Supplementary data for

Artificial Intelligence in Steam Cracking Modeling: A Deep Learning Algorithm for Detailed Effluent Prediction

Pieter P. Plehiers [a], Steffen H. Symoens [a], Ismaël Amghizar [a], Guy B. Marin [a], Christian V. Stevens [b], Kevin M. Van Geem [a,*]

[a] *Laboratory for Chemical Technology, Department of Materials, Textiles and Chemical Engineering, Ghent University, Gent 9052, Belgium*
[b] *SynBioC Research Group, Department of Green Chemistry and Technology, Faculty of Bioscience Engineering, Ghent University, Gent 9000, Belgium*

* Corresponding author.
E-mail: Kevin.VanGeem@UGent.be (K.M. Van Geem).

1. Mathematical aspects

1.1. Activation functions

1.1.1. *Rectified linear unit*

The rectified linear unit (ReLU) activation function is a linear function, but with a lower bound. The mathematical expression is given in Eq. (S1). The fact that the function has no upper bound can potentially lead to instability in the network. The ReLU function is compared to other activation functions used in this work in Fig. S1. The simple nature of this activation function makes it an excellent choice in very large networks seen for example in convolutional artificial neural networks (ANNs) in computer vision and image recognition.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \quad \text{(S1)}$$

1.1.2. *Sigmoid*

The sigmoid function has both a lower (0) and upper (1) bound. This type of activation function is typically used in the output layer, when outputs are bounded values, *i.e.* in this work, wherever the outputs have been normalized individually. Mathematically the sigmoid is given by Eq. (S2), graphically by Fig. S1.

$$\mathrm{f(x)} = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}} \quad \text{(S2)}$$

1.1.3. *Hyperbolic tangent*

The hyperbolic tangent (tanh) function also has lower (–1) and upper (1) bound. It is used for similar purposes as the sigmoid function, but in cases where the data has been normalized to the interval [–1,1] instead of [0,1]. The mathematical expression for the tanh function is given by Eq. (S3), Fig. S1 plots this function.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad \text{(S3)}$$

1.1.4. *Softmax*

The softmax activation function differs from the previous functions, in that it does not depend on the output value of a single node, but on the output values of all nodes in the considered layer. The softmax output of a node in a certain layer is calculated via Eq. (S4). This implies that all the outputs of the layer (o) sum to one, making a softmax activation function well suited for ranking candidates in decision problems or in this work, ensuring that predicted compositions are normalized to one.

$$f(x) = \frac{o}{\sum_{i\ in\ layer} o_i} \quad \text{(S4)}$$

1.2. Principal component analysis

Principal component analysis (PCA) is a data analysis and dimension reduction method that transforms the multidimensional variable space into another variable space in which the base vectors (principal components or PCs) align with the directions of maximal variation in the initial dataset. In this work, the main purpose of the PCA is to identify outliers in the dataset and to predict a priori for which naphthas poor predictions are to be expected. PCA boils down to an eigenvalue problem so that each data point in the variable space can be transformed into a point in the PC space via Eqs. (S5)–(S7), in which **z** is the vector of principal components of the input $\boldsymbol{x}$, and $\boldsymbol{A}$ is the transformation matrix of which the columns contain the eigenvectors of the variance matrix of the dataset, $\boldsymbol{S}$ is the variance covariance matrix of the dataset and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues.

$$\det(\boldsymbol{S} - \lambda \boldsymbol{I}) = 0 \tag{S5}$$

λ are the individual eigenvalues and $\boldsymbol{I}$ is the identity matrix**.**

$$\mathbf{S} = \mathbf{A}\boldsymbol{\Lambda}\boldsymbol{A}^{-1} \tag{S6}$$

$$\boldsymbol{z} = \boldsymbol{A} \cdot \boldsymbol{x} \tag{S7}$$

It can be shown that the variance of the data along the principal components equals the respective eigenvalues of those PCs.

In multivariate analyses such as the one presented here, the distance between two points can be of interest, in particular the distance of a new point to the dataset average. The multivariate distance measure considered here is the Mahalanobis distance, which—when calculated in the PC space—takes on the simple form of Eq. (S8) or in the explicit form as Eq. (S9). Based on this mathematical representation, the Mahalanobis distance is distributed according to the Hotelling T^2 distribution, allowing the definition of a critical Mahalanobis distance (MD) in Eq. (S10). n is the number of samples in the training set, while d is the dimensionality of the PC space. α is the confidence level and represents the amount of data that will be enclosed within the ellipsoids of critical MD. For the naphtha composition dataset, n equals 240, d equals 3 and a probability α of 0.9 is chosen, resulting in a critical MD of 2.5. For the effluent prediction dataset, n equals 12240 and d equals 6, resulting in a critical MD of 3.3 for the same probability level of 0.9.

$$MD^2 = \boldsymbol{z}^T \cdot \boldsymbol{\Lambda}'^{-1} \cdot \boldsymbol{z} \tag{S8}$$

$$MD^2 = \sum_i \frac{z_i^2}{\lambda_i} \tag{S9}$$

$$MD^2_{crit,\alpha} = T^2_{crit,\alpha} = \frac{d(n-1)}{n-d} F_{\alpha,d,n-d} \tag{S10}$$

T^2 and F are the Hotelling T^2 distribution and F statistic, respectively.

