A Deriving a Gaussian approximation to the posterior

We repeat the derivation in §2.2 for a Gaussian, rather than a point estimate of the posterior.

Recall that if $p(\mathbf{x})$ is the likelihood (5) of $\mathbf{x}$ under a DDPM, then in the first expectation of (2) we should use $q(\mathbf{h} = \{\mathbf{x}_T, ..., \mathbf{x}_1\}|\mathbf{x}_0 = \mathbf{x}) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. A computationally and notationally convenient form for the approximate posterior over $\mathbf{x} = \mathbf{x}_0$ is a scalar-covariance Gaussian:

$$q(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \sqrt{1-\psi}\eta, \psi\mathbf{I}). \tag{16}$$

We can sample $\mathbf{x}_t$ at any arbitrary time step as

$$q(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\eta, (1-\bar{\alpha}_t)\mathbf{I}) \tag{17}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=0}^{t} \alpha_t$, with the convention that $\beta_0 = \psi$ (note the difference with (8), which is the special case of (17) with $\beta_0 = 0$). Analogously to [13], we can also extract a conditional Gaussian $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \psi, \eta)$ and express the first expectation in (2) as

$$-\mathbb{E}_{q(\mathbf{x})q(\mathbf{h}|\mathbf{x})}[\log p(\mathbf{x}, \mathbf{h}) - \log q(\mathbf{x})q(\mathbf{h}|\mathbf{x})] = \sum_t \mathrm{KL}(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \eta, \psi) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)), \tag{18}$$

which after reparametrization [13] leads to

$$\sum_t w_t(\beta)\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2], \quad \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\eta + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \tag{19}$$

where the link between the stage t noise reconstruction $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ and the model's expectation $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ is

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right). \tag{20}$$

Assuming uniform weighting of the noising steps as before, the free energy in (2) reduces to

$$F = \sum_t \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2] - \mathbb{E}_{q(\mathbf{x})}[\log c(\mathbf{x}, \mathbf{y})]. \tag{21}$$

Unlike (12), (21) involves an expectation over a Gaussian variable. To optimize through this expectation, one could use the reparametrization trick: $\mathbb{E}_{q(\mathbf{x})}[\log c(\mathbf{x}, \mathbf{y})] = \mathbb{E}_{\boldsymbol{\epsilon}_q\sim\mathcal{N}(0,\mathbf{I})}[\log c(\sqrt{1-\psi}\boldsymbol{\eta} + \boldsymbol{\epsilon}_q\sqrt{\psi}, \mathbf{y})]$.

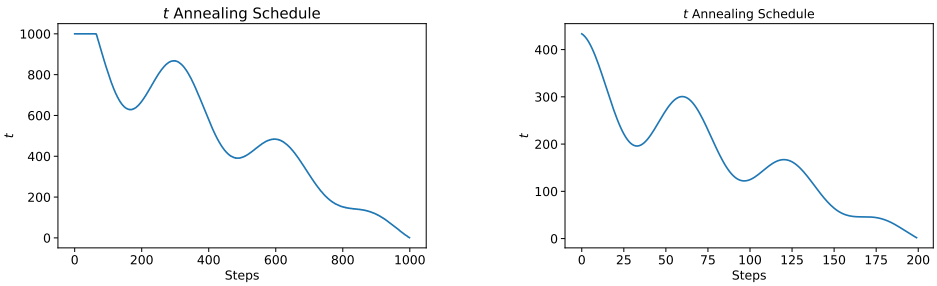
B Experiment details and extensions

B.1 MNIST

Training the DDPM. To train the diffusion model we used the U-Net architecture of [7] with a linear β_t schedule and $T = 1000$ diffusion steps. We trained the network for 10 epochs, with a batch size of 128 samples, using the Adam optimizer and a learning rate of 10^{-4}.

Performing inference. For all inference examples, we performed 1000 optimization steps with the Adam optimizer and a learning rate 10^{-2}. We employed a cosine-modulated, linearly decreasing t annealing schedule as shown in Fig. B.1 (a). We empirically designed this annealing process following the observation that the linearly decreasing t values guide the inference procedure in a coarse-to-fine manner that starts by deciding the overall structure of the inferred sample and then adding in details. We also added the oscillating component to allow for revisions of the coarser structures that are to be made after having inferred specific details.

