A Details on the models and benchmarks

A.1 Benchmarking details

A.1.1 Regression

For regression on the dataset, we perform leave-one-out cross validation. For the single solvents, we leave out one solvent at a time. For the full data, we leave out one solvent ramp at a time. We measure the performance of the model on each leave-one-out data split, then take the mean of their performance across the dataset. We exclude any experiments involving acetonitrile and acetic acid, due to the observed side-reactions. In addition, when considering the testing in single solvent data, we create a set of single data-points by averaging over repeated measurements, in order to remove mean error weighting from the longer residence times, in order to understand if the models catch the time-series nature of the data.

A.1.2 Transfer learning

As above, we perform leave-one-out cross validation on the solvent ramps in the catechol dataset. However, when we train each model, we append the training data from the ethyl dataset, alongside a binary feature indicating which dataset each observation is from. We also replace the three outputs of the catechol dataset (SM, Product 2, Product 3) with a single column, Product, which is the sum of the two products. This allows us to compare across the two datasets, since the ethyl dataset only has a Product column.

A.1.3 Active learning and Bayesian optimization

For Bayesian optimization we optimize the weighted objective function:

$$f(S_A, S_b, b, \tau, T) = \lambda_1(P_2 + P_3) + \lambda_2 \frac{P_2}{P_2 + P_3} - \lambda_3 \frac{T - 175}{50} - \lambda_4 \tau \tag{4}$$

where S_A is solvent A, S_B is solvent B, b is the percentage composition of solvent B, τ is the residence time, T the temperature, and P_2 / P_3 the yields of Products 2 and 3 respectively. We set the weight parameter values to:

$$\lambda_1 = 5; \quad \lambda_2 = 1; \quad \lambda_3 = 3; \quad \lambda_4 = \frac{1}{20}$$

For the Upper Confidence Bound acquisition function we use the standard exploration parameter $\beta = 1.96$.

For locations with repeated measurements we simply consider average of all observations as the true product yields. All acquisition function optimizations are done through a simple exhaustive search of the space.

A.2 Model details

In this section, we provide the details necessary to reproduce the models used in the experiments. Any information that is not listed here can be found in our code, at `https://github.com/jpfolch/catechol_solvent_selection`.

A.2.1 Gaussian processes

We implement the GP models in this paper in BoTorch v0.13.0 [61]. We use the priors recommended by Hvarfner et al. [71], to ensure good performance across featurizations of different dimensions. We use an RBF kernel, with the lengthscale prior

$$p(\ell) = \mathcal{LN}(\sqrt{2} + \log\sqrt{D}, \sqrt{3})$$

All GPs were trained using the MLII likelihood (maximum a posterior), with a training timeout of 30 seconds. For all of the GP extensions (in Table 3), we use the Spange featurization.

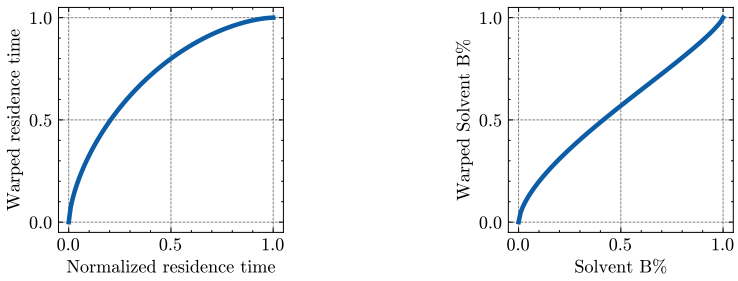

Figure 6: An example of a learned input warping, after training the GP on the full dataset.

BaselineGP. This model is a GP trained only using the residence time, and the temperature. This model does not factor in which solvent each experiment is from.

DeepGP. This model first trains a BaselineGP, then uses that as a mean function for another GP. In this way, far away from known solvents this model will fall back to the BaselineGP as a prior.