2. Detailed in- and outputs

This section contains a short description of the in- and outputs of the different networks. The input vector to Network 1 has the following form: density (as specific gravity), vapor pressure, total paraffin weight fraction, total isoparaffin weight fraction, total olefin weight fraction, total naphthene weight fraction and total aromatic weight fraction. All inputs are to be normalized using the normalization procedure described in Section 3.1. The output vector contains—in that order—initial-, 50%- and final boiling points, which are all normalized. Reverse normalization will yield temperatures (Unit: °C).

The input vector to Network 2 contains, again correctly normalized, initial boiling point (IPB), mid boiling point (BP50), final boiling point (FBP), total paraffin weight fraction, total paraffin weight fraction, total isoparaffin weight fraction, total olefin weight fraction, total naphthene weight fraction and total aromatic weight fraction. The output is a vector of which the sum of the elements is close to one. The vector is shown below.

A_6	A_7	A_8-1	A_8-2	A_9	N_5	N_6-1	N_6-2	N_7	N_8	N_9	O_5	O_6		
P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}

A_8-1 corresponds to xylene, while A_8-2 corresponds to ethylbenzene. Similarly, N_6-1 and N_6-2 correspond to methylcyclopentane and cyclohexane, respectively.

The input of Network 3 combines the output of Network 2 described above and the normalized characterizations of the process, resulting in the vector below.

Output vector of Network 2	COT	COP	P/E	M/P	E/E

COT is coil outlet temperature; COP is coil outlet pressure; P/E is the propylene to ethylene ratio; M/P is the methane to propane ratio; E/E is the ethylene to ethane ratio.

The direct output of the network contains 28 species and lumped component. The table below lists these 28 components from left to right and top to bottom as they appear in the model output.

H_2	CO_x	C_2H_2	C_2H_4	C_3H_4
C_3H_6	n/iso-butane	1,3-Butadiene	Butenes	C_5H_{12}
C_5 isoparaffins	C_5 olefins	Cyclopentane	Cyclopentadiene	C_6H_{14}
C_6 isoparaffins	C_6 olefins	C_6 naphthenes	Benzene	Toluene
Xylene	Styrene	C_{7+} paraffins	C_{7+} isoparaffins	C_{7+} olefins
C_{7+} naphthenes	C_{7-9} aromatics	C_{10+} aromatics		

The final network uses the output of Network 2 and gives an output containing 5 normalized properties in the following order: IBP, BP50, FBP specific gravity and vapor pressure.

3. Additional data on model parameter optimization

3.1. General hyperparameter optimization method

Due to the vastness, high-dimensionality and unstructured nature of the hyperparameter space of a given artificial network, it is nearly impossible to determine the globally optimal hyperparameters. Therefore, a structured, heuristic approach is used to arrive to a (locally) optimal set of hyperparameters. A similar approach is used for all models described in the following sections. The network architecture (number of layers and connection of the layers) is chosen based on the considerations in "Setup of the Artificial Neural Networks" and is fixed before proceeding to the hyperparameter optimization. The heuristic search for a local optimum follows a three-dimensional grid-like pattern, in which the influence of the different parameters is evaluated. The three dimensions of the grid are: the number of nodes in each layer, the activation function in each layer and the training batch size. Four different node counts are evaluated: 256, 512, 1024, 2028. The activation function is either a ReLU function or a sigmoid function. Softmax is used only for selected outputs of Network 2. Batch sizes of 8, 16, 32, 64 and 128 are tested. This results in a large number of models, of which a selection is presented in the following sections. The optimal number of training epochs for each model is found at the point where the validation objective function reaches its minimum. For each of these models, the mean squared error on the validation dataset is determined. The optimal set of hyperparameters is derived from the model with the minimal mean squared error.

3.2. From density and vapor pressure to boiling points

Traditionally, the mean squared error is used as objective function in regression problems as it yields a maximum likelihood estimate for the parameters. For this network, the performance of the network has been evaluated using both the mean absolute error and the mean squared error as objective function. It should be noted that besides the actual regression of the network weights, network hyperparameter optimization must also take place. The latter takes place by evaluating the performance on the validation set, *i.e.* the optimal network hyperparameters are obtained by minimizing the validation objective function, in contrast to the training objective function for the network weights. Fig. S2 shows the validation mean squared error and mean absolute error for two models which differ only in the type of training objective function. It is clear that the minimum of the validation mean squared error does not depend strongly on the used training objective function. The minimum of both the validation mean squared error and mean absolute error occurs around the same number of epochs when using the mean absolute error as training objective. Both approaches will converge towards the same solution in the multidimensional parameter space, but via different routes and at different rates. The mean absolute error (MAE) is then chosen for practical reasons, as the corresponding validation mean squared error displays a clearer and earlier minimum in validation mean squared error and because it minimizes both the validation mean absolute error and mean squared error.

