1 Synthetic dataset setup and further experiments

In order to measure the ability of our method metaCDE to learn multimodality and heteroscedasticity in the response variable, we construct the datasets as follows: we sample $y_i \sim \text{Uniform}(0, 1)$ and set $x_i = \cos(ay_i + b) + \epsilon_i$, where a and b vary between tasks, with noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Here we sample a from $U(8, 12)$ and phase vary from $U(0, \pi)$. Intuitively, this just corresponds to rotated sine curves as can be seen in the examples in Section 5 of the Appendix. (see most left column for best density estimate of the true density.)

In this experiment we are given a variable number of context points during testing time ranging from $15, 30$ and 50. This is done to investigate the ability of metaCDE to adapt even in small dataset situations.

MetaCDE/Neural Process/MetaNN are trained with $15, 30$ and 50 context points on the tasks and with 80 target points during training. At testing time, we simply pass the data through our model without having to retrain on the new unseen dataset. Note that we report again the p-values of the Wilcoxon signed one-sided test and we can see that as we decrease the context points, our methods is significantly outperforming the other methods. See Section 5 of Appendix for additional

Each of the non meta learning models DDE, KCEF, ϵ-KDE, LSCDE are trained separately on each new dataset as they cannot share information between tasks.

1.1 Model specifications

For our MetaCDE we used a 3-hidden layer Neural Network with *tanh* activation functions and *Adam* optimizer for all of our feature maps ϕ_x, ϕ_y, b_θ. We cross-validate on held out dataset, over 32 and 64 hidden nodes per layer and $\lambda = 1.0, 0.1, 0.01$ for the regularization parameter. We fixed the learning rate at 1e-3. We also set $\kappa = 10$ as suggested by (2).

- KCEF: we used the CV function that was in built in their Github repository (optimizing the parameters from a range [1e-1, 10] `https://github.com/MichaelArbel/KCEF`) as well as consulted the authors to make sure we are using their method correctly.
- LSCDE: We CV for σ in logspace$(-3, 5, 20)$ and λ in logspace$(-5, 5, 20)$
- ϵ-KDE: We CV over ϵ in linspace$(0.1, 1, 15)$ and bandwidth in linspace$(0.01, 1, 15)$
- DDE: We CV over the bandwidth of 0.5 and 1.0

NOTE: We also tried to use a standard RBF kernel to compute our CMEO and CME with 50 context points on the synthetic data, however, these results were not up to par with what we got with NN. We searched for lengthscales over a range 0.01 to 10, however, the results were not comparable on synth data (84.65 +- 21.15), see Table in section 5.1 for comparison.

The reason for this is first of all, the stationarity of the Gaussian kernel and secondly, only having very little data to construct a good CMEO. Using NN and CMEs we were able to capture the density by training it across tasks and hence learning more efficient embeddings than a gaussian kernel could. In addition, deep kernels have recently shown impressive results (3) and hence using a NN as a feature map is well justified.

2 Neural Network version of our method (MetaNN)

Furthermore, in order to investigate the importance of the CMEO task representation, we developed a purely neural version of our proposed method, which we call MetaNN. It differs from MetaCDE solely in the task representation. While MetaCDE uses kernel embeddings formalism to represents the task using CMEO calculated on the context points, MetaNN uses the DeepSets (4) approach, where the context pairs (x_i, y_i) are simply concatenated into a vector, and then passed through a neural network. The outputs are then averaged to obtain the task embedding to which any new x_{target} is concatenated to obtain the "neural" equivalent to CME. This neural representation is then being pushed through a Feed forward network in order to have the same dimension as $\phi_y(y_{target})$. This ensures that we can take the inner product to compute s_θ.

The same training procedure as MetaCDE follows. We note that the concatenation of x and y encodes the joint distribution, rather than the conditional as in MetaCDE. While such task representation does preserve the relevant information, it is susceptible to changes in the marginal of x across tasks and conditional representations are intuitively better suited for the task of conditional density estimation. The experiments demonstrate the value of combining the CME formalism with neural representations and we obtain significantly better results with MetaCDE compared to MetaNN. In our experiments we simply used a 3 layer MLP and cross-validate over either 32, 64 hidden nodes and learning rates of 1e-3 or 1e-4.

2.1 Comparison of MetaNN to MetaCDE

Next we will compare the MetaNN and MetaCDE algorithms in order to investigate the importance of the

conditional mean embedding operator. As mentioned above, we have now swapped out the computation of the CMEO for a NN for MeatNN. This representation is then concatenated with the new x_{target} to give us an element in $\mathcal{H}_Y$. We test out MetaNN on the synthetic dataset described in the experimental section.

MetaNN will perform worse on 50 context points as show in the main text but better on 15, where MetaNN achieves a log-likelihood of 114.43 ± 26.44, whereas MetaCDE achieved only 51.73 ± 10.43 (see Figure 1). This can be explained by two factors.

Firstly, a lower number of context points might give us a worse estimate of the conditional mean embedding operator. Secondly, we note that it takes significantly more task examples for the MetaNN to achieve the performance and hence this might have been due to the limited variety in the training task *i.e.* variation in the range of the period and phase parameter. Hence we conjectured that MetaNN might have just memorized the tasks well.

Therefore we have ran additional experiments to on a harder synthetic dataset where we now sample a from $U(4, 14)$ and phase vary from $U(-\pi, \pi)$ (i.e. more variety in the tasks than before where we sampled a from $U(8, 12)$ and phase from $U(0, \pi)$). In this case, MetaNN seems to completely fail and not able to learn anything useful at all. As the tasks in this case are more variable (see figure 2). Hence, we have not included MetaNN in the below figures when comparing with other conditional density methods.