Decomposed kernel. We take inspiration from Ru et al. [58], and separate our kernel into two parts. Specifically, we consider the input to the model to be the concatenation of the solvent featurization, f, and the non-featurized inputs, x, which include residence time and temperature. We then use the following kernel,

$$k_{\text{decomp}}([x, f], [x', f']) = k_x(x, x') \cdot k_f(f, f') + k_x(x, x') + k_f(f, f')$$

Similarly to the deep GP, this allows the features in x to still contribute to the prediction, even when the unseen solvent is far from the known solvents.

Multitask GP. We use two different types of multitask GP in this paper. First, in Section 3.3, we use a multitask GP to represent each of the three measured yields. This kernel consists of a data kernel, and a task kernel,

$$k_{\text{MT}}([x, o], [x', o']) = k_x(x, x') \cdot k_o(o, o'),$$

where k_o is an $O \times O$ matrix (for this dataset, $O = 3$) that is used to learn the correlations between the outputs. Since all outputs are observed for each experiment, we can use a Kronecker structured kernel.

In Section 3.4, we use another multitask GP with 2 tasks, where each task corresponds to one of the two datasets. We use the same kernel as above, however only one task is observed at each reaction condition.

Input warping. In Section 3.3, we describe how the underlying chemistry is nonstationary. To attempt to address this, we take inspiration from Snoek et al. [60] and Balandat et al. [61], learning a bijective map $\phi : [0, 1] \to [0, 1]$ that can capture the nonlinear effect of mixing solvents. This map has hyperparameters that can be learned,

$$S_{A \cup B}(b) = (1 - \phi(b))S_A + \phi(b)S_B, \qquad \phi(b) = 1 - (1 - b^\alpha)^\beta,$$

where ϕ is the Kumaraswamy cumulative distribution function. We place a log normal prior on the parameters, $\alpha, \beta \sim \mathcal{LN}(0, \sqrt{0.3})$. This prior has median value of 1, which corresponds to a linear mapping.

We also use the input warping for the residence time. Since most of the reaction occurs in the first few minutes of the reaction, the lengthscale is far shorter compared to the later parts of the reaction. We find that this is indeed learned by the model, as shown in Figure 6; the mapping effectively 'spreads out' the observations early in the reaction, while compressing the later observations that tend to have a slower rate of change. Whilst the warping for the solvent composition learns a slight sigmoidal shape, we show experimentally in Section 3.3 that warping this feature does not improve regression performance.

A.2.2 Neural networks

Two types of neural network models were constructed for the regression tasks. The first was a standalone multilayer perceptron (MLP) model, and the second combined a large language model (LLM) backbone with an MLP head.

For the single-solvent task, the MLP model took as input the reaction time, temperature, and a feature vector representing the solvent. The network architecture consisted of two hidden layers with 128 and 64 neurons, respectively, each followed by ReLU activations and dropout (dropout rate of 0.5), and an output layer with 3 neurons.

For the mixed-solvent task, the MLP model used the same architecture, but the solvent input was computed as a sigmoid-weighted combination of the individual solvent feature vectors:

$$S_{A\cup B} = (1 - \sigma_\theta(b))\, S_A + \sigma_\theta(b) S_B,$$

where S_A and S_B are the featurizations of solvents A and B, b is the percentage of solvent B in the mixture, and σ_θ is a sigmoid function with trainable parameters θ.

The second model architecture used pretrained LLMs—**RXNFP** and **ChemBERTa**—to generate embeddings from reaction SMILES strings. For the single-solvent task, the SMILES representation of the reaction using the selected solvent was passed through the LLM to obtain the corresponding embedding. For the mixed-solvent task, the SMILES strings of the reactions carried out in solvents A and B, denoted RS_A and RS_B, were each processed independently through the LLM to produce embeddings $\mathbf{E}_A$ and $\mathbf{E}_B$, respectively. These embeddings were then combined using a sigmoid-weighted sum:

$$\mathbf{E}_{A\cup B} = (1 - \sigma_\theta(b))\, \mathbf{E}_A + \sigma_\theta(b) \mathbf{E}_B,$$

where b is the percentage of solvent B in the mixture and σ_θ is a sigmoid function with trainable parameters θ.

The resulting embedding was concatenated with the time and temperature, and passed through an MLP with the same architecture as the standalone MLP model. The LLM backbones were kept frozen during training, and only the MLP head was optimized.