The effect of the batch size has been evaluated on a network with 512 nodes in each layer. It is assumed that the similar behavior for the batch size can be observed in all other networks. From Fig. S3, a batch size of 8 is chosen as it achieves the

lowest MAE. Corresponding to intuition, this batch size also requires the least number epochs to attain the minimal MAE. Equal amounts of nodes in each hidden layer are observed to perform best, with 256 nodes per layer being considered optimal. Fig. S4 compares the validation MAE of different networks. It is observed that a network with 1024 nodes in each layer achieves a lower validation MAE than the chosen optimal network. While the validation MAE is an important metric, the difference between the training and validation MAE should also be taken into consideration. A large difference between these two values is indicative of overfitting, *i.e.* the network is too large or complex. The difference between the training and validation MAE for the chosen network with 256 nodes per layer is only 0.003, while for the network with 1024 nodes per layer, the difference increases to 0.01. Finally, better results are obtained when using a ReLU activation function instead of a sigmoid function for the first hidden layer and sigmoid functions for all other layers. This is illustrated in Fig. S5. The optimal configuration achieves the minimum MAE after 1181 epochs. From this point on, the validation loss baseline starts increasing again, indicating the commencement of overtraining. The time required for training depends on several factors, such as the batch size, dataset size, number of epochs and the total number of hidden nodes in the network. Therefore, only the training time of the final models is reported.

3.3. Feedstock reconstruction

Fig. S6 indicates a high validation loss using a batch size of 8. This is the result of instability in the gradient descent algorithm. The accuracy of the prediction for the gradient decreases with decreasing batch size. In certain cases, such as this one, this can lead to divergence of the loss function and failure to find the optimal weights. Contrary to expectations, more epochs seem to be required at lower batch sizes. Again, this could be related to the aforementioned loss in accuracy of the gradient prediction. Fig. S7 shows that two networks achieve similar performance. The network with 256 nodes in the first hidden layer and 512 in the second performs slightly better than the one with 512 in both and is less complex, making this the first choice network. It markedly outperforms the one with 256 in both layers. Optimal performance is attained after 45 285 epochs. Training of the final model is completed in 1 h 39 m on an NVIDIA Quadro M4000 graphics processing unit (GPU).

3.4. Detailed effluent prediction

Due to the wide range of component concentrations, both MAE and mean absolute percentage error (MAPE) are considered as loss function. A model trained on the MAPE achieves a 32% lower MAPE than one trained on the MAE, while the MAE increases by only 18%. The main reason for this increased performance is the wide range of concentrations covered by the different components which must simultaneously be optimized. With the MAE, the optimization tends to favor improving the predictions for components with high concentrations. As the concentration of the less dominant components can be of significant importance in down-stream processing, it is important that they are accurately predicted. Using the MAPE ensures that the optimization also attempts to improve the prediction on these low-concentration components. Fig. S8 indicates a clear performance gain with a batch size of 8. Fig. S9 shows that for all tested models, the MAPE decreases with increasing model complexity. A model with 2048 nodes in each layer has also been tested, but the MAPE stagnated at more than 80%. Therefore, the model with 1024 nodes in each of the input layers and 2048 in the next layer is chosen. The minimal MAE this model achieves is $< 0.001\%$, which is arguably below the accuracy of standard analytical equipment. Training time on an NVIDIA Quadro M4000 GPU for the final model is around 18 h. The main reason for the significant longer training time compared to the previous models is the extensiveness of the dataset.

3.5. Property estimation

The MAE is chosen as loss function. Fig. S10 shows that the MAE is relatively insensitive to the batch size. In line with the findings for previous models, a batch size of 8 is chosen. The combination of 256 nodes in the first layer with ReLU activation and 256 sigmoid nodes in the second provides the best performance, according to Fig. S11. Fig. S12 shows that using a sigmoid activation function in both the first and second layers results in slower training, without any performance gain. The final model is trained in 15 min on the same GPU as the previous models.

