A Proofs

Proof of Theorem 3.1. By the definition of $\mathbb{P}_1$, it holds that $\pi^* \in \Pi(\mathbb{P}_0, \mathbb{P}_1)$. It suffices to show that π^* attains the optimal cost. Let **Cost**(π) be the value of weak OT functional for a plan π, i.e.,

$$\textbf{Cost}(\pi) \overset{\text{def}}{=} \int_{\mathcal{X}} C(x, \pi(\cdot|x)) d\mathbb{P}(x).$$

We consider weak OT (2) between $\mathbb{P}_0 \in \mathcal{P}(\mathcal{X})$ and $\mathbb{P}_1 \in \mathcal{P}_p(\mathcal{X})$ and use its dual form (7):

$$\textbf{Cost}(\mathbb{P}_0, \mathbb{P}_1) = \sup_f \Big\{ \int_{\mathcal{X}} f^C(x) d\mathbb{P}_0(x) + \int_{\mathcal{Y}} f(y) d\mathbb{P}_1(y) \Big\} =$$

$$\sup_f \Big\{ \int_{\mathcal{X}} \inf_{\nu \in \mathcal{P}_p(\mathcal{Y})} \{C(x,\nu) - \int_{\mathcal{Y}} f(y) d\nu(y)\} d\mathbb{P}_0(x) + \int_{\mathcal{Y}} f(y) d\mathbb{P}_1(y) \Big\} \geq$$

$$\int_{\mathcal{X}} \inf_{\nu \in \mathcal{P}_p(\mathcal{Y})} \{C(x,\nu) - \int_{\mathcal{Y}} f^*(y) d\nu(y)\} d\mathbb{P}_0(x) + \int_{\mathcal{Y}} f^*(y) d\mathbb{P}_1(y).$$

Now we use the fact that $\pi^*(\cdot|x)$ minimizes (8) for all $x \in \mathcal{X}$:

$$\int_{\mathcal{X}} \inf_{\nu \in \mathcal{P}_p(\mathcal{Y})} \{C(x,\nu) - \int f^*(y) d\nu(y)\} d\mathbb{P}_0(x) + \int_{\mathcal{Y}} f^*(y) d\mathbb{P}_1(y) =$$

$$= \int_{\mathcal{X}} \big\{ C(x, \pi^*(\cdot|x)) - \int_{\mathcal{Y}} f^*(y) d\pi^*(y|x) \big\} d\mathbb{P}_0(x) + \int_{\mathcal{Y}} f^*(y) d\mathbb{P}_1(y) =$$

$$\int_{\mathcal{X}} C(x, \pi^*(\cdot|x)) d\mathbb{P}_0(x) - \int_{\mathcal{X}} \int_{\mathcal{Y}} f^*(y) d\pi^*(y|x) \underbrace{d\mathbb{P}_0(x)}_{= d\pi_0^*(x)} + \int_{\mathcal{Y}} f^*(y) d\mathbb{P}_1(y) =$$

$$\int_{\mathcal{X}} C(x, \pi^*(\cdot|x)) d\mathbb{P}_0(x) - \int_{\mathcal{X} \times \mathcal{Y}} f^*(y) d\pi^*(x,y) + \int_{\mathcal{Y}} f^*(y) d\mathbb{P}_1(y) =$$

$$\int_{\mathcal{X}} C(x, \pi^*(\cdot|x)) d\mathbb{P}_0(x) - \int_{\mathcal{Y}} f^*(y) d\pi_1^*(y) + \int_{\mathcal{Y}} f^*(y) d\mathbb{P}_1(y) =$$

$$\int_{\mathcal{X}} C(x, \pi^*(\cdot|x)) d\mathbb{P}_0(x) + \underbrace{\int_{\mathcal{Y}} f^*(y) d(\mathbb{P}_1 - \pi_1^*)(y)}_{=0 \text{ since } \pi_1^* = \mathbb{P}_1} =$$

$$\int_{\mathcal{X}} C(x, \pi^*(\cdot|x)) d\mathbb{P}_0(x) = \textbf{Cost}(\pi^*). \tag{18}$$

We see that **Cost**(π^*) is not greater than the optimal **Cost**$(\mathbb{P}_0, \mathbb{P}_1)$, i.e., π^* is optimal. At the same time, from the derivations above, it directly follows that f^* is an optimal potential. □

Proof of Theorem 3.2. We are going to use our Theorem 3.1. First, we check that (13) holds for $\pi^*(\cdot|x)$ defined by (14). Analogously to [42, Theorem 1], for each $x \in \mathcal{X}$, we derive

$$\inf_{\nu \in \mathcal{P}_p(\mathcal{Y})} \{C_{c,\epsilon}(x,\nu) - \int_{\mathcal{Y}} f^*(y) d\nu(y)\} = \inf_{\nu \in \mathcal{P}_p(\mathcal{Y})} \underbrace{\Big\{ \int_{\mathcal{Y}} \big[c(x,y) - f^*(y)\big] d\nu(y) - \epsilon H(\nu) \Big\}}_{\overset{\text{def}}{=} \mathcal{G}_x(\nu)}.$$

Minimizing $\mathcal{G}_x$, one should consider only $\nu \in \mathcal{P}_{p,ac}(\mathcal{Y}) \subset \mathcal{P}_p(\mathcal{Y})$. Indeed, for $\nu \notin \mathcal{P}_{p,ac}(\mathcal{Y})$, it holds that $\mathcal{G}_x(\nu^*) = +\infty$ since $c(x,y)$ is lower bounded and $-H(\nu) = +\infty$. We continue

$$\inf_{\nu \in \mathcal{P}_{p,ac}(\mathcal{Y})} \Big\{ -\epsilon \int_{\mathcal{Y}} \log \exp \Big(\frac{f^*(y) - c(x,y)}{\epsilon} \Big) d\nu(y) + \epsilon \overbrace{\int_{\mathcal{Y}} \log \frac{d\nu(y)}{dy} d\nu(y)}^{= -H(\nu)} \Big\} =$$

$$\inf_{\nu \in \mathcal{P}_{p,ac}(\mathcal{Y})} \Big\{ -\epsilon \int_{\mathcal{Y}} \log \big(Z_x \cdot \frac{d\pi^*(y|x)}{dy} \big) d\nu(y) + \epsilon \int_{\mathcal{Y}} \log \frac{d\nu(y)}{dy} d\nu(y) \Big\} =$$

$$-\epsilon \log Z_x + \inf_{\nu \in \mathcal{P}_{p,ac}(\mathcal{Y})} \Big\{ -\epsilon \int_{\mathcal{Y}} \log \frac{d\pi^*(y|x)}{dy} d\nu(y) + \epsilon \int_{\mathcal{Y}} \log \frac{d\nu(y)}{dy} d\nu(y) \Big\} =$$

$$-\epsilon \log Z_x + \inf_{\nu \in \mathcal{P}_{p,ac}} \epsilon \text{KL}\left(\nu \| \pi^*(\cdot|x)\right). \quad (19)$$

Since $\pi^*(\cdot|x) \in \mathcal{P}_{p,ac}(\mathcal{Y})$, by the assumption of the current Theorem, we conclude that it is the unique minimum of $\mathcal{G}_x(\nu)$ in $\mathcal{P}_{p,ac}(\mathcal{Y})$. Now to apply our Theorem 3.1, it remains to check that all its assumptions hold. We only have to check that $C_{c,\epsilon}$ given by (6) is lower bounded, jointly lower semi-continuous and convex in the second argument.

Analogously to (19), we derive

$$C_{c,\epsilon}(x,\nu) = \int_{\mathcal{Y}} c(x,y)d\nu(y) - \epsilon H(\nu) = \underbrace{-\epsilon \log M_x}_{\geq -\epsilon \log M} + \epsilon \underbrace{\text{KL}\left(\nu \| \nu_x\right)}_{\geq 0} \geq -\epsilon \log M, \quad (20)$$

where $\frac{d\nu_x(y)}{dy} \stackrel{\text{def}}{=} M_x^{-1} \exp\big(-\frac{c(x,y)}{\epsilon}\big)$. This provides a lower bound on the cost $C_{c,\epsilon}$. From the first equality in (20), we see that $C_{c,\epsilon}$ is jointly lower semi-continuous because the first term $\int_{\mathcal{Y}} c(x,y)d\nu(y)$ is jointly lower semi-continuous by the assumptions and the entropy term $-H(\nu)$ is lower semi-continuous in $\mathcal{P}_1(\mathcal{Y})$ [48, Ex. 45] and hence in $\mathcal{P}_p(\mathcal{Y})$ as well ($p \geq 1$). The last step is to note that $C_{c,\epsilon}(x,\nu)$ is convex in ν thanks to the convexity of $-H(\nu)$.

Finally, if $\int_{\mathcal{X}} C_{c,\epsilon}\big(x,\pi^*(\cdot|x)\big)d\mathbb{P}_0(x) < \infty$, then $-\int_{\mathcal{X}} H\big(\pi^*(\cdot|x)\big)$ is finite. Let U be the subset of plans $\pi \subset \Pi(\mathbb{P}_0,\mathbb{P}_1)$ where $-\int_{\mathcal{X}} H\big(\pi(\cdot|x)\big)$ is finite. It is not empty since $\pi^* \in U$. At the same time, it is a convex set and functional $\pi \mapsto -\int_{\mathcal{X}} H\big(\pi(\cdot|x)\big)d\mathbb{P}_0(x)$ is **strictly** convex in U thanks to the strict convexity of the (negative) entropy $\nu \mapsto -H(\nu)$ on the set of distributions where it is finite. Thus, $\pi \mapsto \int_{\mathcal{X}} C_{c,\epsilon}\big(x,\pi(\cdot|x)\big)d\mathbb{P}_0(x)$ is strictly convex in U and π^* is the unique minimum.

For completeness, we note that if $\int_{\mathcal{X}} C_{c,\epsilon}\big(x,\pi^*(\cdot|x)\big)d\mathbb{P}_0(x) = +\infty$, this situation is trivial, as the cost of every plan turns to be equal to $+\infty$. As a result, every plan is optimal.