To further investigate this phenomenon, we have created a new task based on samples on Gaussian Processes (GPs). Here we sample 2 GPs with an Gaussian kernel with lengthscale 1 as well as a uniform random variable from $q \sim U(1, 3)$. We then added u to one of the sampled GPs and hence created a multimodal dataset in y (see figure 3). This task has a lot more variability than the previous synthetic dataset task. Below, we illustrate how MetaCDE is still able to perform well whereas MetaNN completely fails to learn anything useful. This illustrates that CMEO includes useful additional inductive biases in our model. In particular, by using a CMEO, we explicitly tell the model which entries are covariates and which are responses and that the relevant property of the data for this task is the conditional distribution.

3 Normalization network b_θ

First of all, we want to note that learning the b_θ is not a novel idea but in fact has been proposed by the seminal paper on noise contrastive estimation (2). (2) note that one can actually learn the normalization constant and show empirically that that the learnt density is actually close to a properly normalized density. Intuitively, the reason this happens is as follows.

$$P_\theta(\text{True}|y, x) := \frac{p_\theta(y|x)}{p_\theta(y|x) + \kappa p_f(y)}. \tag{1}$$

Assuming that we have a Bayes optimal classifier above and that p_f is normalized (we choose this distribution so we know it is normalized), then assuming $p_\theta(y|x)$ was not normalized by a factor of γ then we could just divide numerator and denominator by γ. This would hence just correspond to modifying κ to κ/γ. However the Bayes optimal classifier sees exactly κ more fake samples then real ones and hence in the case Bayes optimality was reached, we should have γ close to 1, i.e. normalized $p_\theta(y|x)$. We even show in experiments that the normalziation needed is minimal (see Figure 5). Additionally, we also noted that using b_θ actually helped in terms of stability in our training as it gives an extra degree of flexibility.

Next, we note that $b_\theta(x)$ depends on $s_\theta(x, y)$ and hence is task/context set dependent. Modelling $b_\theta(x)$ as a neural network that only takes input x would not be task dependent, as it would be the same for each task and only depend on x. Therefore, we set the input of b_θ to be the CME $\widehat{\mu}_{Y=y|X=x}$, which thus incorporates all task information compactly (as $\widehat{\mu}_{Y=y|X=x}$ is computed using the CMEO). We have done further studies of this normalization network and show how it effectively normalizes the density approximately.

To ensure a fair comparison between methods, we post-normalize the densities for all our experiments, i.e. whenever we compute a conditional density, $p(y^*|X = x)$, we first create a grid, $\{y_i\}_{i=1}^{100}$, and consider equally spaced evaluations $p(y_i|X = x)$ over the range of the data, and use them to re-normalize the density model before evaluating performance by computing the log-likelihood at y^*.

We note that most methods are already producing density models that are very close to being normalized so the effect of the post-normalization is minimal.

Fig. 5 illustrates the post-normalization in our GP experiments (i.e. the closer the line is to one the less we need to normalize our density), demonstrating that our approaches are able to learn an approximately normalized density. We however still perform the post normalization in order to remain fair compared to other methods. Another thing to note is that including the information of the CME $b_\theta(x)$ compared to not using it (no CME), allows us to have even less normalization necessary when learning the density. We have also found that the normalization network helps greatly in keeping the learned density approximately normalized.

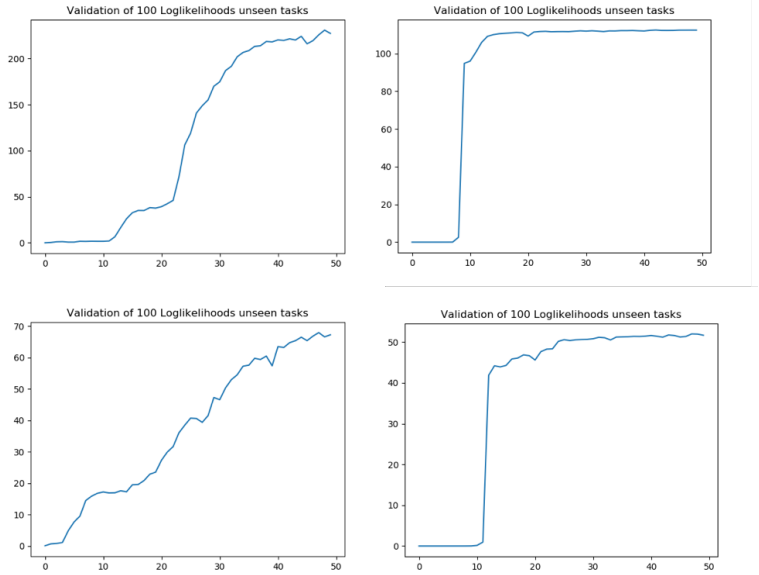

Figure 1: Figure illustrating how MetaNN performs better in low context points settings but seems to learn slower than MetaCDE. Top Row: 30 context points (left) MetaNN (right) MetaCDE; Bottom row: 15 context points (left) MetaNN (right) MetaCDE (x-axis represents 1 unit=10k tasks, y-axis loglikelihood)

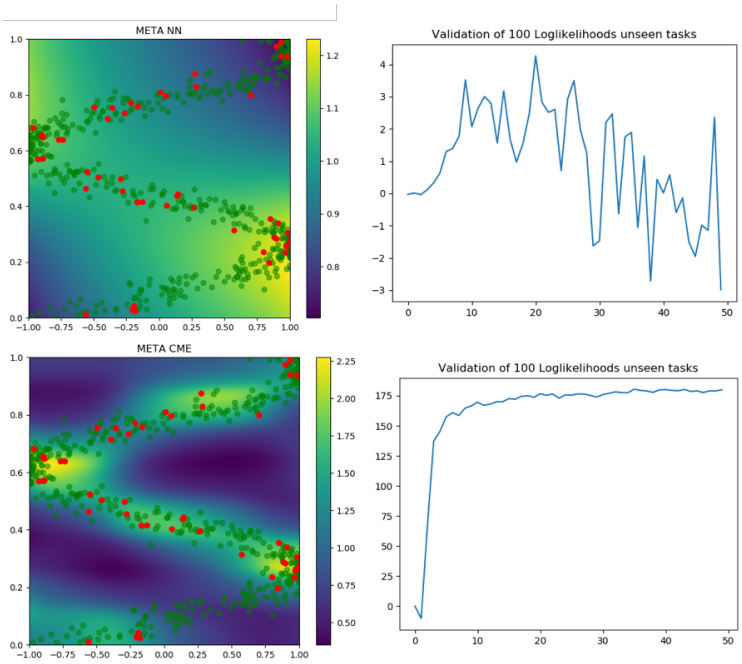

Figure 2: Figure illustrating how MetaNN fails when task become more variable. Top Row: MetaNN; Bottom row: MetaCDE (x-axis represents 1 unit=10k tasks, y-axis loglikelihood)