4. Figures

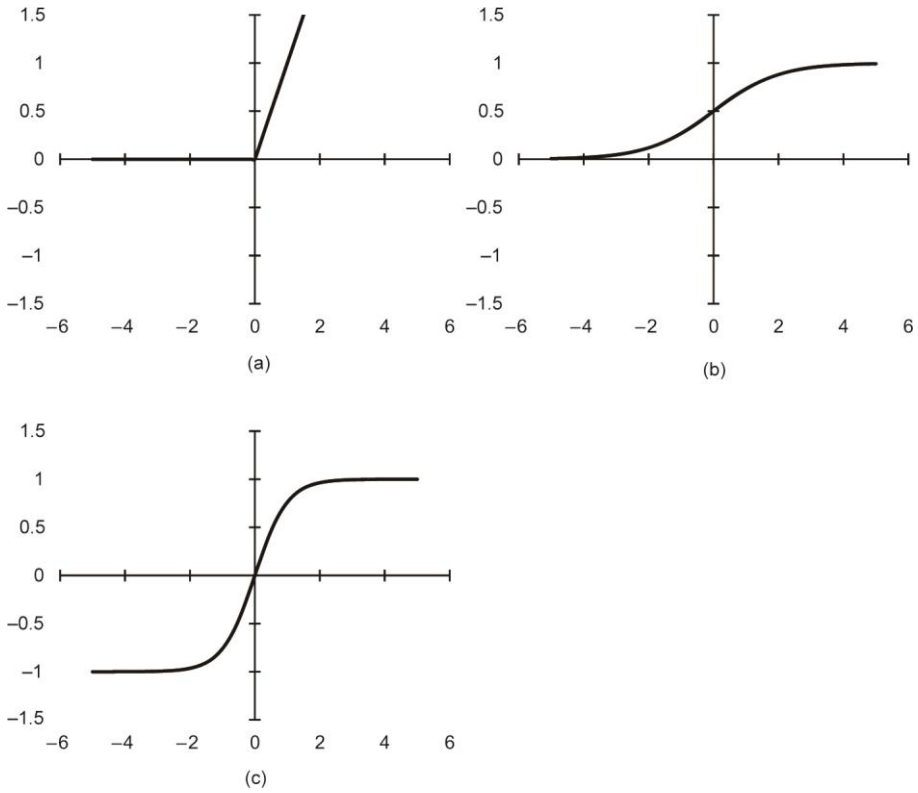

Fig. S1. Plots of different activation functions used in this work. (a) ReLU; (b) sigmoid; (c) tanh.

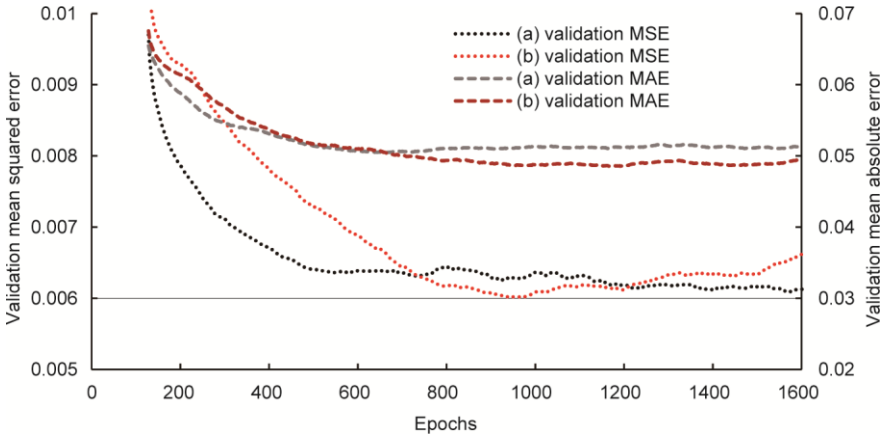

Fig. S2. Validation mean squared error and mean absolute error for two identical networks, except for the training objective function: (a) training objective function is mean squared error (MSE); (b) training objective function is mean absolute error (MAE).

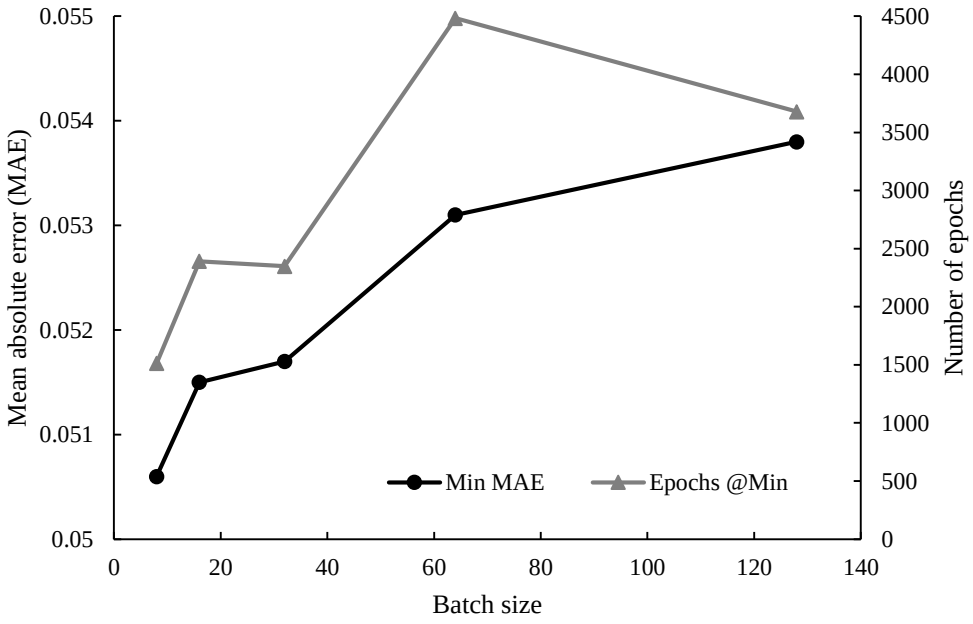

Fig. S3. Minimal MAE and epochs required to attain that minimum with varying batch size for Network 1.