□

Proof of Proposition 3.3. Deriving the actual form of $\pi^*(\cdot|x)$ is an easy exercise. We substitute (15) into (14) and use the quadratic cost $c(x,y) = \frac{||y-x||^2}{2}$:

$$\frac{d\pi^*(y|x)}{dy} = \frac{1}{Z_x}\exp\left(\frac{f^*(y) - c(x,y)}{\epsilon}\right) =$$

$$\frac{1}{Z_x}\exp\left(\frac{\epsilon \log \sum_{n=1}^N w_n \mathcal{Q}(y|b_n, \epsilon^{-1}A_n) - \frac{||y-x||^2}{2}}{\epsilon}\right) =$$

$$\frac{1}{Z_x}\left(\sum_{n=1}^N w_n \mathcal{Q}(y|b_n,\epsilon^{-1}A_n)\right)\exp(-\frac{||y-x||^2}{2\epsilon}) =$$

$$\frac{1}{Z_x}\sum_{n=1}^N w_n\left(\mathcal{Q}(y|b_n,\epsilon^{-1}A_n)\exp(-\frac{||y-x||^2}{2\epsilon})\right) =$$

$$\frac{1}{Z_x}\sum_{n=1}^N w_n\left(\exp\big[-\frac{1}{2}(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n)\big]\exp(-\frac{||y-x||^2}{2\epsilon})\right) =$$

$$\frac{1}{Z_x}\sum_{n=1}^N w_n\left(\exp\big[-\frac{1}{2}(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n) - \frac{||y-x||^2}{2\epsilon}\big]\right) =$$

$$\frac{1}{Z_x}\sum_{n=1}^N w_n\left(\exp\big[-\frac{1}{2}(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n) - \frac{1}{2}(y-x)^T\frac{I}{\epsilon}(y-x)\big]\right) =$$

$$\frac{1}{Z_x}\sum_{n=1}^N w_n\left(\exp\big[-\frac{1}{2}\{(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n) + (y-x)^T\frac{I}{\epsilon}(y-x)\}\big]\right). \quad (21)$$

Next, we prove that (we write just μ_n instead of $\mu_n(x)$ for simplicity):

$$(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n) + (y-x)^T\frac{I}{\epsilon}(y-x) =$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n). \tag{22}$$

Indeed,

$$(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n)+(y-x)^T\frac{I}{\epsilon}(y-x)=$$

$$y^T\frac{A_n}{\epsilon}y-2b_n^T\frac{A_n}{\epsilon}y+b_n^T\frac{A_n}{\epsilon}b_n^T+y^T\frac{I}{\epsilon}y-2x^T\frac{I}{\epsilon}y+x^T\frac{I}{\epsilon}x=$$

$$y^T\underbrace{(\frac{A_n+I}{\epsilon})}_{\Sigma_n^{-1}}y-2(A_nb_n+x)^T\frac{I}{\epsilon}y+b_n^T\frac{A_n}{\epsilon}b_n+x^T\frac{I}{\epsilon}x=$$

$$y^T\Sigma_n^{-1}y-2(A_nb_n+x)^T\frac{I}{\epsilon}y+b_n^T\frac{A_n}{\epsilon}b_n+x^T\frac{I}{\epsilon}x=$$

$$y^T\Sigma_n^{-1}y-2\underbrace{(A_nb_n+x)^T(A_n+I)^{-1}}_{\mu_n^T}\underbrace{\frac{(A_n+I)}{\epsilon}}_{\Sigma_n^{-1}}y+b_n^T\frac{A_n}{\epsilon}b_n+x^T\frac{I}{\epsilon}x=$$

$$y^T\Sigma_n^{-1}y-2\mu_n^T\Sigma_n^{-1}y+b_n^T\frac{A_n}{\epsilon}b_n+x^T\frac{I}{\epsilon}x=$$

$$y^T\Sigma_n^{-1}y-2\mu_n^T\Sigma_n^{-1}y+\mu_n^T\Sigma_n^{-1}\mu_n-\mu_n^T\Sigma_n^{-1}\mu_n+b_n^T\frac{A_n}{\epsilon}b_n+x^T\frac{I}{\epsilon}x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)-\mu_n^T\Sigma_n^{-1}\mu_n+b_n^T\frac{A_n}{\epsilon}b_n+x^T\frac{I}{\epsilon}x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n}{\epsilon}b_n-\mu_n^T\Sigma_n^{-1}\mu_n+x^T\frac{I}{\epsilon}x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n}{\epsilon}b_n-(A_nb_n+x)^T\frac{\Sigma_n}{\epsilon}\Sigma_n^{-1}\frac{\Sigma_n}{\epsilon}(A_nb_n+x)+x^T\frac{I}{\epsilon}x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n}{\epsilon}b_n-(A_nb_n+x)^T\frac{\Sigma_n}{\epsilon^2}(A_nb_n+x)+x^T\frac{I}{\epsilon}x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n}{\epsilon}b_n-$$

$$(A_nb_n)^T\frac{\Sigma_n}{\epsilon^2}A_nb_n-2(A_nb_n)^T\frac{\Sigma_n}{\epsilon^2}x-x^T\frac{\Sigma_n}{\epsilon^2}x+x^T\frac{I}{\epsilon}x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n}{\epsilon}b_n-(A_nb_n)^T\frac{\Sigma_n}{\epsilon^2}A_nb_n-$$

$$2(A_nb_n)^T\frac{\Sigma_n}{\epsilon^2}x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n}{\epsilon}b_n-b_n^T\frac{A_n^T\Sigma_nA_n}{\epsilon^2}b_n-$$

$$2(A_nb_n)^T\frac{\Sigma_n}{\epsilon^2}x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-2(A_nb_n)^T\frac{\Sigma_n}{\epsilon^2}x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-2b_n^T\frac{A_n\Sigma_n}{\epsilon^2}x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-$$

$$2b_n^T\frac{\overbrace{(A_n+I)}^{\epsilon\Sigma_n^{-1}}\Sigma_n-\Sigma_n}{\epsilon^2}x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=$$

$$(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-2b_n^T\frac{\epsilon I-\Sigma_n}{\epsilon^2}x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=$$

$$
\begin{aligned}
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-2b_n^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x+x^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})x=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-\\
&b_n^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})b_n+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+\\
&b_n^T\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}b_n-b_n^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+\\
&b_n^T(\frac{A_n-A_n^T\frac{\Sigma_n}{\epsilon}A_n}{\epsilon}-\frac{I}{\epsilon}+\frac{\Sigma_n}{\epsilon^2})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+\\
&b_n^T(\frac{A_n(I-\frac{\Sigma_n}{\epsilon}A_n)}{\epsilon}-\frac{I}{\epsilon}+\frac{\Sigma_n}{\epsilon^2})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+\\
&b_n^T(\frac{A_n(I-\frac{\Sigma_n}{\epsilon}(\epsilon\Sigma_n^{-1}-I))}{\epsilon}-\frac{I}{\epsilon}+\frac{\Sigma_n}{\epsilon^2})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+\\
&b_n^T(\frac{A_n(\frac{\Sigma_n}{\epsilon})-I+\frac{\Sigma_n}{\epsilon}}{\epsilon})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+\\
&b_n^T(\frac{(\epsilon\Sigma_n^{-1}-I)\frac{\Sigma_n}{\epsilon}-I+\frac{\Sigma_n}{\epsilon}}{\epsilon})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)+b_n^T(\frac{I-\frac{\Sigma_n}{\epsilon}-I+\frac{\Sigma_n}{\epsilon}}{\epsilon})b_n=\\
&(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n).
\end{aligned}
$$

Next, we substitute (22) into (21)

$$
\begin{aligned}
&\frac{1}{Z_x}\sum_{n=1}^{N}w_n\Big(\exp\big[-\frac{1}{2}\{(y-b_n)^T\frac{A_n}{\epsilon}(y-b_n)+(y-x)^T\frac{I}{\epsilon}(y-x)\}\big]\Big)=\\
&\frac{1}{Z_x}\sum_{n=1}^{N}w_n\Big(\exp\big[-\frac{1}{2}\{(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n)+(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n)\}\big]\Big)=\\
&\frac{1}{Z_x}\sum_{n=1}^{N}w_n\exp(-\frac{1}{2}(y-\mu_n)^T\Sigma_n^{-1}(y-\mu_n))\exp(-\frac{1}{2}(x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})(x-b_n))=\\
&\frac{1}{Z_x}\sum_{n=1}^{N}w_n(2\pi)^{\frac{D}{2}}\sqrt{\det(\Sigma_n)}\mathcal{N}(y|\mu_n,\Sigma_n)\mathcal{Q}(x|b_n,\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})=\\
&\frac{1}{Z_x}\sum_{n=1}^{N}\underbrace{w_n(2\pi)^{\frac{D}{2}}\sqrt{\det(\Sigma_n)}\mathcal{Q}(x|b_n,\frac{I}{\epsilon}-\frac{\Sigma_n}{\epsilon^2})}_{\widetilde{w}_n}\mathcal{N}(y|\mu_n,\Sigma_n))=
\end{aligned}
$$

$$\frac{1}{Z_x}\sum_{n=1}^{N}\widetilde{w}_n\mathcal{N}(y|\mu_n,\Sigma_n)=\frac{1}{\sum_{n=1}^{N}\widetilde{w}_n}\sum_{n=1}^{N}\widetilde{w}_n\mathcal{N}(y|\mu_n,\Sigma_n)=$$
$$\sum_{n=1}^{N}\frac{\widetilde{w}_n}{\sum_{n=1}^{N}\widetilde{w}_n}\mathcal{N}(y|\mu_n,\Sigma_n)=\sum_{n=1}^{N}\gamma_n\mathcal{N}(y|\mu_n,\Sigma_n).$$

which finishes the derivation of the expression for the density of $\pi^*(\cdot|x)$.

Now we prove that $\mathbb{P}_1\overset{\text{def}}{=}\pi_1^*\in\mathcal{P}_2(\mathcal{Y})$. For each x, consider $\frac{d\pi^*(y|x)}{dy}=\sum_{n=1}^{N}\gamma_n\mathcal{N}(y|\mu_n(x),\Sigma_n)$. Its second moment is given by $\sum_{n=1}^{N}\gamma_n\big(\|\mu_n(x)\|^2+\operatorname{Tr}\Sigma_n\big)$. Note that

$$\|\mu_n(x)\|=\|(A_n+I)^{-1}(A_nb_n+x)\|\leq$$
$$\|(A_n+I)^{-1}\|\cdot\|A_nb_n+x\|\leq\|(A_n+I)^{-1}\|\cdot(\|A_nb_n\|+\|x\|),$$

where $\|\cdot\|$ applied to matrix means the operator norm. Hence, one may conclude that $\|\mu_n(x)\|^2$ is upper bounded by some quadratic polynomial of $\|x\|$, i.e., there exist constants $\alpha_n\in\mathbb{R},\beta_n\in\mathbb{R}_+$ such that $\|\mu_n(x)\|^2\leq\alpha_n+\beta_n\cdot\|x\|^2$. We derive

$$\int_{\mathcal{Y}}\|y\|^2d\pi_1^*(y)=\int_{\mathcal{X}}\int_{\mathcal{Y}}\|y\|^2d\pi^*(y|x)\underbrace{d\pi_0^*}_{=d\mathbb{P}_0(x)}(x)=\int_{\mathcal{X}}\sum_{n=1}^{N}\gamma_n\big(\|\mu_n(x)\|^2+\operatorname{Tr}\Sigma_n\big)d\mathbb{P}_0(x)\leq$$
$$\int_{\mathcal{X}}\sum_{n=1}^{N}\gamma_n\big(\alpha_n+\beta_n\|x\|^2+\operatorname{Tr}\Sigma_n\big)d\mathbb{P}_0(x)=$$
$$\sum_{n=1}^{N}\gamma_n\big(\alpha_n+\operatorname{Tr}\Sigma_n\big)+\big(\sum_{n=1}^{N}\beta_n\gamma_n\big)\int_{\mathcal{X}}\|x\|^2d\mathbb{P}_0(x)<\infty$$

since $\mathbb{P}_0\in\mathcal{P}_2(\mathcal{X})$ by the assumption of the proposition.