When performing the optimization step of Algorithm 1, we observed that it was important to gradually reduce the effect of the condition $c(\mathbf{x}, \mathbf{y})$ in order to obtain good sample quality. In practice, we linearly decreased the weight of the conditional component of the loss, from $\lambda_T = 10^{-2}$ to $\lambda_1 = 0$ as we performed the optimization steps from $T, \ldots, 1$. This can be attributed to the fact that the conditions we used provide guidance for the steps made at larger values of t, where the shape and orientation of a digit are decided. When combining two or more conditions the weighting is applied to all of them.



(a) (b)

Figure B.1: Inference t annealing schedules for the (a) MNIST (b) Land Cover experiments. We do not necessarily need to optimize for all $T = 1000$ values to generate samples, as shown in (b). The TSP and FFHQ experiments use similarly defined schedules.

B.2 FFHQ

Performing inference. For the conditional generation experiments on the FFHQ dataset we utilized the pretrained DDPM model provided by [3]. The face attribute classifier network was a ResNet-18 network trained on the face attributes given in the CelebA dataset [25]. To run our inference algorithm we performed 200 optimization steps with the Adamax optimizer, choosing $(\beta_1, \beta_2) = (0.9, 0.999)$ and a linearly decreasing learning rate from 1 to 0.5. The t annealing schedule was similar to the one used for the land cover segmentation experiments (Fig. B.1 (b)) but for t values now ranging from 1000 to 200. Additionally, in this experiment we found that balancing between the diffusion and auxiliary losses with a carefully chosen weighting term was difficult. Thus, we opted for a different approach where we clipped the gradient norm of the auxiliary objective to $\frac{1}{2}$ of the gradient norm of the diffusion denoising loss.

Further discussion on samples. In Fig. B.2 we demonstrate additional conditionally-generated samples from the unconditional DDPM and the attribute classifier. In the first set of examples we show that although we may find modes of $p_{DDPM}(x)$ that satisfy to a level the condition $\mathbf{y}$ set by the classifier, the sample quality is not always on-par with unconditionally generated samples, like those presented in [3]. We can attribute that to the fact that for natural images, in contrast to segmentation labels, the mode may not always be a good-looking sample from the distribution. Our method to mitigate that, along with the classifier noise artifacts left from the optimization process, is to run the diffusion denoising procedure starting from a low temperature t. Although this may improve the visual quality of the result, in some cases our choice of t is not large enough to move the sample far enough from the inferred $\mathbf{x}$. If we choose a larger t however we risk erasing the attributes we aimed to generate in the first place.

In the second set of samples, we first show how conflicting attributes are resolved. When the constraint is set to satisfy two attributes that contradict each other we observed that the inferred sample $\mathbf{x}$ tends to gravitate towards a single randomly-chosen direction. This is evident in the first two examples where we set the *not male* attribute along with a male-correlated attribute. In each of them only a single condition, either the not male or the male-related, is satisfied. In the *blonde+black hair* example we could argue that a mix of the two attributes is present in the inferred sample. However, the classifier predictions for that specific image tell us that the person shown is exclusively blonde.

We also show a set of failure cases where the classifier 'painted' the features related to the desired attribute but the diffusion prior did not complete the sample in a correct way. For instance, in the *eyeglasses* example we see that the classifier has drawn an outline of the eyeglass edges on the generated face but the diffusion model has failed to pick up the cue. Similarly, when asking for *wavy hair* we see curves that can fool the classifier into thinking that the person has curly hair, or when the attributes set are *smiling+mustache* we observe a comically drawn mustache on the generated face. Since the conditioning depends both on the diffusion prior and the robustness of the classifier we believe that with better classifier training we could improve the result in such cases.

(a)

(b)

Figure B.2: (a) Additional conditional samples $\mathbf{x}$ for constraints $c(\mathbf{x}, \mathbf{y})$ with various attribute sets $\mathbf{y}$. (b) Failure cases of conditional generation with their attribute sets $\mathbf{y}$. For both sets of images we show the inference results (top) and the image after denoising as in [31] to remove artifacts that appear due to optimizing the classifier constraint (bottom).

B.3 Land cover

Training the DDPM. The land cover DDPM was trained on $\frac{1}{4}$-resolution, 64×64 patches of land cover labels, randomly sampled from the Pittsburgh, PA tiles of the EnviroAtlas dataset [32]. For the diffusion network, we used the U-Net architecture of [7], a linear β_t schedule and $T = 1000$ diffusion steps. We trained with 10^5 batches of size 32, using the Adam optimizer and a learning rate of 10^{-4}. Additional samples from the unconditional diffusion model are shown in Fig. B.3. We observe that the model has learned both structures that are independent of the geography, such as the continuity of roads and the suburban building planning, and PA-specific ones, such as buildings nested in forested areas, which may not be as common in AZ for instance.