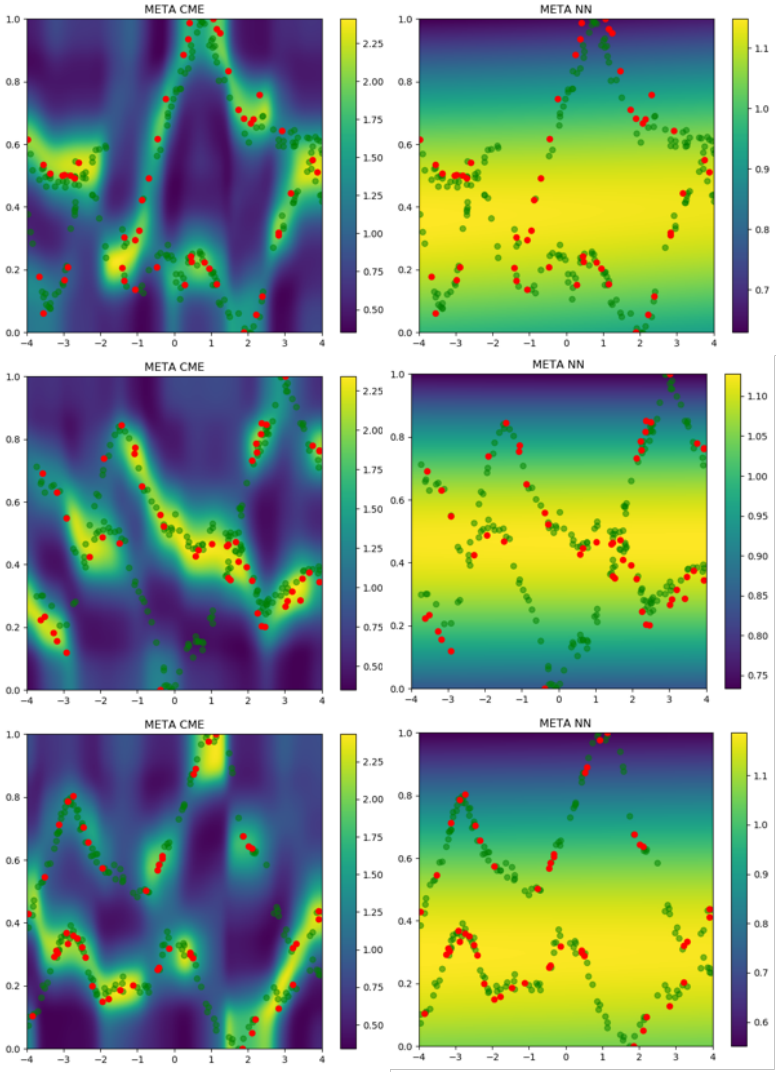

Figure 3: Density maps of the GP example (Left)MetaCDE (Right)(MetaNN)

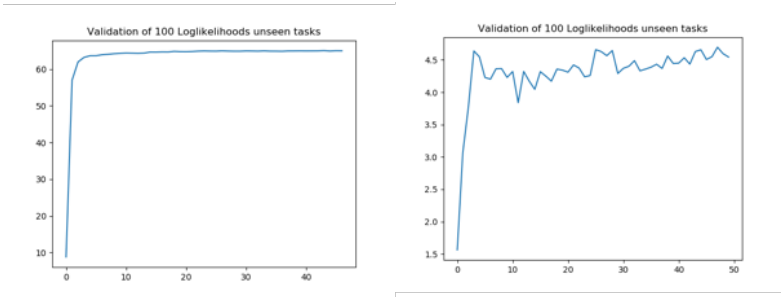

Figure 4: Evolution of the log-likelihood (x-axis represents 1 unit=10k tasks, y-axis loglikelihood) (Left)MetaCDE (Right)(MetaNN)

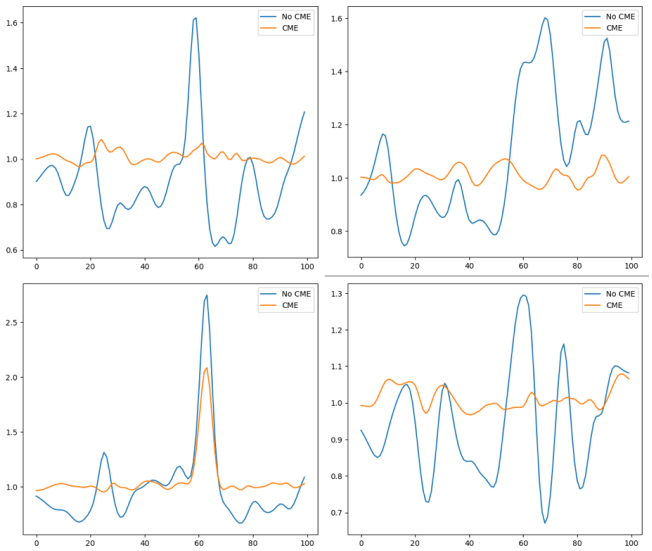

Figure 5: Post normalization needed after learning the density. Comparison shows the regimes where b_θ network includes CME as an input and the one where it does not.