It remains to prove that π^* is the unique EOT plan. According to our Theorem 3.2, one only has to ensure that $\int_{\mathcal{X}}C_{c,\epsilon}\big(x,\pi^*(\cdot|x)\big)d\mathbb{P}_0(x)<\infty$. Just for completeness, we highlight that $\int_{\mathcal{X}}C_{c,\epsilon}\big(x,\pi^*(\cdot|x)\big)d\mathbb{P}_0(x)$ is *lower*-bounded since $C_{c,\epsilon}$ is lower bounded, see the proof of Theorem 3.2. Anyway, this is indifferent for us. We recall that π^* is an optimal plan between $\mathbb{P}_0$ and $\mathbb{P}_1=\pi_1^*$ and f^* is an optimal potential by our construction. Thanks to the duality, we have

$$\int_{\mathcal{X}}C_{c,\epsilon}\big(x,\pi^*(\cdot|x)\big)d\mathbb{P}_0(x)=\int_{\mathcal{X}}(f^*)^{C_{c,\epsilon}}(x)d\mathbb{P}_0(x)+\int_{\mathcal{Y}}f^*(y)d\mathbb{P}_1(y)=$$
$$\int_{\mathcal{X}}\big[-\epsilon\log Z_x\big]d\mathbb{P}_0(x)+\int_{\mathcal{Y}}f^*(y)d\mathbb{P}_1(y),\tag{23}$$

where in transition to (23) we used our findings of line (19). Note that $\int_{\mathcal{Y}}f^*(y)d\mathbb{P}_1(y)$ is finite since $f^*\in\mathcal{C}_2(\mathcal{Y})$ is dominated by a quadratic polynomial, and we have already proved that $\mathbb{P}_1$ has finite second moment. It remains to upper bound the first term in (23). We note that

$$Z_x=\int_{\mathcal{Y}}\exp\bigg(\frac{f^*(y)-\frac{1}{2}\|x-y\|^2}{\epsilon}\bigg)dy=(\sqrt{2\pi\epsilon})^D\int_{\mathcal{Y}}\exp\bigg(\frac{f^*(y)}{\epsilon}\bigg)\mathcal{N}(y|x,\epsilon I)dy\geq$$
$$(\sqrt{2\pi\epsilon})^D\exp\bigg(\int_{\mathcal{Y}}\frac{f^*(y)}{\epsilon}\mathcal{N}(y|x,\epsilon I)dy\bigg)\geq(\sqrt{2\pi\epsilon})^D\exp\bigg(\int_{\mathcal{Y}}\frac{\beta+\alpha\|y\|^2}{\epsilon}\mathcal{N}(y|x,\epsilon I)dy\bigg)=\tag{24}$$
$$(\sqrt{2\pi\epsilon})^D\exp\bigg(\frac{\beta+\alpha(\|x\|^2+\epsilon D)}{\epsilon}\bigg),\tag{25}$$

where in transition to line (24) we used the Jesnsen's inequality and $\alpha,\beta\in\mathbb{R}$ are some constants for which $f^*(\cdot)\geq\beta+\alpha\|\cdot\|^2$. They exist since $f^*\in\mathcal{C}_2(\mathcal{Y})$. Indeed, there exist $\tilde{\alpha},\tilde{\beta}:\ |f^*(\cdot)|\leq\tilde{\beta}+\tilde{\alpha}\|\cdot\|^2\Rightarrow f^*(\cdot)\geq-\tilde{\beta}-\tilde{\alpha}\|\cdot\|^2$, and we set $\alpha=-\tilde{\alpha},\beta=-\tilde{\beta}$. In turn, line (25) uses the explicit formula for the second moment of $\mathcal{N}(y|x,\epsilon I)$. We use (25) to upper bound the first term in (23):

$$\int_{\mathcal{X}}\big[-\epsilon\log Z_x\big]d\mathbb{P}_0(x)\leq\int_{\mathcal{X}}\big[-\epsilon\log\bigg(\{(\sqrt{2\pi\epsilon})^D\exp\bigg(\frac{\beta+\alpha\|x\|^2+\alpha\epsilon D}{\epsilon}\bigg)\bigg\}\big]d\mathbb{P}_0(x)=$$

$$-\frac{\epsilon D}{2}\log(2\pi\epsilon)-\beta-\alpha\epsilon D-\alpha\int_{\mathcal{X}}\|x\|^2 d\mathbb{P}_0(x).$$

It remains to note that the last value is finite, since $\mathbb{P}_0 \in \mathcal{P}_2(\mathcal{X})$ by the assumption. □

Proof of Corollary 3.4. We note that $\frac{d\pi^*(y|x)}{dy} \propto \exp\big(\frac{f^*(y)-\frac{1}{2}\|x-y\|^2}{\epsilon}\big)$. Therefore,

$$\exp\big(\frac{f^*(y)}{\epsilon}\big) \propto \frac{d\pi^*(y|x)}{dy}\exp\big(\frac{1}{2\epsilon}\|x-y\|^2\big) \propto \frac{d\pi^*(y|x)}{dy}\cdot\big[\mathcal{N}(y|x,\epsilon I)\big]^{-1}. \tag{26}$$

By comparing (26) with (11), we see that $\exp\big(\frac{f^*(y)}{\epsilon}\big)$ indeed coincides with the Schrödinger potential $\phi^*(y)$. Formula (12) for the optimal drift follows from [38, Proposition 4.1][1]. □

Proof of Corollary 3.5. First, we prove that constructed $\mathbb{P}_1 \stackrel{\text{def}}{=} \pi_1^*$ actually has finite entropy. This is needed to ensure that the assumptions of [38, Proposition 4.1]. This proposition provides the formula for the optimal drift (12) via the Schrödinger potential. We write

$$0 \leq \text{KL}\left(\pi_1^* \| \mathcal{N}(\cdot|0,I)\right) = -H(\pi_1^*) - \int_{\mathcal{Y}} \log\mathcal{N}(y|0,I)d\pi_1^*(y) =$$
$$-H(\pi_1^*) + \frac{D}{2}\log(2\pi) + \frac{1}{2}\int_{\mathcal{Y}}\|y\|^2 d\pi_1^*(y). \tag{27}$$

From our Proposition 3.3 it follows that $\mathbb{P}_1 = \pi_1^*$ has finite second moment. Hence, the latter constant in (27) is finite. Therefore, $H(\pi_1^*)$ is upper bounded. To lower bound $H(\pi_1^*)$, recall that each $\pi^*(\cdot|x)$ is a mixture of N Gaussians (Proposition 3.3) with (x-independent) covariances Σ_n. Thus, its density $\frac{d\pi^*(y|x)}{dy}$ is upper bounded by $\xi \stackrel{\text{def}}{=} \max_n \big[(2\pi)^{-D/2}\big](\det\Sigma_n)^{-1/2} > 0$ which also means that

$$\frac{d\pi_1^*(y)}{dy} = \int_{\mathcal{X}} \frac{d\pi^*(y|x)}{dy} d\pi_0^*(x) \leq \int_{\mathcal{X}} \xi d\pi_0^*(x) \leq \xi.$$

We conclude that

$$H(\pi_1^*) = -\int \log\frac{d\pi_1^*(y)}{dy}d\pi_1^*(y) \geq -\int \log\xi d\pi_1^*(y) = -\log\xi, \tag{28}$$

i.e., $H(\pi_1^*)$ is lower-bounded as well.

Having in mind our previous Corollary, we just substitute $\exp\big(\frac{f^*(y)}{\epsilon}\big)$ of LSE (15) potential f^* as the Schrödinger potential $\phi^*(y)$ to (12). We derive

$$v^*(x,t) = \epsilon\nabla\log\int_{\mathbb{R}^D}\mathcal{N}(y|x,(1-t)\epsilon I)\varphi^*(y)dy =$$
$$\epsilon\nabla\log\int_{\mathbb{R}^D}\mathcal{N}(y|x,(1-t)\epsilon I)\exp(\frac{f^*(y)}{\epsilon})dy =$$
$$\epsilon\nabla\log\int_{\mathbb{R}^D}\mathcal{N}(y|x,(1-t)\epsilon I)\exp(\frac{\epsilon\log\sum_{n=1}^N w_n\mathcal{Q}(y|b_n,\epsilon^{-1}A_n)}{\epsilon})dy =$$
$$\epsilon\nabla\log\sum_{n=1}^N w_n\int_{\mathbb{R}^D}\mathcal{N}(y|x,(1-t)\epsilon I)\mathcal{Q}(y|b_n,\epsilon^{-1}A_n))dy =$$
$$\epsilon\nabla\log\sum_{n=1}^N w_n\int_{\mathbb{R}^D}\big(2\pi\epsilon(1-t)\big)^{-\frac{D}{2}}\exp(-(y-x)^T\frac{I}{2\epsilon(1-t)}(y-x))\mathcal{Q}(y|b_n,\epsilon^{-1}A_n)dy =$$

[1]The authors of [38] consider SB with the *reversible* Wiener prior R, i.e., the standard Brownian motion starting at the Lebesgue measure. They deal with $\inf_{T\in\mathcal{F}(\mathbb{P}_0,\mathbb{P}_1)}\text{KL}\left(T\|R\right)$ which matches (up to an additive constant) our formulation (9) for $\epsilon = 1$. Indeed, using the measure disintegration theorem, one can derive $\text{KL}\left(T\|R\right) = -H(\mathbb{P}_0) + \text{KL}\left(T\|W^\epsilon\right)$. For other $\epsilon > 0$, the analogous equivalence holds true.

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\int_{\mathbb{R}^D}\exp(-(y-x)^T\frac{I}{2\epsilon(1-t)}(y-x))\mathcal{Q}(y|b_n,\epsilon^{-1}A_n)dy+ \underbrace{\epsilon\nabla\log\big(\big(2\pi\epsilon(1-t)\big)^{-\frac{D}{2}}\big)}_{=0}=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\int_{\mathbb{R}^D}\exp(-(y-x)^T\frac{I}{2\epsilon(1-t)}(y-x))\exp(-(y-b_n)^T\frac{A_n}{2\epsilon}(y-b_n))dy=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\int_{\mathbb{R}^D}\exp\big(-\frac{1}{2(1-t)}\{(y-x)^T\frac{I}{\epsilon}(y-x)+(y-b_n)^T\frac{\overbrace{(1-t)A_n}^{A_n^t}}{\epsilon}(y-b_n)\}\big)dy$$

Next, we use (22) but with A_n^t instead of A_n and Σ_n^t instead of Σ_n. Also, we denote $\mu_n^t=(A_n^t+I)^{-1}(A_n^t b_n+x)$:

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\int_{\mathbb{R}^D}\exp\big(-\frac{1}{2(1-t)}\{(y-x)^T\frac{I}{\epsilon}(y-x)+(y-b_n)^T\frac{\overbrace{(1-t)A_n}^{A_n^t}}{\epsilon}(y-b_n)\}\big)dy=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\int_{\mathbb{R}^D}\exp\Bigg(-\frac{1}{2(1-t)}\big\{(y-\mu_n^t)^T\left(\Sigma_n^t\right)^{-1}(y-\mu_n^t)+ (x-b_n)^T(\frac{I}{\epsilon}-\frac{\Sigma_n^t}{\epsilon^2})(x-b_n)\big\}\Bigg)dy=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}\Big\{w_n\exp\big(-\frac{1}{2}(x-b_n)^T\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}(x-b_n)\big) \int_{\mathbb{R}^D}\exp\big(-\frac{1}{2}(y-\mu_n^t)^T\frac{\left(\Sigma_n^t\right)^{-1}}{(1-t)}(y-\mu_n^t)\big)dy\Big\}=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\mathcal{Q}\big(x\big|b_n,\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}\big)\int_{\mathbb{R}^D}\exp\big(-\frac{1}{2}(y-\mu_n^t)^T\frac{\left(\Sigma_n^t\right)^{-1}}{(1-t)}(y-\mu_n^t)\big)dy=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\mathcal{Q}\big(x\big|b_n,\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}\big)\int_{\mathbb{R}^D}(2\pi)^{\frac{D}{2}}\det((1-t)\Sigma_n^t)^{\frac{1}{2}}\mathcal{N}(y|\mu_n^t,(1-t)\Sigma_n^t)dy=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\mathcal{Q}\big(x\big|b_n,\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}\big)(2\pi(1-t))^{\frac{D}{2}}\det(\Sigma_n^t)^{\frac{1}{2}}\underbrace{\int_{\mathbb{R}^D}\mathcal{N}(y|\mu_n^t,(1-t)\Sigma_n^t)dy}_{=1}=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\mathcal{Q}\big(x\big|b_n,\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}\big)(2\pi(1-t))^{\frac{D}{2}}\det(\Sigma_n^t)^{\frac{1}{2}}=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\mathcal{Q}\big(x\big|b_n,\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}\big)\det(\Sigma_n^t)^{\frac{1}{2}}+\underbrace{\epsilon\nabla\log\big((2\pi(1-t))^{\frac{D}{2}}\big)}_{=0}=$$

$$\epsilon\nabla\log\sum_{n=1}^{N}w_n\sqrt{\det(\Sigma_n^t)}\mathcal{Q}\big(x\big|b_n,\frac{\epsilon I-\Sigma_n^t}{\epsilon^2(1-t)}\big),$$

which finishes the proof. □

B Mixtures Benchmark Pairs: Details and Results

Parameters for constructing benchmark pairs. In our benchmark pairs, we choose all their hyperparameters manually to make sure the constructed distributions $\mathbb{P}_0,\mathbb{P}_1$ are visually pleasant and distinguishable. As $\mathbb{P}_0$, we always use the centered Gaussian whose covariance matrix is $0.25I$. We use LSE function (15) with $N = 5$ for constructing the distribution $\mathbb{P}_1$. In each setup, all A_n are the same and given in Table 3. We pick w_n such that $\gamma_n = \frac{1}{5}\mathcal{N}(x|b_n, (\frac{1}{\epsilon}I - \frac{1}{\epsilon^2}\Sigma_n)^{-1})$. We sample b_n randomly from a uniform distribution on a sphere with the radius $R = 5$.