The `ChemBERTa` model and tokenizer used were `seyonec/ChemBERTa-zinc-base-v1`, loaded via the Hugging Face `transformers` library. Similarly, the pretrained `RXNFP` model and tokenizer used are available from the `rxnfp` repository.

All models were trained using a learning rate of 0.001, a batch size of 32, for up to 400 epochs, or until reaching a maximum runtime of 720 minutes.

A.2.3 ODE

The ODE models were trained with a learning rate of 0.001, and 100 epochs. For the latent state and latent dynamics, we used a 32-dimensional space, and for all of the other representations we used a 64-dimensional space. Further information can be found in the provided code.

A.3 Additional results

We showcase additional results for Neural Processes [54] and graph Gaussian processes [50, 51] in table 5.

B Details on data collection

B.1 Reactor details

Here we include the reactor and detail procedures.

The automated reactor setup used to collect the data is shown in Figure 7. Knauer Azure 4.1S pumps fitted with stainless steel 10 mL pump heads were used as pumps 1 and 2. All tubing used for the entire reactor was made of 316 stainless steel (1.5875 mm OD, 1 mm ID). An Agilent inline jet weaver HPLC mixer (350 μL volume) was used as an inline mixer to ensure the reactant solution was homogeneous before entering the reactor. An Agilent 6890 GC oven was used to heat the stainless steel coiled reactor (1.5875 mm OD, 1 mm ID, 7.95 mL volume) during the reaction to the desired temperature. A customized cooling system made from an aluminum block and a Peltier assembly

Table 5: Regression performance on the single solvent dataset. Mean squared error (MSE) and negative log predictive density (NLPD) are averaged across all leave-one-out data splits. We include the shortest path kernel (sp) and the encoded shortest path kernel (esp).

		Single solvent	
Model	Featurization	MSE (↓)	NLPD (↓)
NP	acs	0.153	-1.173
	drfps	0.139	-1.587
	fragprints	0.135	-1.495
	spange	0.089	-1.472
GraphGP	sp	0.046	2.464
	esp	1.068	2.453

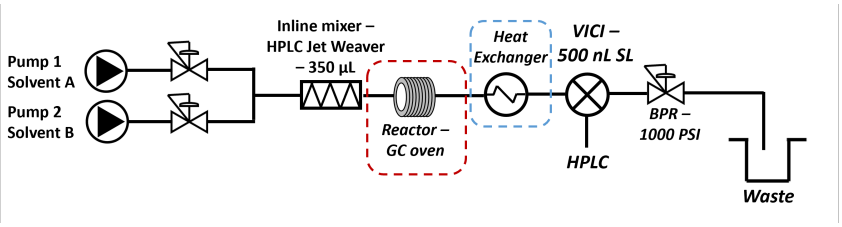

Figure 7: Piping & instrumentation diagram of the automated continuous flow coiled reactor used to collect the transient flow data reported in this paper.

was then placed inline to rapidly cool the flow of solution and quench the reaction. A Vici four port-2 position sampling valve followed the Peltier to sample small aliquots (500 nL) into the HPLC for online analysis measurements of the reaction. An IDEX 1000 PSI BPR was then placed before the waste tubing of the reactor to depressurize the reaction solution back to atmospheric pressure. The pumps, oven and Vici valve were automated by code developed in house in Python.

B.1.1 Methods

A typical reaction run was performed as following:

1. The reactant solutions were made up by adding allyl phenyl ether (50 μL) and the internal standard - ethyl benzene (50 μL) in to both solvent A and solvent B (250 mL) in separate volumetric flasks.

2. The reactant pumps were primed with their respective solvents and pumped through the system at 1 mL min^{-1} for 15 minutes.

3. The pumps were then primed with the reactant solutions and pumped through the system at 1 mL min^{-1} for 5 minutes.

4. The HPLC was started and a sequence was created to record external sampling via the Vici Valve.

5. The python code that runs the experiments was then initialized and the experiment was started.

6. Once the reaction run was completed, the reactor is flushed with their respective solvents for 10 minutes at 1 mL min^{-1}, followed by a flush of the system with a miscible solvent (usually IPA) and cleaned for the next reaction. The data was stored in a SQL database and is then deconvolved offline.