4 Choice of the Fake Distribution in NCE

The choice of the fake distribution plays a key role in the learning process here, especially because we are interested in conditional densities. In particular, if the fake density is significantly different from the marginal density $p(y)$, then our model could learn to distinguish between the fake and true samples of y simply by constructing a "good enough" model of the marginal density $p(y)$ on a given task while completely ignoring the dependence on x (this will then lead to feature maps that are constant in x).

This becomes obvious if, say, the supports of the fake and the true marginal distribution are disjoint, where clearly no information about x is needed to build a classifier – i.e. the classification problem is "too easy".

Thus, ideally we wish to draw fake samples from the true marginal $p(y)$ in a given task. While we could achieve this by drawing a y paired to another x, i.e. from the empirical distribution of pooled ys in a given task, recall that we also require to be able to compute the fake density pointwise. Hence, we propose to use a kernel density estimate (KDE) of y's as our fake density in any given task.

In particular, a kernel density estimator of $p(y)$ is computed on all responses y (context and target). To sample from the this fake density, we draw from the empirical distribution of pooled y's and add Gaussian noise with standard deviation being the bandwidth of the KDE (we are using a Gaussian KDE for simplicity here; other choices of kernels are of course possible with appropriate modification of the type of noise). As our experiments demonstrate, this choice ensures that the fake samples are sufficiently hard to distinguish from the true ones, requiring the model to learn meaningful feature maps which capture the dependence between x and y and are informative for the CDE task.

Furthermore, we want to note that we chose κ i.e. number of fakes samples to be 10, mainly because in the original noise contrastive paper, they use also used 10 and we noticed in our experiments that even when increasing κ the performance didn't increase by much and hence just fixed $\kappa = 10$.

Finally, we note that in principle it is possible to consider families of fake distributions which depend on the conditioning variable x. We do not explore this direction here and will leave this for future work.

5 Illustration of Synthetic dataset

In this section we illustrate that our proposed method, metaCDE, performs well even when data becomes scarce. Hence we applied our method on several tasks, where we have 50, 30 and 15 datapoints as context. In each of the experiments we also plotted the corresponding density, the method gave us and we can clearly see that only metaCDE is able to recover the "rotated sine curve" (synthetic data experiments) in all 3 cases, which is also shown in the loglikelihood estimates.

5.1 Using 50 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**197.84 ± 22.45**	-81.11±18.53	162.98 ± 69.01	44.95 ± 74.36	-388.30 ± 703.17	116.31 ± 236.99
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	8.144e-07	<2.2e-16	< 2.2e-16	2.384e-07

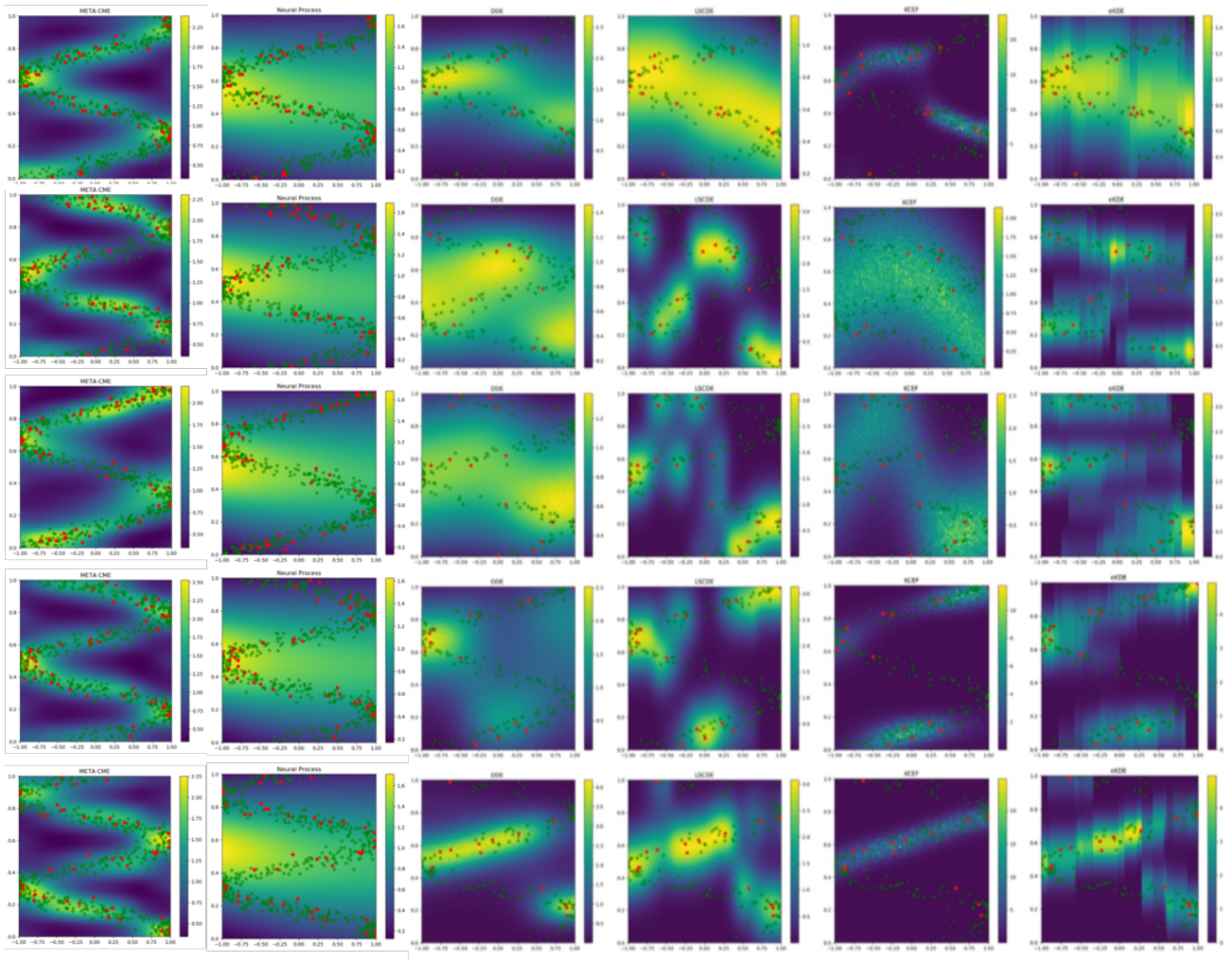

Figure 6: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

5.2 Using 30 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**113.27 ± 17.36**	-48.98±12.26	64.61 ± 54.33	-23.02 ± 65.31	-233.38 ± 528.99	29.64 ± 195.49
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	4.577e-14	<2.2e-16	< 2.2e-16	4.917e-13