	$D=2$	$D=16$	$D=64$	$D=128$
$\epsilon=0.1$	$\frac{1}{16}I$	$\frac{1}{16}I$	$\frac{1}{16}I$	$\frac{1}{16}I$
$\epsilon=1$	$\frac{1}{16}I$	$\frac{1}{16}I$	$\frac{1}{16}I$	$\frac{1}{16}I$
$\epsilon=10$	$\frac{9}{40}I$	$\frac{1}{100}I$	$\frac{1}{100}I$	$\frac{1}{100}I$

Table 3: Matrices A_n that we use to construct our mixtures benchmark pairs.

Evaluation details. For computing $\text{B}\mathbb{W}_2^2\text{-UVP}(\widehat{\pi}_1, \mathbb{P}_1)$, we use 10^5 random samples from $\mathbb{P}_1$ and 10^5 random samples from learned distribution $\widehat{\pi}_1$. For computing $\text{cB}\mathbb{W}_2^2\text{-UVP}(\widehat{\pi}, \pi^*)$, we use the hold-out test set containing 1000 samples $x \sim \mathbb{P}_0$. We compute the expectation and covariance matrices of $\pi^*(\cdot|x)$ analytically (Proposition 3.3) and we estimate the expectation and covariance matrix of $\widehat{\pi}(\cdot|x)$ by using 10^3 samples. We present results of evaluation in Table 4 and Table 5.

We present an additional *trivial* baseline for the conditional metric $\text{cB}\mathbb{W}_2^2\text{-UVP}(\widehat{\pi}, \pi^*)$, which is given by the independent plan $\mathbb{P}_0 \times \mathbb{P}_1$. We compare other methods with this baseline in Table 5.

	$\epsilon=0.1$				$\epsilon=1$				$\epsilon=10$			
	$D=2$	$D=16$	$D=64$	$D=128$	$D=2$	$D=16$	$D=64$	$D=128$	$D=2$	$D=16$	$D=64$	$D=128$
⌊LSOT⌉	-	-	-	-	-	-	-	-	-	-	-	-
⌊SCONES⌉	-	-	-	-	1.06	4.24	6.67	11.54	1.11	2.98	1.33	7.89
⌊NOT⌉	0.016	0.63	1.53	2.62	0.08	1.13	1.62	2.62	0.225	2.603	1.872	6.12
⌊EgNOT⌉	0.09	0.31	0.88	0.22	0.46	0.3	0.85	0.12	0.077	0.02	0.15	0.23
⌊ENOT⌉	0.2	2.9	1.8	1.4	0.22	0.4	7.8	29	1.2	2	18.9	28
⌊MLE-SB⌉	0.01	0.14	0.97	2.08	0.005	0.09	0.56	1.46	0.01	1.02	6.65	23.4
⌊DiffSB⌉	2.88	2.81	153.22	232.67	0.87	0.99	1.12	1.56	-	-	-	-
⌊FB-SDE-A⌉	2.37	2.55	68.19	27.11	0.6	0.63	0.65	0.71	-	-	-	-
⌊FB-SDE-J⌉	0.03	0.05	0.25	2.96	0.07	0.13	1.52	0.48	-	-	-	-

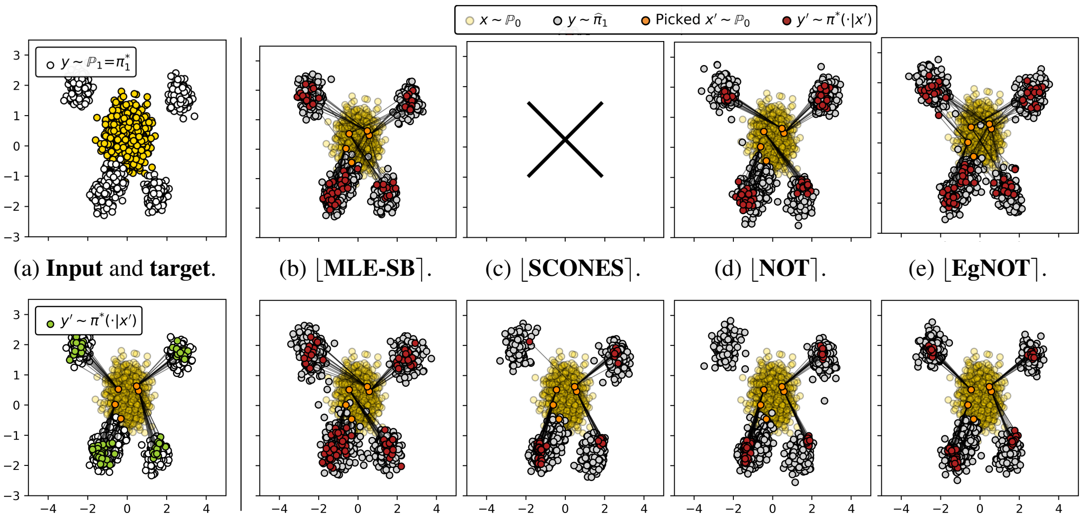
Table 4: Comparisons of $\text{B}\mathbb{W}_2^2\text{-UVP}\downarrow(\%)$ between the target $\mathbb{P}_1$ and learned marginal π_1. Colors indicate the metric value: $\text{B}\mathbb{W}_2^2\text{-UVP} \leq 0.5$, $\text{B}\mathbb{W}_2^2\text{-UVP} \in (0.5, 1]$, $\text{B}\mathbb{W}_2^2\text{-UVP} > 1.0$.

	$\epsilon=0.1$				$\epsilon=1$				$\epsilon=10$			
	$D=2$	$D=16$	$D=64$	$D=128$	$D=2$	$D=16$	$D=64$	$D=128$	$D=2$	$D=16$	$D=64$	$D=128$
⌊LSOT⌉	-	-	-	-	-	-	-	-	-	-	-	-
⌊SCONES⌉	-	-	-	-	34.88	71.34	59.12	136.44	32.9	50.84	60.44	52.11
⌊NOT⌉	1.94	13.67	11.74	11.4	4.77	23.27	41.75	26.56	2.86	4.57	3.41	6.56
⌊EgNOT⌉	129.8	75.2	60.4	43.2	80.4	74.4	63.8	53.2	4.14	2.64	2.36	1.31
⌊ENOT⌉	3.64	22	13.6	12.6	1.04	9.4	21.6	48	1.4	2.4	19.6	30
⌊MLE-SB⌉	4.57	16.12	16.1	17.81	4.13	9.08	18.05	15.226	1.61	1.27	3.9	12.9
⌊DiffSB⌉	73.54	59.7	1386.4	1683.6	33.76	70.86	53.42	156.46	-	-	-	-
⌊FB-SDE-A⌉	86.4	53.2	1156.82	1566.44	30.62	63.48	34.84	131.72	-	-	-	-
⌊FB-SDE-J⌉	51.34	89.16	119.32	173.96	29.34	69.2	155.14	177.52	-	-	-	-
Independent	166.0	152.0	126.0	110.0	86.0	80.0	72.0	60.0	4.2	2.52	2.26	2.4

Table 5: Comparisons of $\text{cB}\mathbb{W}_2^2\text{-UVP}\downarrow(\%)$ between the optimal plan π^* and the learned plan $\widehat{\pi}$. **Colors** indicate the ratio of the metric to the *independent baseline* metric: ratio ≤ 0.2, ratio $\in (0.2, 0.5)$, ratio > 0.5.

Colors for the Table 2. To assign a color for the metric $\text{B}\mathbb{W}_2^2\text{-UVP}$ and $\text{cB}\mathbb{W}_2^2\text{-UVP}$ for each ϵ in the Table 2, we use the following rule: we assign the rank 1 if a method's metric for a given dimension D has the color green, the rank 2 if a method's metric $\text{B}\mathbb{W}_2^2\text{-UVP}$ has the color orange and the rank 3 if a method's metric $\text{B}\mathbb{W}_2^2\text{-UVP}$ has the color red. To get the average rank, we take the mean of 4 ranks obtained for each dimension D and round it (1.5 and 2.5 are rounded to 1 and 2 respectively).

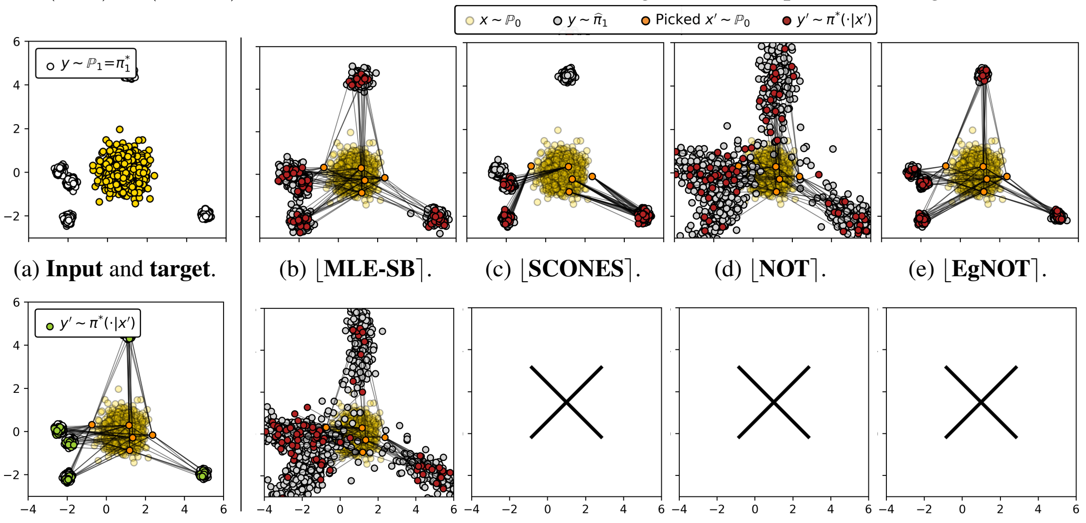
Extra qualitative results of EOT/SB solvers. In Figure 5 and Figure 6, we present the additional qualitative comparison of solvers on our mixtures benchmark pairs in $D = 16$ with $\epsilon \in \{0.1, 10\}$. The figures are designed similarly to Figure 3 for $(D, \epsilon) = (16, 1)$ in the main text. Note that case $\epsilon = 10$ (Figure 6) is extremely challenging; only **⌊EgNOT⌉** provides more-or-less reasonable results.



