiser and minimize squared error. We set $\lambda = 0$ in (1), using only the composed component of the objective. We tune the L2 regularization strength out of {0, 0.1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6} according to squared error on the validation set. The denoiser is trained for two epochs on unlabeled data and all other models (baseline and composed) are trained for 1000 epochs on the labeled data, using the validation set for early stopping.

The baseline and base predictors use 7-layer feedforward networks where the one-hot vector for character identity is first fed through an embedding layer with embedding dimension 62, and the one-hot vector for the font uses an embedding layer of embedding dimension 100. These embeddings are then concatenated and fed through a 7-layer feedforward network.

Our models were trained on 1 Titan Xp or Titan RTX GPU. All image generation experiments took around 3 hours to run for each of the pre-training and fine-tuning steps.

Robustness to choice of denoising perturbations. In Table 5, we test the robustness of composing with a denoiser on different choices of noising functions. In addition to Gaussian blur, we change the contrast of the image by a factor of 0.5



(a) Vary labeled data

(b) Vary unlabeled data

Figure 7: Performance with varying labeled and unlabeled data (while fixing the other to the default in the original experiment) in font generation. The gap between the direct and composed models increases with increasing labeled and unlabeled examples.

or emboss the image. All noising functions improve beyond the baseline predictor.

Performance with varying labeled and unlabeled data size. We trained the denoiser with 100 and 5000 additional unlabeled fonts (originally 50k unlabeled fonts). The Composed test MSE was 0.182 and 0.175 for 100 and 5000 unlabeled fonts respectively, well below the Baseline test MSE (0.193) and approaching the result with 50k unlabeled fonts (0.171). Varying labeled data size in font generation (500, 1500, 2500, 4500 examples), the Composed model has lower test MSE than Baseline on all sample sizes, with an average relative MSE decrease of 11%. We visualize these results in Figure 7.

C. SANSTYPE pseudocode-to-code dataset

Each program in the dataset is generated in two phases. Each line in the program is templated and have pseudocode and noisy code templates to accompany it. When there are multiple options for any of code, pseudocode, or noisy code, a uniformly random option is sampled.

In the first phase, 1 to 4 variables with random types (`bool, string, int`) are instantiated. They may be assigned a random value or take on a value from `stdin`. Integers take random values in 0 to 100 and there are 10 possible strings in the form `str_i` for $i \in \{0,...,9\}$. There are 10 possible variable names in the form `var_i` for $i \in \{0,...,9\}$.

In the second phase, up to 5 additional code lines are generated. Twenty percent of the time, a new variable with a random type is instantiated and assigned a value. The other 80% consists of type-preserving operations on an existing variable or variables, such as setting a value, prepending/concatenating, adding/subtracting an integer, taking the logical AND, and conditional swap based on the value of a boolean variable.

The dataset presents the following challenges for the predictor, both stemming from ambiguity in the pseudocode:

1. Type inference: the words 'set' and 'add' are used almost everywhere for instantiation, initialization, and type-preserving operations.
2. Initialization: A particular example of ambiguity is that the pseudocode for value updates and initialization are often the same (`set <var> to <value>`), thus requiring the predictor to look in the pseudocode context to figure out whether to insert a type and what the correct type to insert should be.
3. Scopes and variable shadowing: Initialization can be particularly tricky during a conditional swap, where a scope is introduced with the if block. Inside the if block, one may reinitialize a variable that already exists (shadowing) or initialize a new variable which does not already exist. If the variable is not shadowed, it may change the value of the variable in the outer scope, causing a change in the program output. A locally-scoped variable cannot be used outside the if-block, so they may have to be reinitialized. In this dataset, variables are always newly initialized in the if-block.

Perturbations for denoising. We generate perturbations for denoising by inducing random semantic and syntactic changes:

1. Replacing, deleting, or inserting a random type in instantiation, initialization, and variable modification statements.