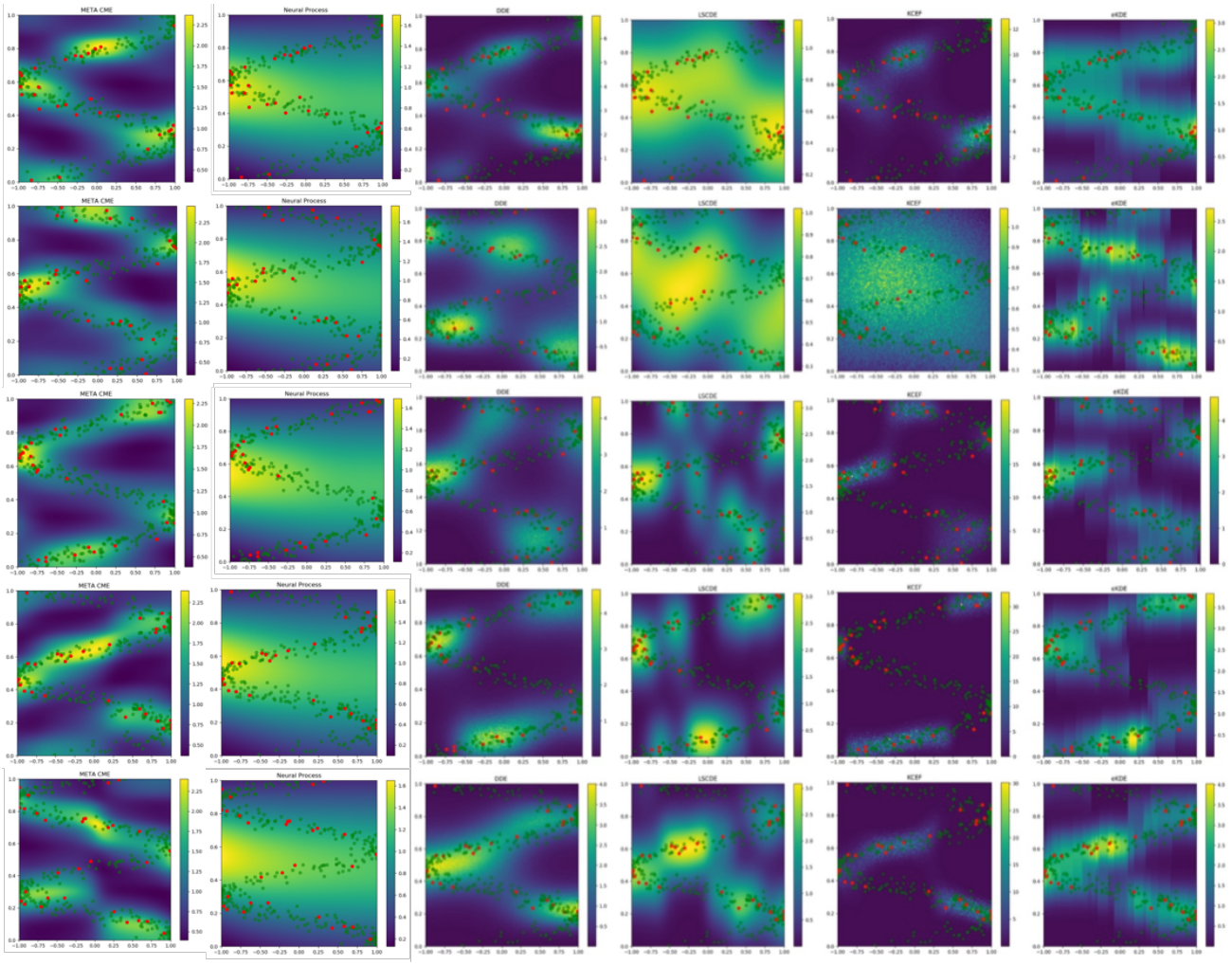

Figure 7: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

5.3 Using 15 context points

	MetaCDE	NP	DDE	LSCDE	KCEF	ϵ-KDE
Sythn. Data Mean over 100 log-likelihoods	**51.73 ± 10.48**	-24.39±8.20	0.58 ± 40.70	-57.99 ± 59.13	-142.19 ± 259.59	-87.50 ± 224.13
Sythn. Data P-value for Wilcoxon test	NA	< 2.2e-16	4.577e-14	<2.2e-16	< 2.2e-16	< 2.2e-16