(a) **Input** and **target**. (b) ⌊**MLE-SB**⌉. (c) ⌊**SCONES**⌉. (d) ⌊**NOT**⌉. (e) ⌊**EgNOT**⌉.

(f) **True** EOT plan π^*. (g) ⌊**ENOT**⌉. (h) ⌊**DiffSB**⌉. (i) ⌊**FB-SDE-A**⌉. (j) ⌊**FB-SDE-J**⌉.

Figure 5: Qualitative results of EOT/SB solvers on our mixtures benchmark pair with $(D, \epsilon) = (16, 0.1)$. The distributions are visualized using 2 PCA components of target distr. $\mathbb{P}_1$.



(a) **Input** and **target**. (b) ⌊**MLE-SB**⌉. (c) ⌊**SCONES**⌉. (d) ⌊**NOT**⌉. (e) ⌊**EgNOT**⌉.

(f) **True** EOT plan π^*. (g) ⌊**ENOT**⌉. (h) ⌊**DiffSB**⌉. (i) ⌊**FB-SDE-A**⌉. (j) ⌊**FB-SDE-J**⌉.

Figure 6: Qualitative results of EOT/SB solvers on our mixtures benchmark pair with $(D, \epsilon) = (16, 10)$. The distributions are visualized using 2 PCA components of target distr. $\mathbb{P}_1$.

Computational complexity. Sampling from $\mathbb{P}_0$ is lightspeed as it is just sampling a Normal noise. Sampling from $\mathbb{P}_1$ is also fast, as it is the Gaussian mixture (Proposition 3.3).

C Images Benchmark Pairs: Details and Results

Parameters for constructing image benchmark pairs. We fix $N = 100$ random samples from $\mathbb{P}_0$ for b_n and choose all $A_n \equiv I$. We use w_n such that $\gamma_n = \frac{1}{100}\mathcal{N}(x|b_n, (\frac{1}{\epsilon}I - \frac{1}{\epsilon^2}\Sigma_n)^{-1})$.

GLOW details. We use the code from the repository with the default parameters:

`https://github.com/rosinality/glow-pytorch`

After training, the latent variable z is sampled from $N(0, \sigma^2 I)$ with $\sigma^2 = 0.49$ for image generation. That is, the image distribution $\mathbb{P}_0$ is produced by the mapping $z \sim N(0, \sigma^2 I)$ to the image space with the learned normalizing flow G, i.e., $\mathbb{P}_0 \overset{\text{def}}{=} G\sharp\mathcal{N}(\cdot|\sigma^2 I)$ in our construction.

MCMC in the latent space of the normalizing flow. We test EOT/SB solvers in $\mathbb{P}_1 \to \mathbb{P}_0$ direction, i.e., recovering $\pi^*(x|y)$ and generating clean samples x from noised y. Unfortunately, the reverse conditional OT plans $\pi^*(x|y)$ are not as tractable as $\pi^*(y|x)$. However, we note that

$$\frac{d\pi^*(x|y)}{dy} \propto \frac{d\pi^*(y|x)}{dy}\frac{d\mathbb{P}_0(x)}{dx}, \tag{29}$$

i.e., the density of $\pi^*(\cdot|y)$ it known up to the normalizing constant. Recall that here $\mathbb{P}_0$ is constructed using the normalizing flow and $\pi^*(\cdot|x)$ is a Gaussian mixture (Proposition 3.3), i.e., we indeed know the values of both terms. Therefore, one may use the well-celebrated Langevin dynamics to sample from $\pi^*(y|x)$. Unfortunately, we found that such sampling in the image space is rather slow.

To overcome this issue, we employ the Langevin sampling in the latent space of the normalizing flow. It is possible since the normalizing flow is a bijection between the space of images and the latent space. We use the standard notation z for the latent variable and $G : \mathbb{R}^D \to \mathbb{R}^D$ for the normalizing flow, i.e., $x = G(z) \sim \mathbb{P}_0$ for $z \sim p(z) \overset{\text{def}}{=} \mathcal{N}(z|0, \sigma^2 I)$. In this case, we have

$$\frac{d\pi^*(z|y)}{dz} = \frac{d\pi^*(x|y)}{dx}|\det J_{G^{-1}}(x)| \propto \frac{d\pi^*(y|x)}{dy}\frac{d\mathbb{P}_0(x)}{dx}|\det J_{G^{-1}}(x)| =$$
$$\frac{d\pi^*\big(y|G(z)\big)}{dx}\underbrace{\frac{d\mathbb{P}_0(x)}{dx}|\det J_{G^{-1}}(x)|}_{p(z)} = \frac{d\pi^*\big(y|G(z)\big)}{dy}p(z),$$

and we can derive the *score function* $\nabla_z \log \frac{d\pi^*(z|y)}{dz}$ which is needed for the Langevin dynamic as

$$\nabla_z \log \frac{d\pi^*(z|y)}{dz} = \nabla_z \frac{d\pi^*(y|G(z))}{dy} + \nabla_z \log p(z). \tag{30}$$

Hence, instead of doing non-trivial Langevin in the data space with $\nabla_x \frac{d\pi^*(x|y)}{dx}$, one may equivalently do the sampling in the latent space by using the score (30) and then get $x = G(z)$. We empirically found this approach works much better, presumably due to the fact that (30) is just the score of the Normal distribution which is slightly adjusted with the information coming from $\pi^*\big(y|G(z)\big)$.

For sampling, we employ the **Metropolis-adjusted Langevin algorithm** with the time steps $10^{-3}, 10^{-4}$ and 10^{-5} for $\epsilon = 10$, $\epsilon = 1$ and $\epsilon = 0.1$, respectively. It provides the theoretical guarantees that the constructed Markov chain $z_1, z_2, \dots, \dots$ converges to the distribution $\frac{d\pi^*(z|y)}{dz}$. For initializing the Markov chain, we sample a pair $(x, y) \sim \pi^*$ and use $z = G^{-1}(x)$ as the initial state for the Langevin sampling to get new samples from $\pi^*(\cdot|y)$. This trick allows for improving the stability of sampling and the convergence speed since it provides a good starting point. We use $N = 200$ steps for all the setups for the Metropolis-adjusted Langevin algorithm.

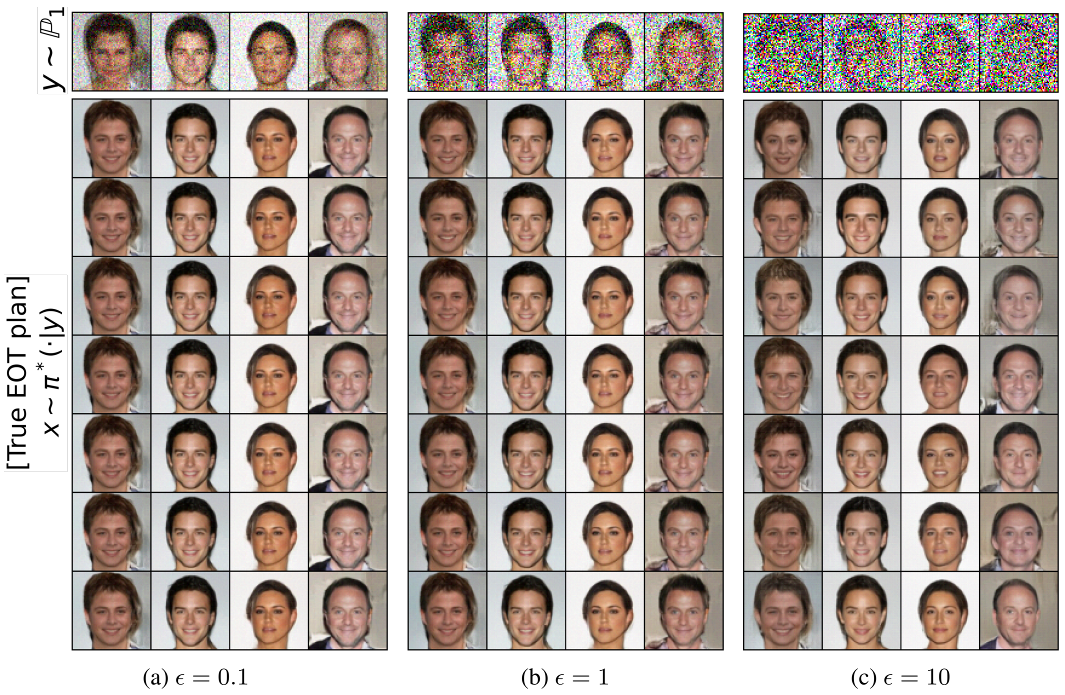
In Figures 7 and 8, we provide additional examples of the samples from the ground truth plan π^*.

Metric 1. For each $\epsilon = 0.1, 1, 10$ we prepare a test set with 10^4 samples from $\mathbb{P}_0$. We use this set to calculate the FID [24] metric between the ground truth distribution $\mathbb{P}_0$ and the model's marginal distribution π_1 to estimate how well the model restores the target distribution. This allows to access the generative performance of solvers, i.e., the quality of generated images and matching the target distribution. However, *this metric does not assess the accuracy of the recovered EOT plan.*

Metric 2. For each $\epsilon = 0.1, 1, 10$, we prepare a test set containing 100 "noised" samples $y \sim \mathbb{P}_1$ and 5K samples $x \sim \pi^*(\cdot|y)$ for each "noised" sample y, i.e., 5K $\times$ 100 images for each ϵ in consideration. We propose to compute **conditional FID** to evaluate the difference between the conditional plans $\pi^*(\cdot|y)$ and $\widehat{\pi}(\cdot|y)$. That is, for each y we compute FID between $\pi^*(\cdot|y)$ and $\widehat{\pi}(\cdot|y)$, and then average the result for all test y. Clearly, such an evaluation is approximately $100\times$times more consuming than computing the base FID. However, *it allows us to fairly assess the quality of the recovered EOT solution, and we recommend this metric as the main for future EOT/SB studies.*

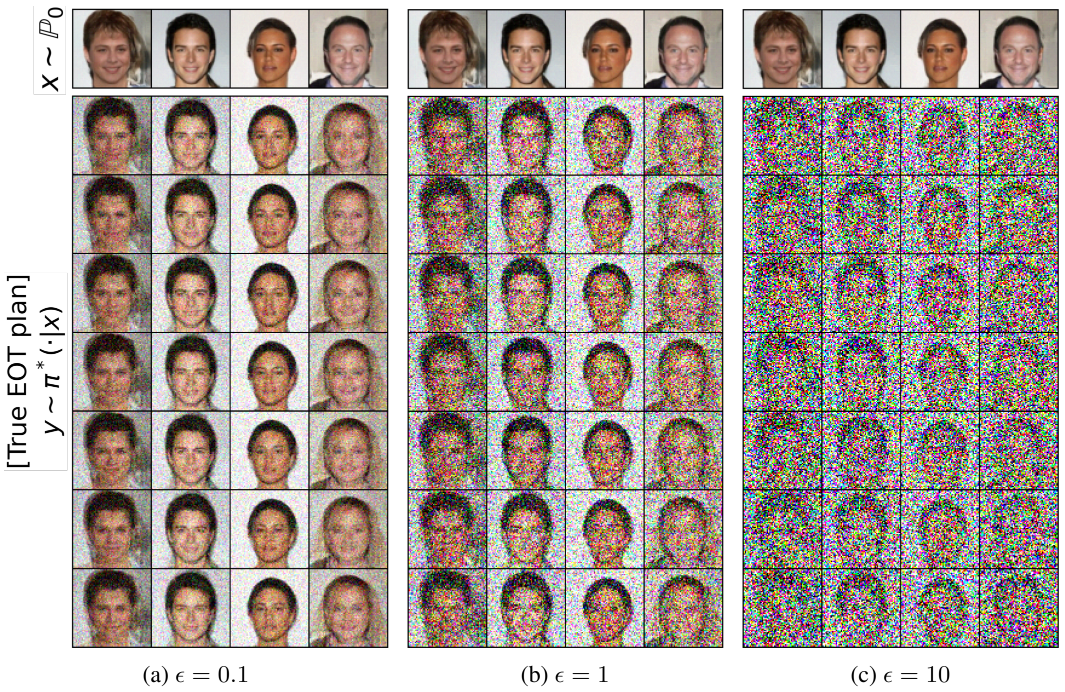
In Tables 6, 7, we present the evaluation results for $\lfloor$**ENOT**$\rceil$ [23]. We again emphasize that, to the best of our knowledge, there is no scalable *data→data* EOT/SB solver to compare against. Hence, we report the results as-is for future methods to be able to compare with them as the baseline.