Performing inference. Since we initialize the inference procedure with the weak labels we require fewer optimization steps and do not have to start the search from $t = 1000$. Thus, to infer the land cover segmentations we only perform 200 optimization steps using the Adam optimizer, with a linearly decreasing learning rate from 5×10^{-3} to 5×10^{-6} and $(\beta_1, \beta_2) = (0, 0.999)$. The annealing schedule we designed for this task reflects the needs for fewer overall steps and is shown in B.1 (b). We also decrease the weights of both conditional components of the loss, from $\lambda_T = 1$ to $\lambda_1 = 0$ as we perform the optimization steps $T, \ldots, 1$ to reduce their influence on the final inferred sample. In addition, we linearly decrease the σ parameter that is used to convert the one-hot representations learned from the DDPM model to probabilities, from $\sigma_T = 0.2$ to $\sigma_1 = 0.02$ to mimic the uncertainty of this conversion process. Further examples of land cover segmentation inference are shown in Fig. B.4. Despite the fact that the DDPM was trained only on PA land cover labels we show how the weak label guidance allows us to perform inference in completely new geographies, such as

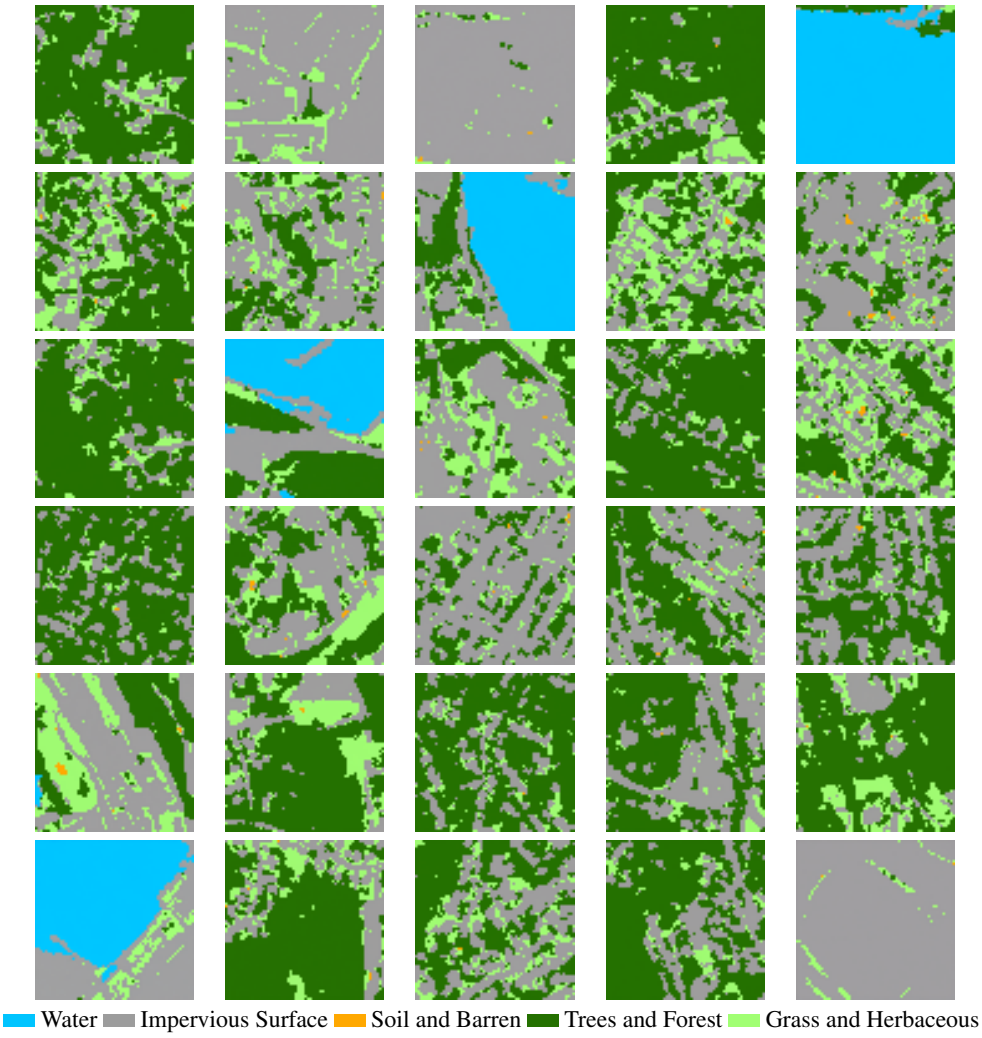


Figure B.3: Unconditional samples from the DDPM trained on land cover segmentations.

that of AZ (last two rows), where the most prominent label is now Soil and Barren. We can still observe a few artifacts of the PA-related biases the model has learned, like the tendency to add uninterrupted forested areas but the transferability of the semantic model is still far superior than an of an image-based one.