2. Print and input statements: removing double arrows («), reversing arrow directions, removing `cout`.

Experimental details. We use the sentencepiece BPE tokenizer (Kudo & Richardson, 2018; Sennrich et al., 2016a) with a joined dictionary size of 600. There are 1000 labeled pairs of pseudocode to code and 20000 unlabeled code examples. Separate validation and test sets of 500 examples are independently generated. All predictors use a Transformer architecture with encoder/decoder using 3 layers and 2 attention heads, embedding dimension 256, and FFN embedding dimension 1024. The denoiser uses a Transformer architecture with encoder/decoder using 5 layers and 4 attention heads, embedding dimension 256, and FFN embedding dimension 1024. We use dropout probability 0.1, attention dropout probability 0.2, ReLU dropout probability 0.2, weight decay 0.0001. We use the cross entropy loss. We train the baseline and backtranslation models (models trained frome scratch) for 500 epochs and pre-train the denoiser for 50 epochs. We do standard fine-tuning and composed fine-tuning for 300 epochs. For all predictors, we use the validation set to do early stopping. For composed fine-tuning, we use the cross entropy loss of the composition $\Pi \circ f_\theta$ on the validation set for early stopping.

D. Training details

Computing the gradient of the first term in the composed objective (Equation (1)) requires computing the partition function of $p_\beta(y \mid y')p_\theta(y' \mid x)$, marginalizing over outputs y'. Since the output space is large, this is intractable. Instead, we optimize a lower bound on the log-likelihood of the composed predictor. To see this, note that the log-likelihood of the composed predictor is lower bounded by the first term of the objective via Jensen's inequality:

$$\mathbb{E}_{x,y}[\log \mathbb{E}_{y' \sim p_\theta}[p_\beta(y \mid y')]] \geq \mathbb{E}_{x,y}[\mathbb{E}_{y' \sim p_\theta}[\log p_\beta(y \mid y')]]. \tag{17}$$

To optimize Equation (1) where the output space is discrete, we use the REINFORCE estimate of the gradient for the first term (Williams, 1992). We do not use the Gumbel-softmax reparameterization here since the Transformer model autoregressively conditions on its discrete outputs, but different choices of the base model can enable use of reparameterized gradients.

The composed predictor optimizes

$$\operatorname*{argmax}_\theta \mathbb{E}[\mathbb{E}_{\hat{y} \sim p_\theta}[\log p_\beta(y \mid \hat{y})]] + \lambda \mathbb{E}[\log p_\theta(y \mid x)] \tag{18}$$

where the outer expectations are over the data, where the samples are (x,y) pairs. Here, θ are the parameters of the learned model and β are the parameters of the denoiser. The score function estimate of the gradient is

$$\nabla_\theta \mathbb{E}[\mathbb{E}_{\hat{y} \sim p_\theta}[\log p_\beta(y \mid \hat{y}) \nabla_\theta \log p_\theta(\hat{y} \mid x)]] + \lambda \mathbb{E}[\nabla_\theta \log p_\theta(y \mid x)]. \tag{19}$$

For examples that the model predicts wrongly, the model is encouraged to put smaller probabilities on these examples. This may cause the first term of the gradient to have a very large magnitude, causing numerical instability. We resolve this by clamping $\log p_\beta(y \mid \hat{y})$ to $\max(\log p_\beta(y \mid \hat{y}), \gamma)$ where $\gamma = -50$ in all experiments. Even after clamping, the composed loss is roughly an order of magnitude larger than the standard loss; to normalize the losses, we scale the composed loss by 0.1. In all pseudocode-to-code experiments, we use $\lambda = 1$ to balance between the fitting the objective directly and using the composed objective.

Our models were trained on 1 Titan Xp or Titan RTX GPU. All SANSTYPE experiments took around 3 hours to run, which includes pre-training and fine-tuning steps.