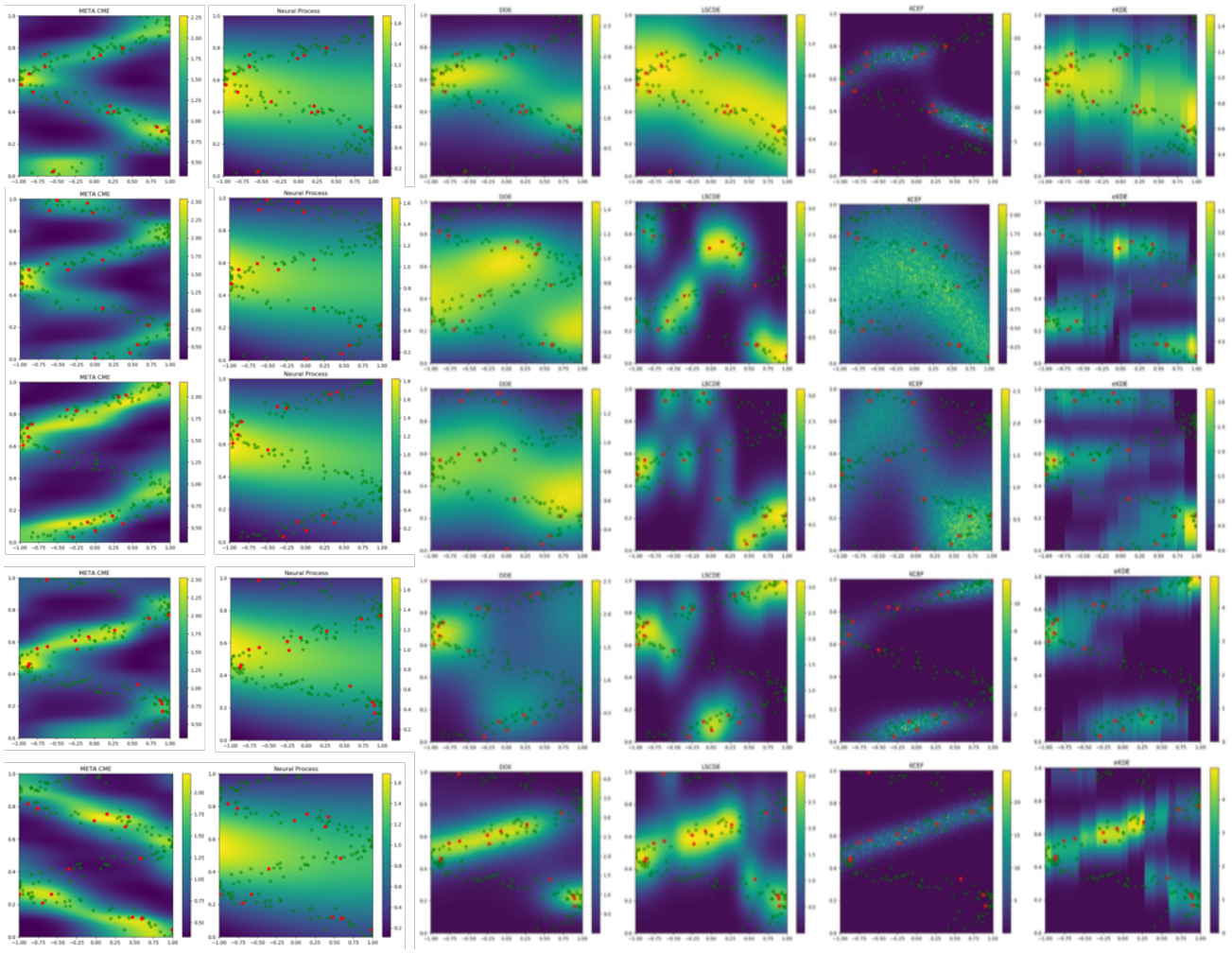

Figure 8: In Order (synthetic dataset): MetaCDE (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE
The red dots are the context/training points and the green dots are points from the true density.

6 Further insight to the Ramachandran plots

6.1 Further details on Ramachandran plots

To better understand the problem that we are tackling, consider the images below which represent different molecules with their respective fragments whose dihedral angle chemist measure. "*A dihedral angle is the angle between two intersecting planes. In chemistry, it is the angle between planes through two sets of three atoms, having two atoms in common. - Wikipedia*".

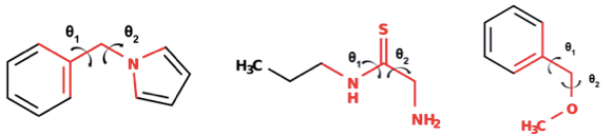

Figure 9: Example of molecules and the respective angles θ_1, θ_2 that we are trying to model. From this image it also becomes apparent how the multimodality arises as there are clearly some spatial symmetries. In the density plots below, the x-axis is θ_1 and the y-axis is θ_2

6.2 Additional information on the experimental setup

In this experiment, we look into the Ramachandran plots for molecules. Each plot indicates the energetically stable region of a pair of correlated dihedral angles in the molecule. Specifically, we are interested in estimating the distributions of these correlated dihedral angles. We compute the conditional density for each correlated dihedral angles, given 20 context points at testing time. For our meta-learning training we use 20 context points and 60 targets points.

Note that the data was extracted from crystallography database (1). It is possible that some specific pairs of dihedral angles are rarely seen in the dataset, Hence, we may obtain a conditional density with high probability on the region without any observations in some cases. This is reasonable as the database covered only a small part of the chemical space and some potential area could be overlooked. Given that we assume that the support of our conditioning variable x ranges from $[-\pi, \pi)$, we will inevitable also compute conditional distribution on areas where the configurations are not defined and hence the densities in those areas can be safely ignored as a computational chemist would not have queried these configurations in the first place.

6.3 Model specifications

For our MetaCDE we used a 3 hidden layer NN with *tanh* activation functions for all of our feature maps. We cross validate over 32 and 64 hidden nodes per layer and $\lambda = 1.0, 0.1$ for the regularization parameter and over $1.0, 0.5, 0.3$ for the KDE lengthscale. We fix the learning rate at $1e$-3 and set $\kappa = 10$.