Our hand-crafted constraint for land cover segmentation inference is split between two objectives; (i) matching the structure of the target image using a local color clustering $\mathbf{z}$ and (ii) forcing the predicted segments' distribution to match the weak label distribution ℓ_{weak} when averaged in non-overlapping blocks of the image.

The local color clustering $\mathbf{z}$ is computed as a local Gaussian mixture with a fixed number of components. To match the structure between the predicted labels and the precomputed clustering we compute the mutual information between the two distributions in overlapping patches of 31×31 pixels. This choice of constraint pushes the inferred land covers segments in a way that they should match locally the color clustering segments. Although this allows us to infer the labels of large structures like roads and buildings it also tends to add noisy labels at areas where the clustering has a high entropy. By gradually reducing the weight of the auxiliary objective however, we allow the inference procedure to 'fill in' these details as it is dictated by the diffusion prior.

The label guidance during inference is provided from probabilistic weak labels ℓ_{weak} which are derived from coarse auxiliary data. These data are composed of the 30m-resolution National Land Cover Database (NLCD) labels, augmented with building footprints, road networks and water-

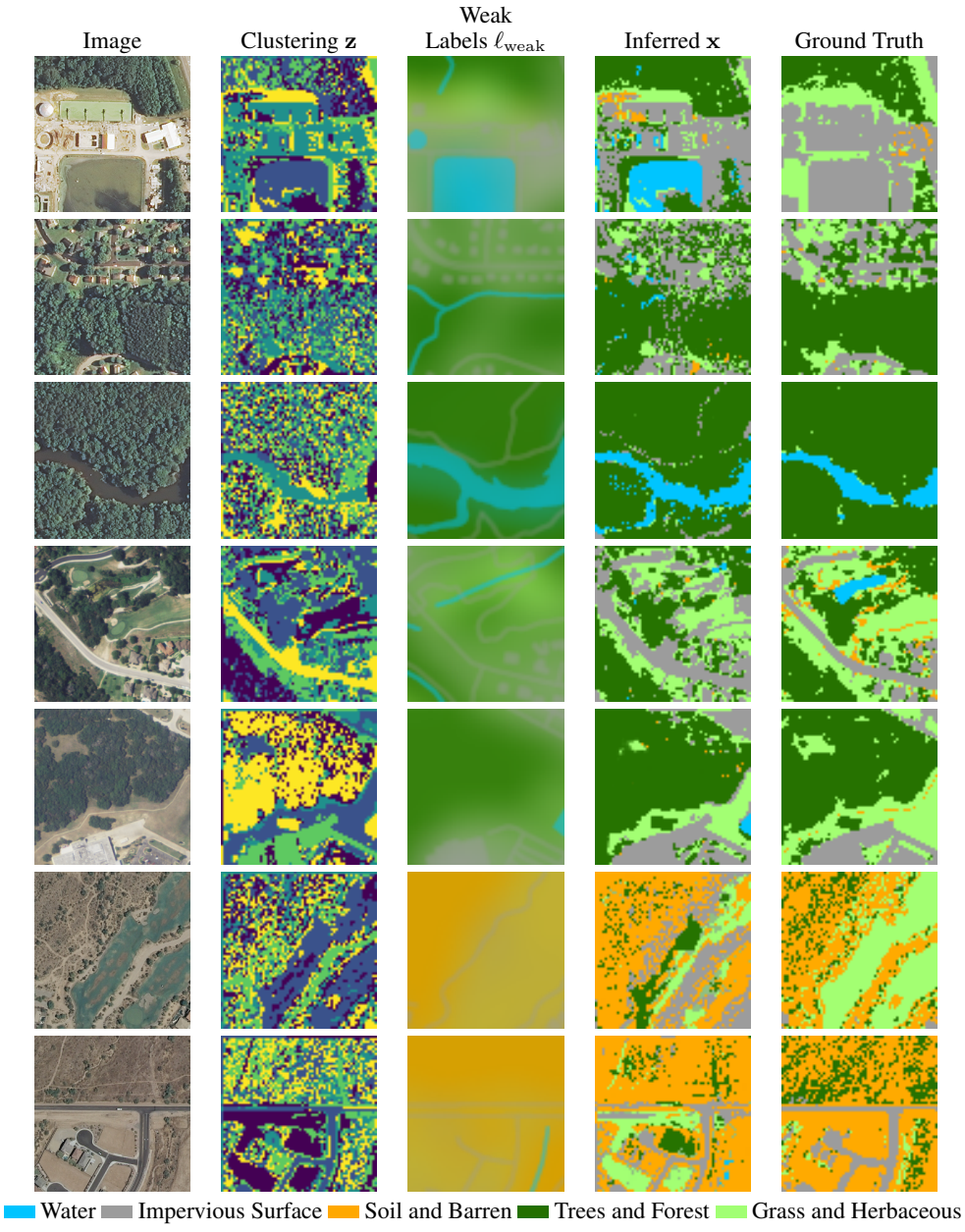


Figure B.4: Segmentation inference results.

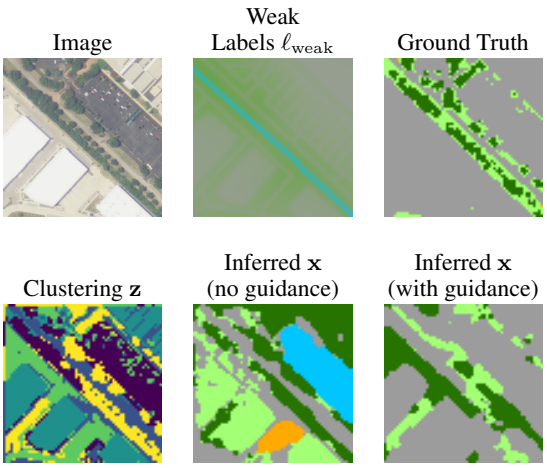


Figure B.5: Inference with and without weak label guidance.

ways/waterbodies [34]. The corresponding weak label constraint is computed as the KL-divergence between the average predicted and weak label distributions in non-overlapping blocks of 31×31 pixels. In the absence of such guidance the inference procedure can easily confuse semantic classes while still producing segmentations that are likely under $p_{DDPM}(x)$. We showcase this in Fig. B.5 where we infer the land cover labels of an image, starting from a random initialization, with and without the weak label guidance.