Computational complexity. Sampling $x \sim \mathbb{P}_0$ is just applying the trained GLOW neural network to noise vectors $z \sim \mathcal{N}(\cdot|0, \sigma^2 I)$. Sampling $y \sim \mathbb{P}_1$ (or $y|x$) takes comparable time, as it is just extra sampling from the Gaussian mixture with x-dependent parameters (Proposition 3.3). In turn, as we noted above, sampling $x|y$ requires using the Langevin dynamic and takes considerable time. To obtain 3 **test** sets of 5K samples $y \sim \pi^*(\cdot|x)$ per each of 100 samples $x \sim \mathbb{P}_0$, we employed $8\times$A100 GPUs. This generation of test datasets took approximately 1 week.



(a) $\epsilon = 0.1$ (b) $\epsilon = 1$ (c) $\epsilon = 10$

Figure 7: Ground truth samples $x \sim \pi^*(\cdot|y)$ on images benchmark pairs.



(a) $\epsilon = 0.1$ (b) $\epsilon = 1$ (c) $\epsilon = 10$

Figure 8: Ground truth samples $y \sim \pi^*(\cdot|x)$ on images benchmark pairs.

ϵ	**0.1**	**1**	**10**
FID	5.99	3.21	4.9

Table 6: Test FID of $\lfloor$**ENOT**$\rceil$ on our images benchmark pairs.

ϵ	**0.1**	**1**	**10**
cFID	40.5	19.8	14.47

Table 7: Test conditional FID of $\lfloor$**ENOT**$\rceil$ on our images benchmark pairs

D Details of EOT/SB Solvers

D.1 Mixtures Benchmark Pairs

$\lfloor$**LSOT**$\rceil$ [49]. We use the part of the code of $\lfloor$**SCONES**$\rceil$ solver from the authors' repository

`https://github.com/mdnls/scones-synthetic/blob/main/cpat.py`

corresponding to learning dual OT potentials `blob/main/cpat.py` and the barycentric projection `blob/main/bproj.py` in the Gaussian case with configuration `blob/main/config.py`.

$\lfloor$**SCONES**$\rceil$ [14]. We use the aforementioned official code for training of dual OT potentials. We employ `sklearn.mixture.GaussianMixture` with 20 components to approximate the score of the target distribution. For the rest, we employ their configuration `blob/main/config.py` with batch size=1024 and the learning rate for Langevin sampling is $5 \cdot 10^{-4}$.

$\lfloor$**NOT**$\rceil$ This algorithm [35, Algorithm 1] is a generic algorithm for weak OT. It works for transport costs $C\big(x, \pi(\cdot|x)\big)$ which are straightforward to estimate by using samples of $\pi(\cdot|x)$. Entropic cost $C_{c,\epsilon}$ (6) does not fit this requirement, as it is not easy to estimate entropy from samples. To do it, one has to know the density of $\pi(\cdot|x)$. Thus, the authors of $\lfloor$**NOT**$\rceil$ skipped EOT setting. We fill this gap and do a minor modification to their algorithm. As the base implementation, we use

lr (potential and transport map)	T_{steps}	σ_z
$1e-4$	99	1.0

Table 8: $\lfloor$**NOT**$\rceil$ training parameters for the mixture benchmark pairs experiment.

`https://github.com/iamalexkorotin/NeuralOptimalTransport`

Instead of the multi-layer perceptron generator, we take a conditional normalizing flow with RealNVP architecture with context-dependent latent normal distribution. This enables the access to the density of $\pi(\cdot|x)$ and allows applying $\lfloor$**NOT**$\rceil$ algorithm to EOT. Our reimplementation is available at

`https://github.com/Penchekrak/FlowNOT`

Due to the decreased expressivity of RealNVP compared to MLP from $\lfloor$**NOT**$\rceil$, we do more optimization steps for the transport map before updating potential as well as larger parameter count compared to the original solver implementation for a similar task. We use the same set of hyperparameters across all experiments with different (ϵ, D). The hyperparameters are summarized in Table 8.

$\lfloor$**EgNOT**$\rceil$ [42] We use the official code for $\lfloor$**EgNOT**$\rceil$ from

`https://github.com/PetrMokrov/Energy-guided-Entropic-OT`.

For our mixture benchmark pairs experiment, we adapt the author's setup for the Gaussian-to-Gaussian experiment from their original paper [42, §5.2]. In particular, we use the same architectures of neural networks, see [42, Appendix C.2], but change the hyper-parameters of [42, Algorithm 1], since the original ones do not work properly when fitting Gaussian-to-Mixture. We hypothesize that the observed failure is due to the short-run nature of the energy-based training algorithm. We suppose that significantly increasing the number of Langevin steps K used at the training stage may leverage the problem. The specific hyper-parameters of $\lfloor$**EgNOT**$\rceil$ algorithm are the same for all (ϵ, D) pairs and provided in Table 9.

K	K_{test}	$\sqrt{\eta}$	σ_0	N
500	1000	0.05	1.0	1024

Table 9: $\lfloor$**EgNOT**$\rceil$ training parameters for the mixture benchmark pairs experiment.

We initialize the learning rate as $lr = 10^{-5}$ and decrease its value during the training. Similar to the original implementation of [42] we use a replay buffer but found that a high probability ($p = 0.95$) of samples reusage does not improve the quality and sometimes leads to unstable training. In turn, we choose $p = 0.5$. The reported numbers in Tables 4, 5 are gathered by launching the training process for approximately 50K iterations and reporting the best-obtained metric. We understand that such an evaluation procedure is not ideal and does not provide statistically significant results. However, the qualitative results reported in Table 2 seem to show the behaviour of $\lfloor$**EgNOT**$\rceil$ solver on our benchmark setup and reveal the key properties of the approach.

$\lfloor$**ENOT**$\rceil$ [23] We use the official code from

`https://github.com/ngushchin/EntropicNeuralOptimalTransport`

We use the same hyperparameters for this setup as the authors [23, Appendix E], except the number of discretization steps N, which we set to 200 as well as for other Schrödinger Bridge based methods. We also change the learning rate of the potential to $3 \cdot 10^{-4}$ for the setups with $\epsilon = 10$.

$\lfloor$**MLE-SB**$\rceil$ [52]. We tested the official code from

`https://github.com/franciscovargas/GP_Sinkhorn`

Instead of Gaussian processes, we used a neural network as for $\lfloor$**ENOT**$\rceil$. We use $N = 200$ discretization steps as for other SB solvers, 5000 IPF iterations, and 512 samples from distributions $\mathbb{P}_0$ and $\mathbb{P}_1$ in each of them. We use the Adam optimizer with $lr = 10^{-4}$ for optimization.

$\lfloor$**DiffSB**$\rceil$[15]. We utilize the official code from

`https://github.com/JTT94/diffusion_schrodinger_bridge`

with their configuration `blob/main/conf/dataset/2d.yaml` for toy problems. We increase the number of steps of dynamics to 200 and the number of steps of the IPF procedure for dimensions 16, 64 and 128 to 30, 40 and 60, respectively.

$\lfloor$**FB-SDE-J**$\rceil$[9]. We utilize the official code from

`https://github.com/ghliu/SB-FBSDE`

with their configuration `blob/main/configs/default_checkerboard_config.py` for the checkerboard-to-noise toy experiment, changing the number of steps of dynamics from 100 to 200 steps. Since their hyper-parameters are developed for their 2-dimensional experiments, we increase the number of iterations for dimensions 16, 64 and 128 to 15 000.

$\lfloor$**FB-SDE-A**$\rceil$ [9]. We also take the code from the same repository as above. We base our configuration on the authors' one (`blob/main/configs/default_moon_to_spiral_config.py`) for the moon-to-spiral experiment. As earlier, we increase the number of steps of dynamics up to 200. Also, we change the number of training epochs during one IPF procedure for dimensions 16, 64 and 128 to 2,4 and 8 correspondingly.

D.2 Images Benchmark Pairs

$\lfloor$**ENOT**$\rceil$ [23] As well as for the mixtures benchmark pairs, we use the official code from

`https://github.com/ngushchin/EntropicNeuralOptimalTransport`

We use the same hyperparameters for this setup as the authors [23, Appendix F] except the batch size which we set to 16 (`/blob/main/notebooks/Image_experiments.ipynb`).

E Additional Study of Hyperparameters of Solvers

To show that the default solvers parameters described in Appendix D are already a good choice, we additionally try different values of some of the most important hyperparameters. We consider each of the solvers except $\lfloor$**LSOT**$\rceil$ because it is anyway known to poorly perform due to the systematic bias in its solutions [31, 32]. For the evaluation, we consider the mixtures benchmark pair with $D = 64$ and $\epsilon = 1$ where most of the solvers perform reasonably well. In the tables below, we use "$*$" to mark the hyperparameters that we use for comparisons in §4.1.

For $\lfloor$**ENOT**$\rceil$ solver, we consider the number of inner and outer problem iterations during the optimization and present the results in Table 10. The obtained results show that the performance increases slowly with increasing number of iterations of both types.

Outer iters \ Inner iters	1	5	10	20
100	131.1	130.3	74.5	129.3
1000	28.77	47.36	25.91	20.16
10000	24.46	37.36	23.07$*$	**18.03**

Table 10: Comparison of cB$\mathbb{W}_2^2$-UVP $\downarrow$ (%) for $\lfloor$**ENOT**$\rceil$ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

For IPF-based SB solvers $\lfloor$**MLE-SB**$\rceil$, $\lfloor$**DiffSB**$\rceil$, $\lfloor$**FB-SDE-A**$\rceil$ and $\lfloor$**FB-SDE-J**$\rceil$, we try different numbers of IPF iterations and the number of samples used in each iteration. We present the results in Tables 11, 12, 13, 14. All of the IPF-based solvers learn an inversion of a diffusion process at each IPF step but they differ in the way how this is done. The typical number of IPF steps used by each algorithm is affected by this difference. The performance increases slowly with the increase of the two hyperparameters considered, at the cost of a proportional increase in iterations or in the number of samples used.