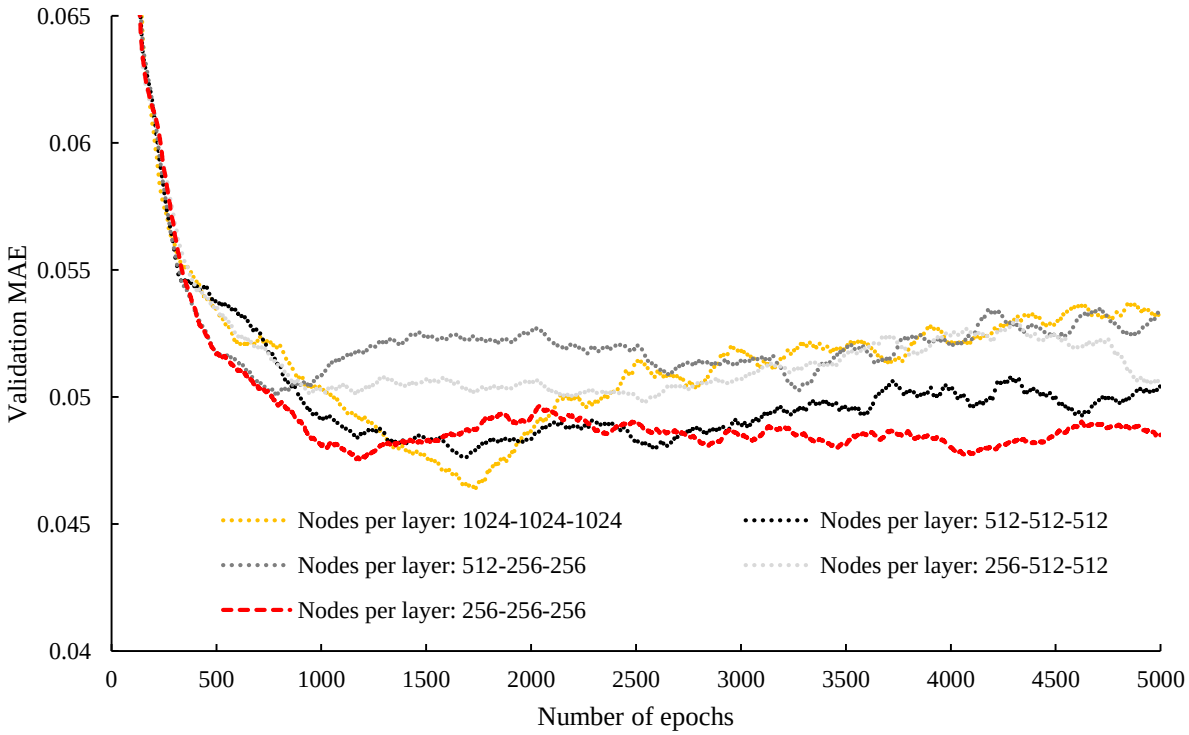

Fig. S4. Validation MAE for several different combinations of nodes per hidden layer for Network 1. All models are trained with a batch size of 8.

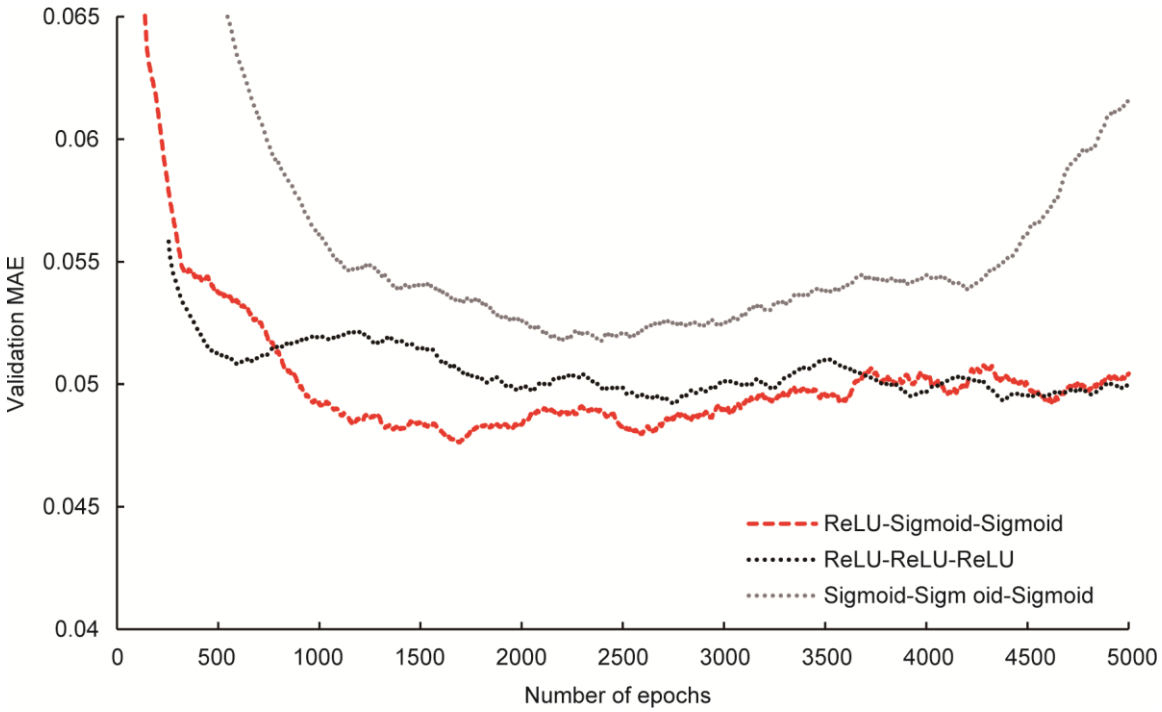

Fig. S5. Validation MAE for different combinations of activation functions in the two hidden layers and the output layer of Network 1. All networks are trained with a batch size of 8.