Domain transfer. Regarding the domain transfer experiments, we initially pretrained the standard inference U-Net [35] on 2×10^4 batches of 16 randomly sampled 64×64 image patches in Pittsburgh, PA, using the Adam optimizer with a learning rate of 10^{-4}. We then inferred the land cover segmentations of 640 randomly-sampled patches in each of the other geographic regions, (NC, TX, AZ) using the inference procedure described above. With these generated labels, we first finetuned the original network on a validation set of 5 tiles to determine the optimal finetuning parameters. For Durham, NC and Austin, TX we only finetune the last layer of the network for a single epoch, using a batch size of 16 patches and a learning rate of 5×10^{-4}. For Phoenix, AZ we require 5 epochs of finetuning the entire network with a learning rate of 5×10^{-4} since the domain shift is larger. Additionally, for all regions, following the experiments of [34], we multiply the predicted probabilities with the weak labels and renormalize.

Finally, in Table 1, we also present the results when the inference network is trained from scratch, to show that the resulting performance is not only an artifact of the pretraining. The U-Net was trained for 20 epochs on all 640 generated samples, with a batch size of 16 and a learning rate of 10^{-3}.

B.4 TSP

DDPM training. The DDPM was trained on 64×64 images of ground truth TSP solutions encoded as images. The architecture was the same U-Net as used in the other experiments, with the architecture from [7] and $T = 1000$ diffusion steps in training. We trained each model for 8 epochs with batch size 16, which took about two days on one Tesla K80 GPU.

Performing inference. At inference time, we performed varying numbers of inference steps (see Table 2 in the main text), using the Adam optimizer with $(\beta_1, \beta_2) = (0, 0.9)$ and a learning rate linearly decaying from 1 to 0.1. The noise schedule was the same as that used in the MNIST experiment (Fig. B.1), with the time interval from 0 to 1000 linearly resampled to the number of inference steps used.

To extract a tour from the inferred adjacency matrix A, we used the following greedy edge insertion procedure.

- Initialize extracted tour with an empty graph with N vertices.