IPF iters \ Samples per iter	64	128	256	512
100	23.45	24.50	16.64	14.23
1000	16.95	15.35	10.71	8.74
5000	11.55	11.24	12.96	**8.41**$*$

Table 11: Comparison of cB$\mathbb{W}_2^2$-UVP $\downarrow$ (%) for $\lfloor$**MLE-SB**$\rceil$ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

IPF iters \ Samples per iter	64	256	512	1024
16	62.66	60.42	58.88	57.02
32	62.90	59.42	57.76$*$	55.08
64	62.84	59.46	57.78	**55.01**

Table 12: Comparison of cB$\mathbb{W}_2^2$-UVP $\downarrow$ (%) for $\lfloor$**DiffSB**$\rceil$ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

IPF iters \ Samples per iter	64	256	512
15000	173.16	163.04	160.5$*$
30000	168.86	165.06	**156.5**

Table 13: Comparison of cB$\mathbb{W}_2^2$-UVP $\downarrow$ (%) for $\lfloor$**FB-SDE-J**$\rceil$ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

IPF iters \ Samples per iter	64	256	512	1024
16	40.86	40.43	39.76	37.74
32	40.44	38.90	38.36$*$	35.46
64	40.00	38.86	38.31	**35.4**

Table 14: Comparison of cB$\mathbb{W}_2^2$-UVP $\downarrow$ (%) for $\lfloor$**FB-SDE-A**$\rceil$ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

For $\lfloor$**SCONES**$\rceil$ and $\lfloor$**EgNOT**$\rceil$ solvers, we consider the number of Langevin steps and the Langevin step size and present the results in Table 15 and Table 16. For $\lfloor$**SCONES**$\rceil$ the results obtained show that the performance increases slowly with increasing Langevin steps and decreasing Langevin step size. For $\lfloor$**EgNOT**$\rceil$ the trends are slightly different, since the optimal Langevin step size seems to be in the interval $[0.1, 0.2]$. Anyway, our selected parameters are reasonable ones because specifying an enormously large number of Langevin steps for these solvers is sort of impractical.

Finally, for $\lfloor$**NOT**$\rceil$ we consider the number of inner problem steps and the hidden size of the used neural network (conditional normalizing flow). We present results in Table 17.

Langevin step size \ Langevin steps	64	256	512	1024
10^{-4}	92.35	89.17	86.48	**86.33**∗
10^{-3}	93.51	90.41	88.22	87.74

Table 15: Comparison of cB$\mathbb{W}_2^2$-UVP ↓ (%) for ⌊**SCONES**⌉ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

Langevin step size \ Langevin steps	100	200	500	1000
0.01	70.9	70.98	72.9	68.13
0.02	71.31	67.14	69.11	69.02
0.05	68.78	68.59	63.73∗	56.84
0.1	64.52	57.45	52.35	51.9
0.2	58.22	60.08	58.93	**41.31**

Table 16: Comparison of cB$\mathbb{W}_2^2$-UVP ↓ (%) for ⌊**EgNOT**⌉ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

Inner steps \ Hidden size	64	128	192	256	320	384	448	512
1	93.14	167.05	149.52	189.0	89.1	161.66	176.43	175.67
5	82.64	86.09	82.18	190.04	147.31	105.46	103.5	150.76
10	163.47	146.68	53.26	137.47	100.84	171.65	115.84	126.96
100	18.68	21.4	**14.64**	18.08	16.66	20.64∗	18.71	15.15
200	61.99	52.74	58.63	53.89	52.44	55.3	55.02	54.75

Table 17: Comparison of cB$\mathbb{W}_2^2$-UVP ↓ (%) for ⌊**NOT**⌉ on mixtures benchmark pairs for $D = 64$, $\epsilon = 1$ and different hyperparameters.

Discussion. From the results it can be seen that for the most solvers' dependence on the considered hyperparameters is almost monotonic and the hyperparameters chosen for the solver comparison on the mixtures setup are in the region where the metric growth is almost saturated.

F Qualitative Evaluation of the Drift Learned with SB methods

Our benchmark primarily aimed at quantifying the recovered **conditional EOT plan** $\widehat{\pi}(\cdot|x)$. Thanks to our Proposition 3.5, our benchmark provides not only the ground truth conditional EOT plan $\pi^*(\cdot|x)$, but the **optimal SB drift** $v^*(x,t)$ as well. This means that for SB solvers we may additionally compare their recovered SB drift $\widehat{v}$ with the ground truth drift v^*. Here we do this for ⌊**MLE-SB**⌉, ⌊**DiffSB**⌉, ⌊**ENOT**⌉, ⌊**FB-SDE-A**⌉, ⌊**FB-SDE-J**⌉ solvers by using our mixtures pairs.

METRICS. Recall that T_{v^*} is the Schrödinger bridge (10) and let $T_{\widehat{v}}$ denote the learned process:

$$dX_t = \widehat{v}(x,t)dt + \sqrt{\epsilon}dW_t, \qquad X_0 \sim \mathbb{P}_0.$$

Both T_{v^*} and $T_{\widehat{v}}$ are diffusion processes which start at distribution $\mathbb{P}_0$ at $t = 0$ and have fixed volatility ϵ. Their respective drifts are v^* and $\widehat{v}$. For each time $t \in [0,1]$, consider

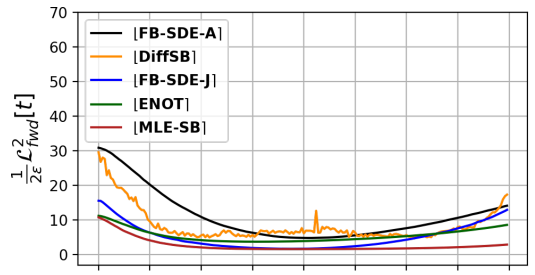
$$\mathcal{L}^2_{\text{fwd}}[t] \overset{\text{def}}{=} \mathbb{E}_{T_{v^*}} \|v^*(X_t,t) - \widehat{v}(X_t,t)\|^2, \tag{31}$$

$$\mathcal{L}^2_{\text{rev}}[t] \overset{\text{def}}{=} \mathbb{E}_{T_{\widehat{v}}} \|v^*(X_t,t) - \widehat{v}(X_t,t)\|^2. \tag{32}$$

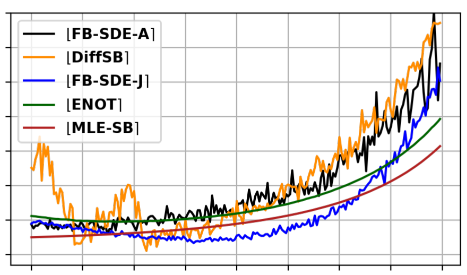
which are the expected squared differences between the ground truth v^* and learned $\widehat{v}$ drifts at the time t. In (31), the expectation is w.r.t. X_t coming from the true SB trajectories of T_{v^*}, while in (32) – w.r.t. the learned trajectories from $T_{\widehat{v}}$. Reporting this metric for all the time steps, all the mixtures pairs and solvers would be an overkill. In what follows, we use this metric for quantitative analysis.

First, for $D = 16$ and $\epsilon \in \{0.1, 10\}$, we plot these metrics (as a function of time t). The results for all the solvers are shown in Figure 9. **Second**, we provide Table 19 where for $D \in \{2, 16, 64, 128\}$ and $\epsilon \in \{0.1, 10\}$ report $\mathcal{L}^2$ metrics averaged over $t \in [0,1]$. Namely, we report

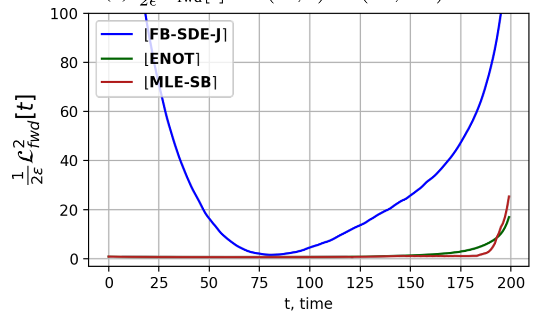
$$\text{KL}\left(T_{v^*} \| T_{\widehat{v}}\right) \overset{\text{def}}{=} \frac{1}{2\epsilon}\int_0^1 \mathcal{L}^2_{\text{fwd}}[t]dt \qquad \text{and} \qquad \text{RKL}\left(T_{v^*} \| T_{\widehat{v}}\right) \overset{\text{def}}{=} \frac{1}{2\epsilon}\int_0^1 \mathcal{L}^2_{\text{rev}}[t]dt. \tag{33}$$



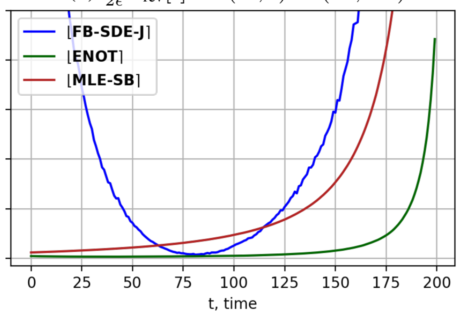
(a) $\frac{1}{2\epsilon}\mathcal{L}^2_{\text{fwd}}[t]$ for $(D, \epsilon) = (16, 0.1)$



(b) $\frac{1}{2\epsilon}\mathcal{L}^2_{\text{rev}}[t]$ for $(D, \epsilon) = (16, 0.1)$



(c) $\frac{1}{2\epsilon}\mathcal{L}^2_{\text{fwd}}[t]$ for $(D, \epsilon) = (16, 1)$



(d) $\frac{1}{2\epsilon}\mathcal{L}^2_{\text{rev}}[t]$ for $(D, \epsilon) = (16, 1)$

Figure 9: $\mathcal{L}^2$ metrics between the ground truth drift v^* and the drift $\widehat{v}$ learned by SB solvers.

We write "KL" and "RKL" not by an accident. Thanks to the well-celebrated Girsanov's theorem, these are indeed the **forward and reverse KL divergences** between processes T_{v^*} and $T_{\widehat{v}}$.

In all the SB solvers, we consider 200 time discretization steps $t = \{\frac{1}{200}, \frac{2}{200}, \dots 1\}$ for their training. During testing, we evaluate $\mathcal{L}^2$ metrics (31) and (32) on the same time steps. To estimate (31) and (32), we use 10^5 samples X_t which are taken from random trajectories of processes T_{v^*} and $T_{\widehat{v}}$. These trajectories are simulated via the standard Euler–Maruyama method.

	$\epsilon=0.1$				$\epsilon=1$			
	$D=2$	$D=16$	$D=64$	$D=128$	$D=2$	$D=16$	$D=64$	$D=128$
⌊**ENOT**⌉	0.61	5.49	6.59	10.36	0.86	1.64	11.43	37.53
⌊**DiffSB**⌉	6.96	12.89	-	-	12.28	>1000	>1000	>1000
⌊**FB-SDE-A**⌉	6.9	11.08	-	-	10.59	>1000	>1000	>1000
⌊**FB-SDE-J**⌉	3.02	5.02	9.60	28.85	18.79	44.79	629.28	>1000
⌊**MLE-SB**⌉	0.62	2.63	4.76	7.86	0.96	1.86	9.66	34.95

Table 18: Forward KL between the ground truth SB process T_{v^*} and the process $T_{\widehat{v}}$ learned with SB solvers on our mixtures benchmark pairs.

	$\epsilon=0.1$				$\epsilon=1$			
	$D=2$	$D=16$	$D=64$	$D=128$	$D=2$	$D=16$	$D=64$	$D=128$
⌊**ENOT**⌉	72.86	78.98	135.29	221.26	18.40	49.65	177.02	348.05
⌊**DiffSB**⌉	11.85	21.16	-	-	121.43	>1000	>1000	>1000
⌊**FB-SDE-A**⌉	12.29	19.40	-	-	100.22	>1000	>1000	>1000
⌊**FB-SDE-J**⌉	8.03	12.11	17.16	49.32	64.37	123.68	>1000	>1000
⌊**MLE-SB**⌉	18.03	28.24	163.34	254.16	22.80	86.07	296.97	636.27

Table 19: Reverse KL between the ground truth SB process T_{v^*} and the process $T_{\widehat{v}}$ learned with SB solvers on our mixtures benchmark pairs.