- KCEF: we used the CV function that was in built in their Github repository (optimizing the parameters from a range $[1e-1, 10]$ `https://github.com/MichaelArbel/KCEF`)
- LSCDE: We CV for σ in logspace$(-3, 5, 20)$ and λ in logspace$(-5, 5, 20)$
- ϵ-KDE: We CV over ϵ in linspace$(0.5, 3, 15)$ and bandwidth in linspace$(0.01, 3, 15)$
- DDE: We CV over bandwidth of 0.5 and 1.0

NOTE: Furthermore it looks like our method is not able to always capture the true trend given the limited amount of data. However, it seems to be able to capture some interesting **multimodal** patterns that would be useful to scientist to include in their models. Recently, there has been work done on these Ramachandran plots for Molecules by handcrafting the density maps. Our model allows us to compute the density maps without prior knowledge.

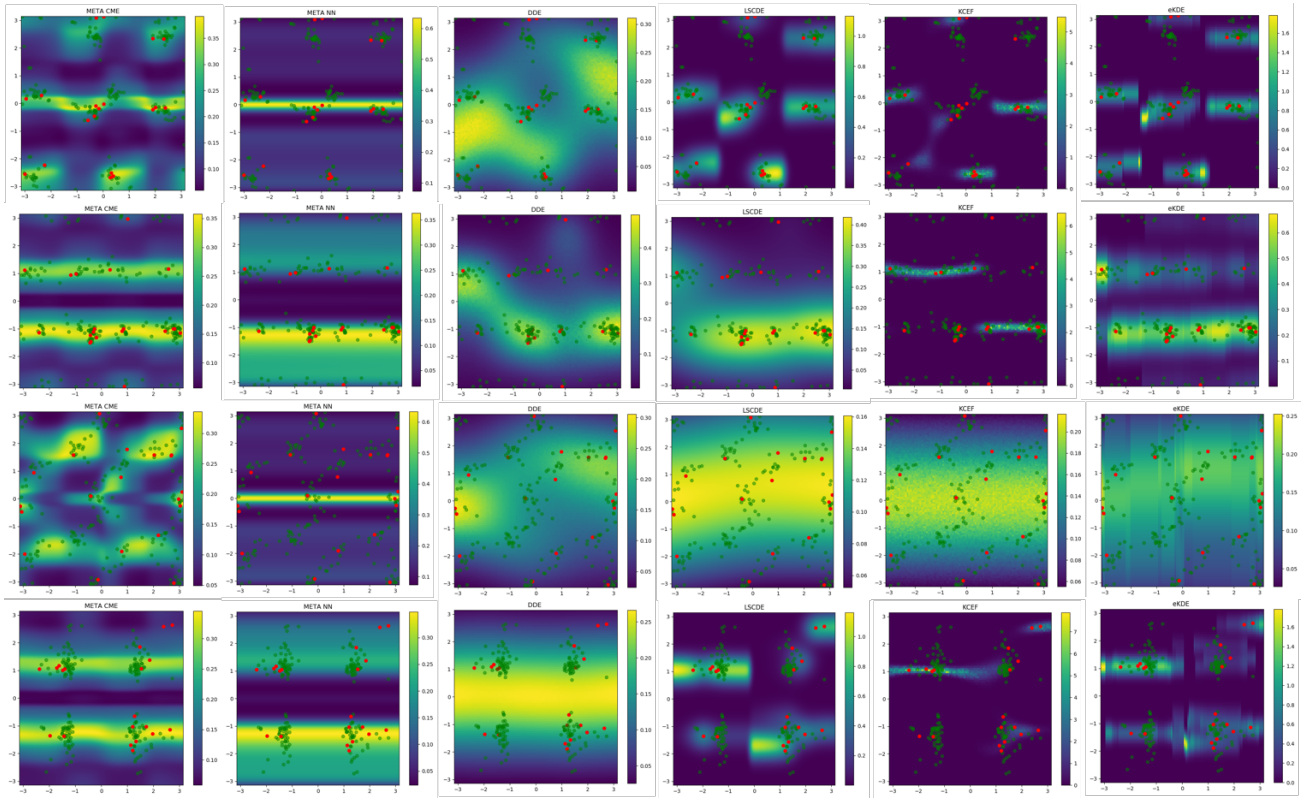

Figure 10: In Order (synthetic dataset): MetaCDE (ours), MetaNN (ours), NP, DDE, LSCDE, KCEF, ϵ-KDE The red dots are the context/training points and the green dots are points from the true density.

7 Illustration of the NYC taxi dataset

7.1 Experimental Setup

We have extracted the publicly available dataset from the website [1]. We have first of all restricted ourselves to drop-off locations in from -74.1 to -73.7 in longitude and 40.6 to 40.9 in latitude. Next we have given our meta learning model 200 datapoints for context during training and 300 for target. At testing time we are presented with 200 context points and are required to compute the conditional density given a tip. In this case each task is one specific pickup location. Again, we are using a 3-hidden layer NN and CV over $32, 64, 128$ nodes and $\lambda = 0.1$ or 1.0 and over $1.0, 0.5, 0.3$ for the KDE lengthscale. We use the *Adam* optimizer and fixed the learning rate to $1\mathrm{e}-3$. We also set $\kappa = 10$.

7.2 Note on the dataset

In the main text we have seen how the drop-off density changes as we increase the amount of tips. This move of density illustrates well the data itself, as one is more likely to pay higher tips for longer journeys. Below we have plotted the drop-off locations of one specific pickup location colored with the respective tips paid.