B.2 Fine-tuning calibration via optimization

The HPLC data we obtained is uncalibrated, which means we cannot calculate yields directly from the peak areas collected from online HPLC measurements. However, the yields of each product follows the linear relationship with peak areas:

$$y_{product} = \epsilon_{product} \times \frac{c_{IS}}{c_0} \times \text{peak\_ratio} \tag{5}$$

where c_{IS} is the internal standard concentration in mol L^{-1}, c_0 is the initial concentration of starting material in mol L^{-1}, and ϵ is the *calibration constant*. The peak_ratio refers to value given by dividing the area of the peak of interest (starting material, product 2 or product 3) by the peak area of the internal standard. This constant is calculated by performing calibrations of the HPLC detector with injections of pure compounds at different concentrations, while keeping the internal standard concentration constant, and therefore observing the linear relationship and obtaining the response factor of the compounds. Obtaining a pure sample of Product 2 and Product 3 however, turned out to be particularly difficult due to the compounds being isomers, making the separation of the pure products tough. Therefore, we instead focused on using the estimates we had and then fine-tuning them via an optimization procedure.

Our initial HPLC tests gave us the following estimates:

$$\hat{\epsilon}_{SM} = \frac{1}{1.5}; \quad \hat{\epsilon}_{P2} = \frac{1}{3}; \quad \hat{\epsilon}_{P3} = \frac{1}{3}$$

From here, we decided to fine-tune the estimates in order for the calculated yields to ensure the reaction yields were mass balanced. We identified specific measurements where we expected full conversion (i.e. the sum of yields should be 100), and we further allowed for experimental concentrations to vary according to the error in the laboratory analytical pipettes used for making the reactant solutions. This results in the following optimization problem, where we penalized deviation from our initial calibration measurements, and deviation from full conversion at specified measurements $\mathcal{K}$:

$$\begin{aligned}
\min_{\{c_i,\,\epsilon_j\}} \quad & \alpha \sum_i (c_i - 2.25)^2 + \beta \sum_j (\epsilon_j - \hat{\epsilon}_j)^2 + \gamma \sum_{k \in \mathcal{K}} \left(\sum_j y_{kj} - 100 \right)^2 \\
\text{where} \quad & y_{ij} = \text{const} \cdot \text{peak_ratio}_{ij} \cdot \epsilon_j \cdot c_i, \quad \forall i = 1, ..., 1227; j \in \{SM, P2, P3\} \\
& c_i = c_{i'} \qquad \text{if } i, i' \text{ are in the same experimental run}
\end{aligned}$$

with constraints to restrict total yield under 100% and possible errors in concentrations:

$$\begin{aligned}
\sum_j y_{ij} \leq 100, \quad & \forall i \\
c_i \in [1.25, 2.5], \quad & \forall i \\
0.2 \leq \epsilon_j \leq 0.5, \quad & \forall j
\end{aligned}$$

where:

- c_i are the corrected concentration ratios,
- ϵ_j are the calibration scaling factors for each compound,
- peak_ratio_{ij} are the observed HPLC peak area ratios,
- $\mathcal{K}$ is the set of indices where full conversion is expected,
- α, β, and γ are weighting parameters.

we optimized with $\alpha = \beta = \gamma = 1$, optimized using `scipy`'s `minimize` function with the Sequential Least Squares Programming (SLSQP) algorithm. To select the initial values, we used a 100,000 initial grid search. This resulted in the following parameter estimates:

$$\epsilon_{SM} = 0.525; \quad \epsilon_{P2} = 0.222; \quad \epsilon_{P3} = 0.361$$

B.3 Spange descriptor interpolation

The descriptors from Spange et al. [40] were obtained from the supplementary material on the paper. However, there are a few values missing from some rows, including for the solvents we gathered data for. In order to estimate the missing values, we trained a multi-task Gaussian process model on the whole table, under a Taniamoto kernel, which we then used to predict the missing values that are used for all the main methods in the paper.