DISCUSSION. Interestingly, we see that the forward KL divergence shows a smoother behaviour than the RKL for almost all SB solvers. According to our evaluation, the ⌊**MLE-SB**⌉ and ⌊**ENOT**⌉ solvers mostly beat every other solver in the forward KL metric. At the same time, the RKL metric of ⌊**ENOT**⌉ is surprisingly the worst. While we make all these observations, we do not know how to explain them. We hope that the question of the interpretation of the KL and RKL values will be addressed in future SB studies.

G Potential Societal Impact

Our proposed approach deals with generative models based on Entropic Optimal Transport and Schrödinger Bridge principles. Such models form and emergent subarea in the field of machine learning research and could be used for various purposes in the industry including image manipulation, artificial content rendering, graphical design, etc. Our benchmark is a step towards improving the reliability, robustness and transparency of these models. One potential negative of our work is that improving generative models may lead to transforming some jobs in the industry.

H Building Benchmarks from Real Data

In this section, we present a simple heuristic recipe to build benchmark pairs similar to some given real-world data. To illustrate the recipe, we consider toy 2D data example and several single-cell datasets [36, 8]. Code and data for the experiments in this section can be found in the `benchmark_construction_examplesdata` folder of our repository.

H.1 Recipe for Building Benchmark Pairs form Data.

For constructing distribution pairs similar to some given data, we consider a pair of original and target datasets obtained from the true distributions $\mathbb{P}_0$ and $\mathbb{P}_1$, respectively. We heuristically initialize the LSE potential (15) $f^*(y) = \epsilon \log \sum_{n=1}^{N} w_n \mathcal{Q}(y|b_n, \epsilon^{-1}A_n)$ with b_n as cluster centers obtained from the K-means clustering algorithm applied to the target data from $\mathbb{P}_1$. The weights w_n are chosen to be $1/N$ and matrices $A_n = \lambda I$ are diagonal where λ is a manually-chosen parameter (shared between all A_n). For any x the conditional plan $\pi^*(\cdot|x)$ for LSE potential f^* is just a Gaussian mixture and the mean of each its component is largely determined by b_n (Proposition 3.3). We empirically found that the resulting constructed distribution $d\widehat{\mathbb{P}}_1(y) = d\pi_1^*(y) = \int d\pi^*(y|x)d\mathbb{P}_0(x)$ from $\mathbb{P}_0$ resembles the Gaussian mixture approximation of the target dataset if one managed to find proper value of λ.

In the rest of this section, we use the described recipe to construct benchmark pairs from data to show that the LSE parameterization of the potential provides a wide class of EOT/SB solutions and even allows constructing a benchmark similar to real data.

H.2 Benchmark Pairs for 2D data.

Code and data for the experiment described in this section can be found in the folder `benchmark_construction_examples/2d_data` of our repository.

To begin with, we present the results of constructing a benchmark pair from 2D data. We consider a Gaussian distribution $\mathbb{P}_0$ as the source distribution and two moons $\mathbb{P}_1$ as the target distribution. We aim to use the previously described recipe §H.1 to find parameters of the LSE potential to construct an EOT solution between $\mathbb{P}_0$ and an approximation of $\mathbb{P}_1$ denoted as $\widehat{\mathbb{P}}_1$. Here we consider EOT with $\epsilon = 0.05$, use $N = 100$ for LSE potentials, and choose $\lambda = 50$. The result is in Figure 10.

As seen from the figure, the constructed target benchmark distribution $\widehat{\mathbb{P}}_1$ is similar to the target distribution $\mathbb{P}_1$. In turn, the EOT plan maps $x \sim \mathbb{P}_0$ to the close regions of the target distribution.

H.3 Single-cell RNA Data

Code and data for the experiment described in this section can be found in the folder `benchmark_construction_examples/single_cell_rna` of our repository.

We consider the same setup as in [36, §5.2]. We use their data from the supplementary materials.[2] The provided data displays the progression of human embryonic stem cells as they differentiate from embryoid bodies into a range of cell types, such as mesoderm, endoderm, neuroectoderm, and neural

[2] `https://openreview.net/forum?id=d3QNWD_pcFv`

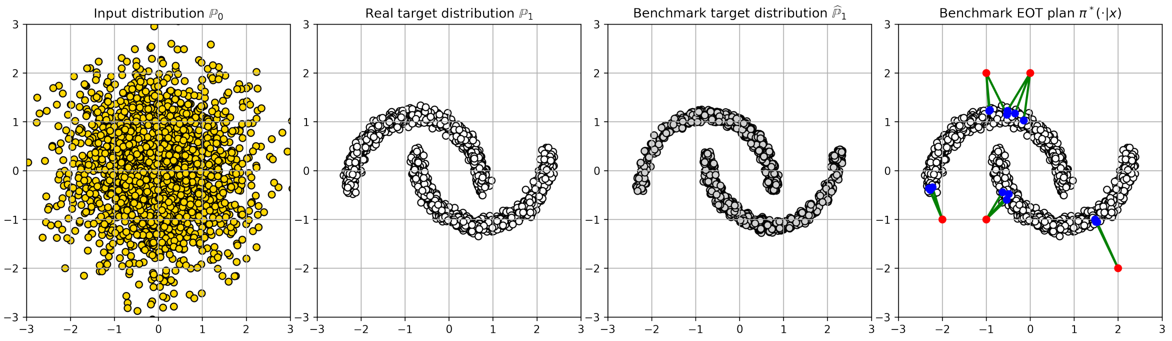


Figure 10: *Gaussian* $\to$ *Two Moons* benchmark pair.

crest, throughout a span of 27 days. The cell samples (approximately 2000 ones per each time period) were gathered at five distinct intervals (t_0: day 0 to 3, t_1: day 6 to 9, t_2: day 12 to 15, t_3: day 18 to 21, t_4: day 24 to 27). These collected cells were evaluated via scRNAseq, subjected to quality control filtering, and then projected onto a 5-dimensional feature space utilizing principal component analysis (PCA).

To construct the benchmark pair using the LSE potential, we consider $N = 250$, $\epsilon = 100$ and $\lambda = 100$ and employ the train data at times t_0 and t_4. Then we use the constructed benchmark plan $\pi^*(\cdot|x)$ to map source data at time t_0 to the data at time t_4 and obtain benchmark target distribution samples $\widehat{\mathbb{P}}_1$. Finally, we fit TSNE [51] to the combined dataset of samples from $\mathbb{P}_1$ and $\widehat{\mathbb{P}}_1$ and then plot their projections in Figure 11. The resulting plots are very similar, confirming that the constructed benchmark target data resembles the considered single-cell target data.

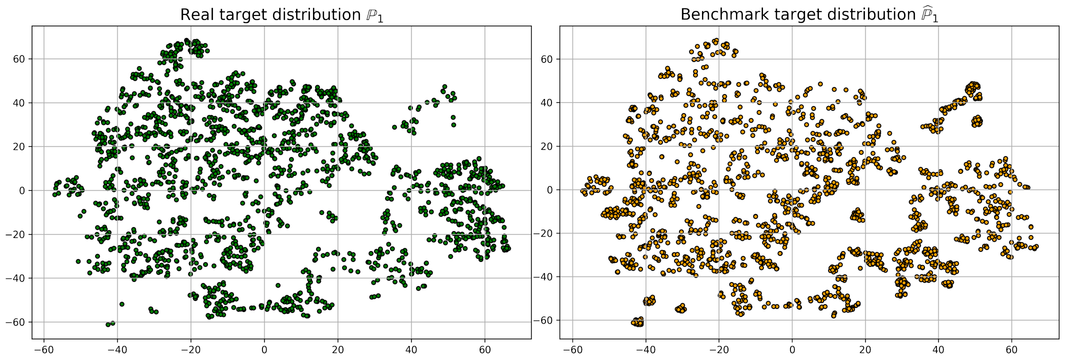


Figure 11: TSNE visualization of Single-cell RNA target data and our constructed target data.

H.4 Single-cell Drugs Data

Code and data for the experiment described in this section can be found in the folder `benchmark_construction_examples/single_cell_drugs` of our repository.

Method	scGen	cAE	CellOT [8]	EOT Benchmark (ours)
MMD↓	0.0241	0.0074	0.0013	0.0036

Table 20: MMD↓ distances (on the test data) between the observed perturbed cells $\mathbb{P}_1$ and predicted responses from control cells $\widehat{\mathbb{P}}_1$.

In [8], the authors consider the problem of predicting single-cell drug responses for drugs with different molecular effects, using melanoma cell lines profiled by 4i technology (single-cell technology). Utilizing a blend of two melanoma tumor cell lines at a 1:1 ratio, a total of 21,650 cells were imaged. Within this dataset, 11,526 cells existed in the untreated control state, 2,364 received Erlotinib treatment, 2,650 underwent Imatinib treatment, 2,683 were subjected to Trametinib treatment, and

2,417 were treated with a combination of Trametinib and Erlotinib. After preprocessing, each cell is described by 78 features. The train-test split with each drug is 80:20.

In this example, we consider cell data before treatment $(\mathbb{P}_0)$ and after treatment with Erlotninib $(\mathbb{P}_1)$. For the construction of the benchmark pair using an LSE potential, we consider $N = 250$, $\epsilon = 1$ and $\lambda = 20$. As with the single cell RNA data §H.3, we fit the TSNE [51] on a combined dataset of samples from $\mathbb{P}_1$ and $\widehat{\mathbb{P}}_1$ and then plot their projections in Figure 11. As seen from the visualizations, the TSNE projections of the real data and the mapped data are similar.

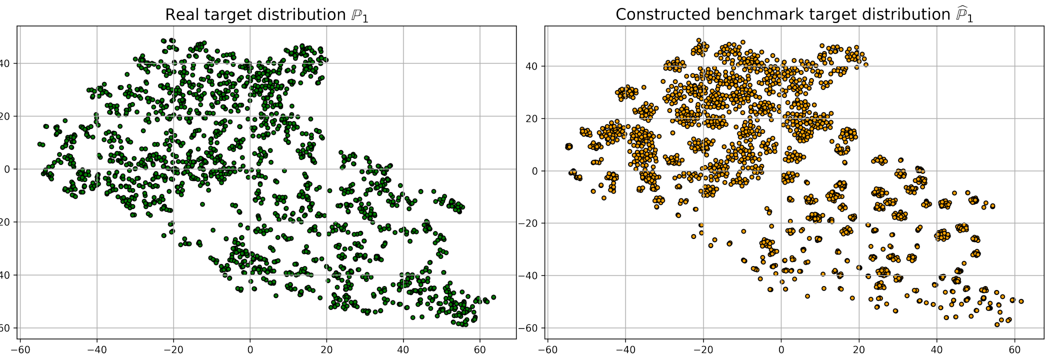


Figure 12: TSNE visualization of Single-cell Drugs target data and our constructed target data.

In addition, we quantitatively evaluate on the test data how well the constructed target distribution $\widehat{P}_1$ matches the true data distribution $\mathbb{P}_1$. We employ the same MMD metric as the authors and present the results in Table 20. The data for the baselines scGen, cAE and the authors' method CellOT are taken from [8]. As one can see, our approach is even better than two of the baselines considered.