BART-style fine-tuning. We implement a two-stage fine-tuning method similar to BART (Lewis et al., 2020) as follows. First, we fix the Transformer decoder and train only the encoder for 200 epochs. Then, we fine-tune the encoder-decoder together for 100 more epochs. We note that the BART paper adds extra layers to the encoder for fine-tuning, while we fine-tune the existing encoder. The first step has similarities to our method, and our theory may give some justification for this step as well.

E. SPOC

Denoising objective. We use 284477 unlabeled code examples from codeforce.com to generate 1137908 pairs of noisy code to valid code. For each unlabeled program, we generate 1 unnoised example and 3 noised examples, where each noised example has one line with an error. We follow (Yasunaga & Liang, 2020) to generate error lines by random semantic and syntactic changes, including insertion, deletion, and replacement of keywords, variables, and syntactical symbols.

Pre-trained model. We use random syntactic and semantic corruptions of additional ∼280k unlabeled code examples from `codeforces.com` as in Yasunaga & Liang (2020). Previous program repair works (Yasunaga & Liang, 2020) utilize compiler error messages to guide the repair model. We only use code as input, and thus the task is relatively difficult. We define p_Π in two parts. First, we train a binary classifier $g_\gamma : \mathcal{Y} \to \{0,1\}$ which detects if a program has an error (error is label 1), trained using the denoising dataset. For an output y', if $g_\gamma(y') = 0$ then we define $p_\Pi(y \mid y') = \delta(y')$ to be a delta distribution on y'. Otherwise, if $g_\gamma(y') = 1$, then $p_\Pi(y \mid y') = p_\nu(y \mid y')$, where p_ν is a Transformer. The Transformer p_ν is first pretrained using a linewise code repair dataset generated from unlabeled examples, then trained on full-program repair where the input program has one random corrupted line with probability 0.75. Thus, taking the parameters of Π to be (γ,ν), we have $\Pi(y') = y'$ if $g_\gamma(y') = 0$ and $\Pi(y') = \mathrm{argmax}_y p_\nu(y \mid y')$ otherwise.

Data processing. Since the SPoC dataset contains a small fraction of programs which have a large number of tokens, we filter out the longest examples from the training data. After filtering, we retain over 95% of the original training set. Similarly to SANSTYPE, special symbole (\$ and ∼) are added to delineate lines and tabs in the pseudocode and code, and we preprocess the code using a byte-pair encoding using SentencePiece (Kudo & Richardson, 2018), with joined vocabulary size 10000.

Data filtering. We train and test on a truncated version of the SPoC dataset (Kulal et al., 2019). We filter out an example during preprocessing if, after adding special tokens for tab and code lines, the number of characters exceeds 1000. This retains 11355 examples out of the full 11925 training examples. We use the given validation splits. When we filter the TESTP and TESTW test datasets, we retain 1441 out of 1820 examples in TESTP and 1659 out of 1752 examples in TESTW.

Training details. For SPoC experiments, we use a Transformer architecture for all models with 5 layers, 8 attention heads, embedding dimension 256, and FFN embedding dimension 1024. We use this architecture for both the denoiser and the models. We use dropout probability 0.4, attention dropout probability 0.2, ReLU dropout probability 0.2, weight decay 0.0001, taken as reasonable defaults from Guzmán et al. (2019). We use a decaying label smoothing schedule with smoothing parameter starting with 0.2 for 150 epochs, then 0.1 and 0.05 for 25 epochs each. We found that reducing the label smoothing parameter near the end of training improves generalization for all models. The composed predictor initializes from the pretrained predictor and are trained for 20 epochs, taking the best model according to (label-smoothed) cross entropy loss of the composition $\Pi \circ f_\theta$ on the validation set. For backtranslation models, we first train a code-to-pseudocode model using the labeled data and use this model to produce synthetic pseudocode examples for unlabeled code. Then, we train a pseudocode-to-code model using the labeled examples and synthetically generated examples. Finally, we use this model as initialization to finetune on the labeled data only.

Our models were trained on 1 Titan Xp or Titan RTX GPU. All SPoC experiments took around 10 hours for the pre-training and fine-tuning steps.