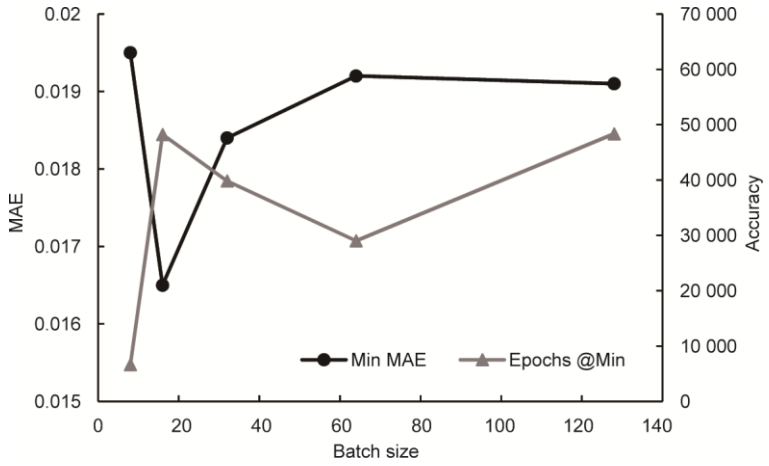

Fig. S6. Minimal MAE and epochs required to attain that minimum, with varying batch size for Network 2.

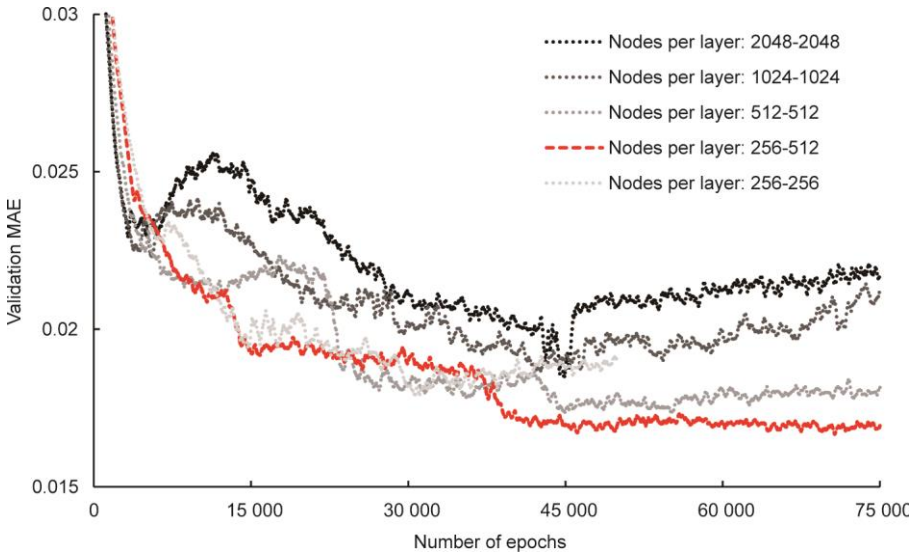

Fig. S7. Validation MAE for several different several different combinations of nodes per hidden layer for Network 2. Both layers use a sigmoid activation function and all models are trained with a batch size of 16.

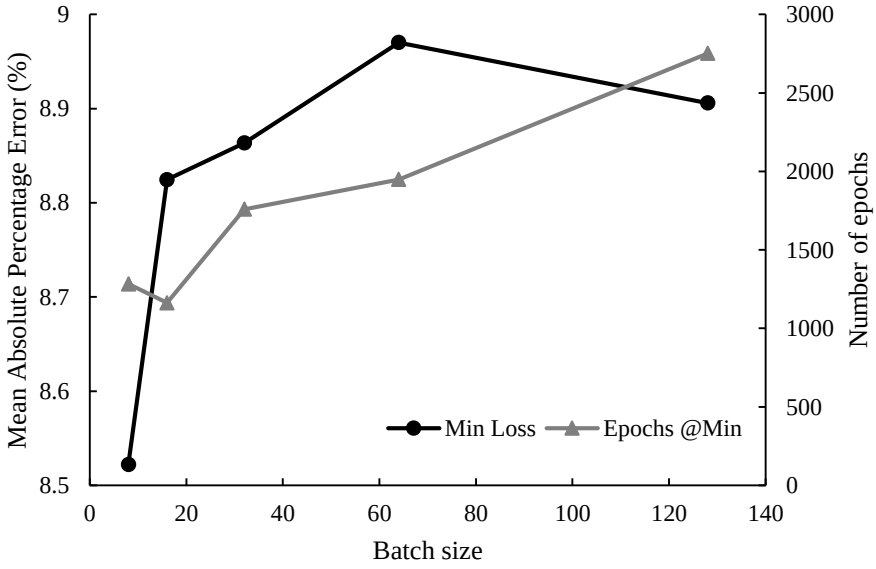

Fig. S8. Minimal mean absolute percentage error (MAPE) and epochs required to attain that minimum, with varying batch size for Network 3.