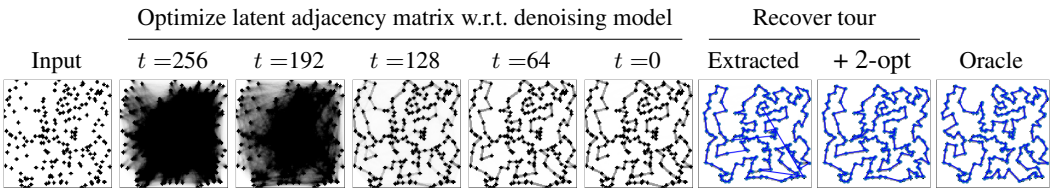


Figure B.6: Latent adjacency matrix inference in a 200-vertex TSP, using a model trained on 64×64 images but 128×128 images at inference time. The discovered tour is 2.12% longer than the optimal one.

- Sort all the possible edges (i, j) in decreasing order of $A_{ij}/\|v_i - v_j\|$ (i.e., the inverse edge weight, multiplied by inferred likelihood). Call the resulting edge list $(i_1, j_1), (i_2, j_2), \ldots$.
- For each edge (i, j) in the list:
  - If inserting (i, j) into the graph results in a complete tour, insert (i, j) and terminate.
  - If inserting (i, j) results in a graph with cycles (of length $< N$), continue.
  - Otherwise, insert (i, j) into the tour.

It is easy to see that this algorithm terminates before the entire edge list has been traversed. The tour is refined by a naïve implementation of 2-opt, in which, on each step, all pairs of edges in the tour $((i, j), (k, l))$ are enumerated and a 2-opt move is performed if the edges cross. For the '2-opt' baseline, the same procedure is performed using a uniform adjacency matrix.

Results on larger problems. Extending the results in Table 2 of the main text, we evaluate the model trained on TSP instances with 20 to 50 nodes on problems with 200 nodes. We find an optimality gap of 3.77% (average number of uncrossing moves 219), compared to 3.81% for 2-opt (average number of uncrossing moves 115), suggesting that the generalization potential is near-saturated at this problem size. As shown in Fig. B.7, the vertices fill the 64×64 image with such high density that it is difficult to see the (light grey) tour; many edges are invisible (compare to Fig. 7 in the main text).

We suggest three directions to solving to this problem that should be explored in later work:

(1) Encoding: The size of the encoding image can be increased (for example, to 128×128) when the number of vertices increases, without changing the model (trained on 64×64 images), which can make denoising predictions on images of any size. We may expect to see better out-of-domain generalization of the denoising model in this setting, as the density of nodes (mean number of black pixels) would match that in the training set. Figs. B.8 and B.6 show the potential of DDPMs to generalize to image sizes larger than those in which they were trained. Inference using 128×128 images gives an optimality gap of 2.59% (average number of uncrossings 81), much lower than that obtained with in-domain image size.

In addition, encoding graphs with smaller dots and thinner lines can be explored, although the generalization difficulties due to image 'crowding' would still appear at a larger value of N.

(2) Fractal behaviour and coarse-to-fine: Taking advantage of the fractal structure of Euclidean TSP solutions, a denoising objective could be used to *locally* refine the tour by minimizing the objective on a *crop* of the image representation (a form of DDPM-guided local search). This could be done in a coarse-to-fine manner by application of the same model at different scales, with a 128×128 representation of a problem with 200 vertices being first optimized with respect to the denoising objective globally, then on 64×64 crops.

(3) Improved extraction: The 2-opt search can be improved by inexpensive heuristics, such as choosing the 2-opt move that most improves the cost on every step, rather than iterating through the edges of the candidate tour in order.

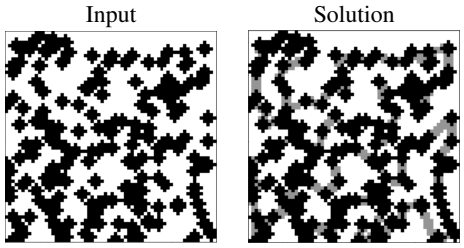


Figure B.7: A TSP instance and the ground truth solution with $N = 200$ vertices encoded in a 64×64 image.

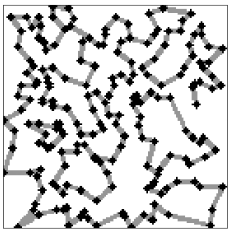


Figure B.8: Unconditional 128×128 samples from the DDPM trained on 64×64 image representations of 50-vertex TSPs.