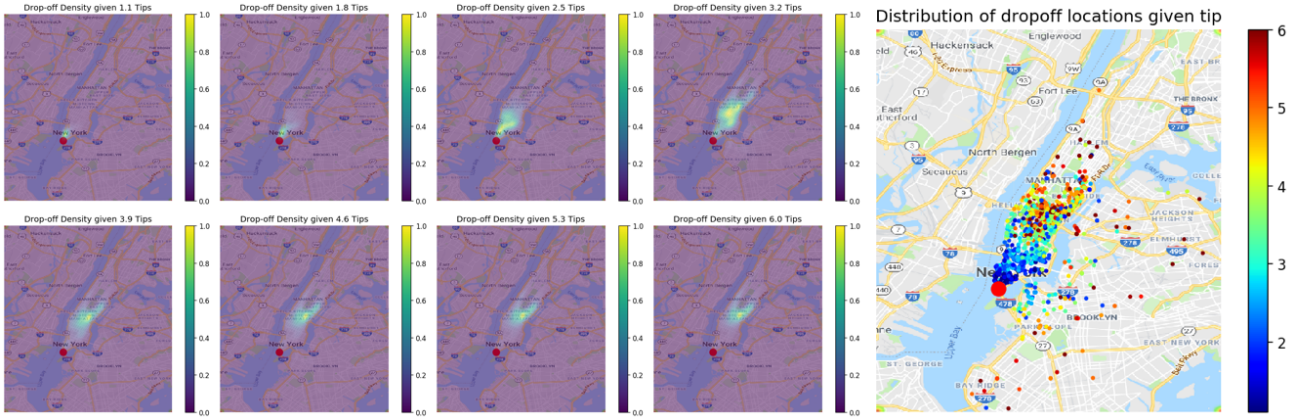

Figure 11: Density of the dropoff locations with increase in tips (right) the context and target points corresponding to the pickup locations (Big red dot the the pickup location)

[1] Data has been taken from: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

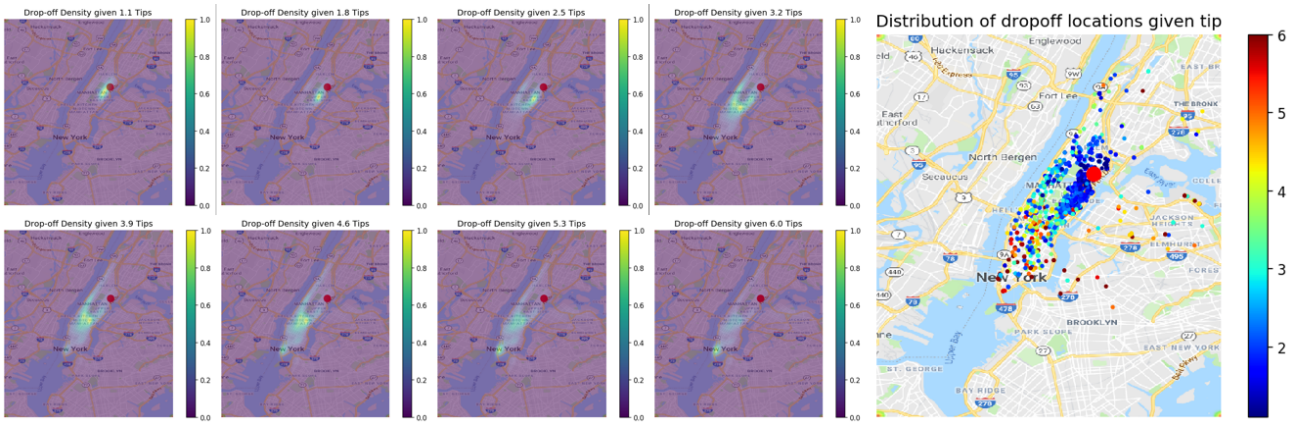

Figure 12: Density of the dropoff locations with increase in tips (right) the context and target points corresponding to the pickup locations (Big red dot the the pickup location)

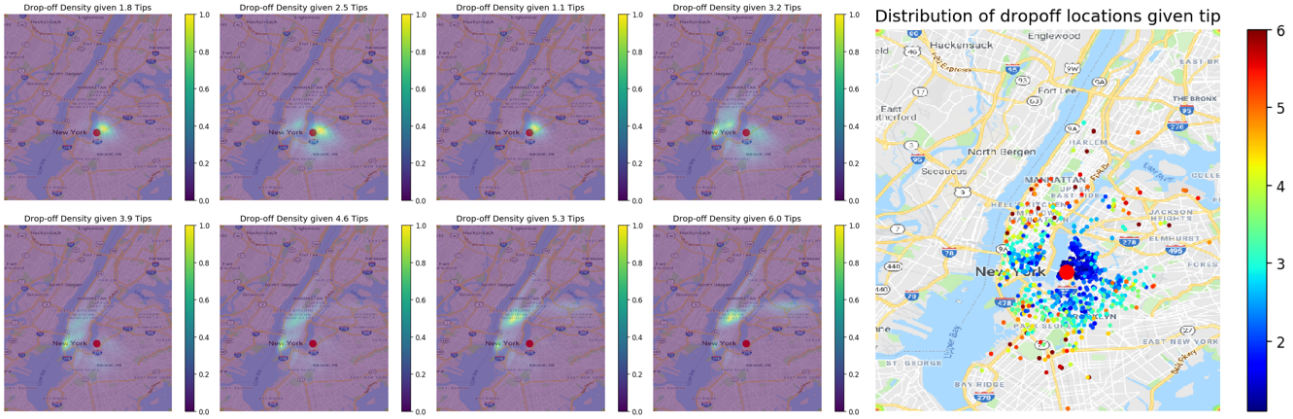

Figure 13: Density of the dropoff locations with increase in tips (right) the context and target points corresponding to the pickup locations (Big red dot the the pickup location)

References

[1] Saulius Gražulis, Adriana Daškevič, Andrius Merkys, Daniel Chateigner, Luca Lutterotti, Miguel Quiros, Nadezhda R Serebryanaya, Peter Moeck, Robert T Downs, and Armel Le Bail. Crystallography open database (cod): an open-access collection of crystal structures and platform for world-wide collaboration. *Nucleic acids research*, 40(D1):D420–D427, 2011.

[2] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[3] Feng Liu, Wenkai Xu, Jie Lu, Guangquan Zhang, Arthur Gretton, and Dougal J Sutherland. Learning deep kernels for non-parametric two-sample tests. *arXiv preprint arXiv:2002.09116*, 2020.

[4] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.