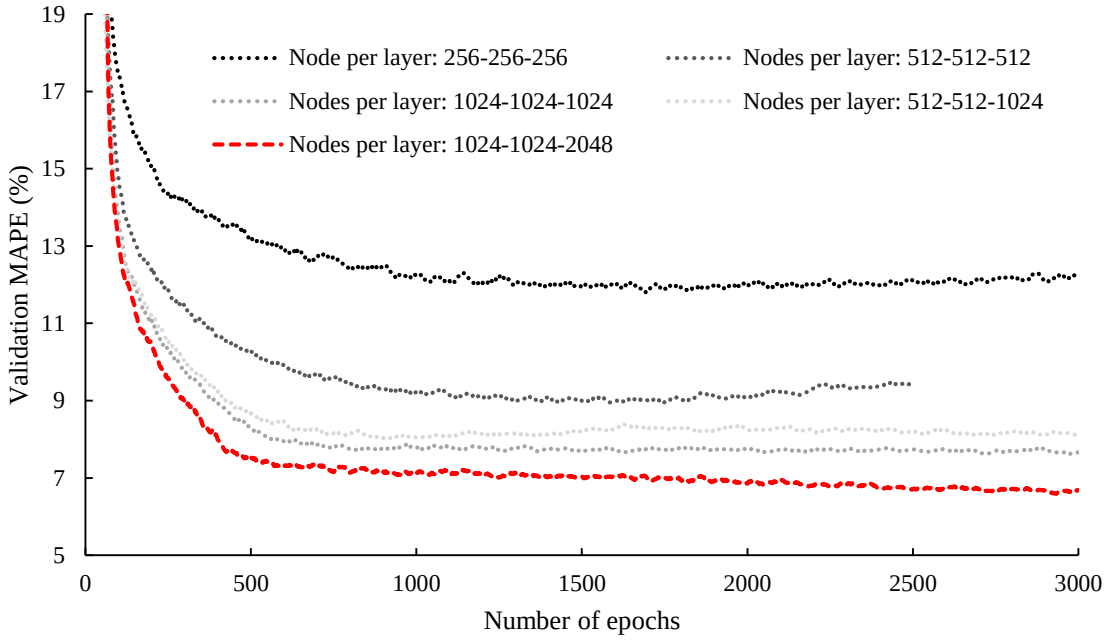

Fig. S9. Validation MAPE for several different several different combinations of nodes per hidden layer for Network 3. All layers use a sigmoid activation function and all models are trained with a batch size of 8.

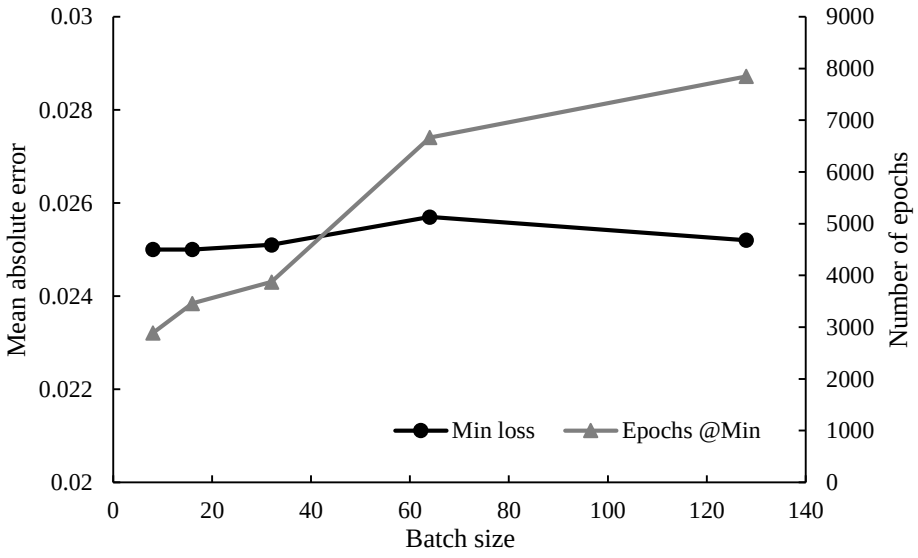

Fig. S10. Minimal MAE and epochs required to attain that minimum with varying batch size for Network 4.

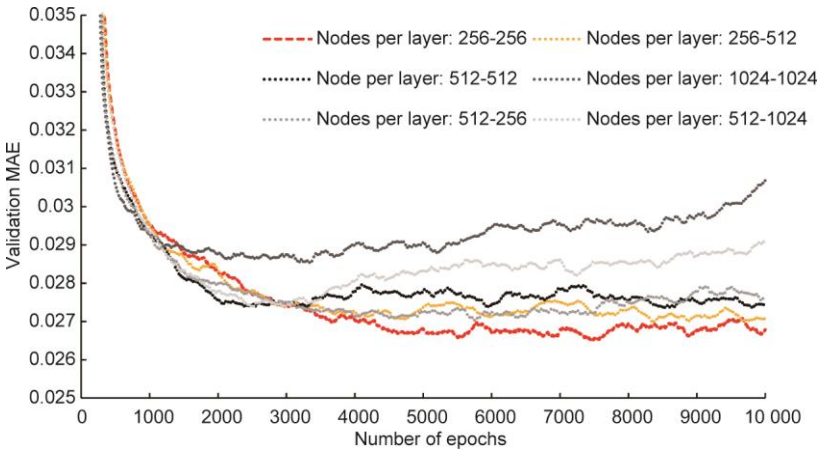

Fig. S11. Validation MAE for several different several different combinations of nodes per hidden layer for Network 4. All models are trained with a batch size of 8.

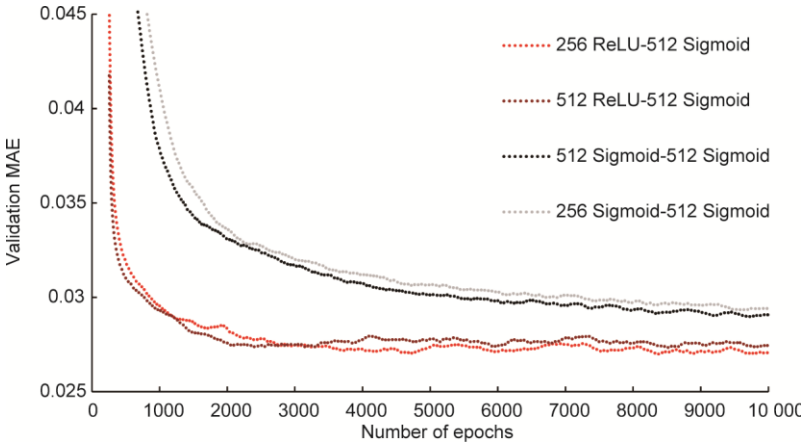

Fig. S12. Validation MAE for variations on Network 4 using different activation functions in the hidden layers. The first layer in the network has 256 or 512 nodes, the second 512. All models are trained with a batch size of 8.

Fig. S13. Absolute correlation matrix of available naphtha data.

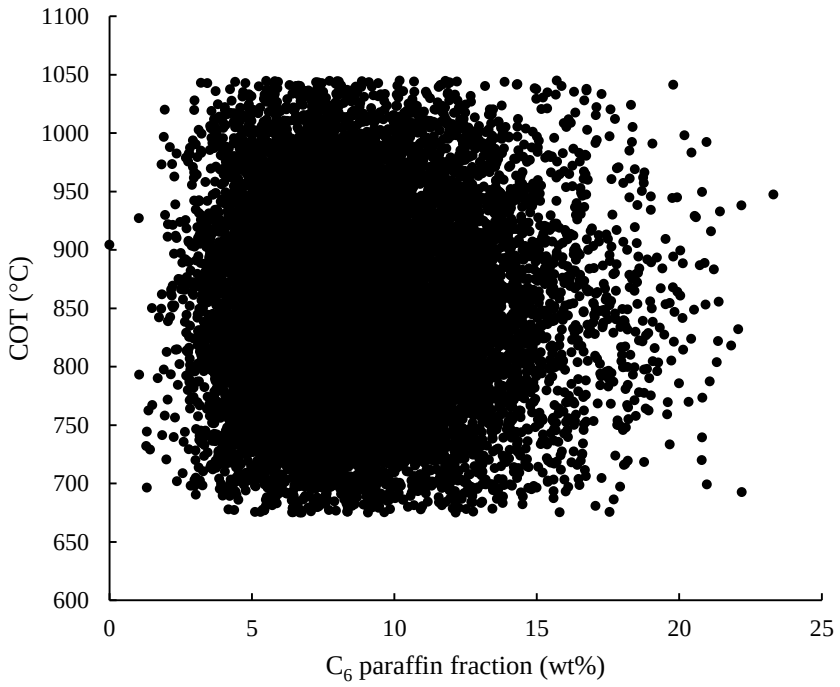

Fig. S14. Coverage of the C_6 paraffin fraction and coil outlet temperature data distribution.

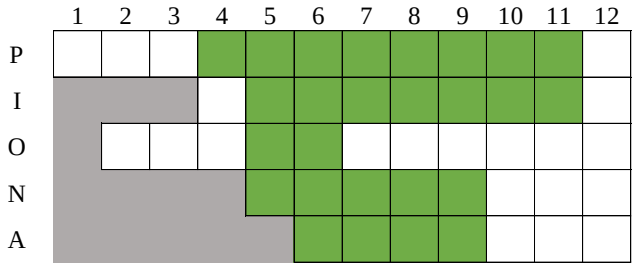

Fig. S15. PIONA matrix of components accounted for in the feedstock reconstruction.

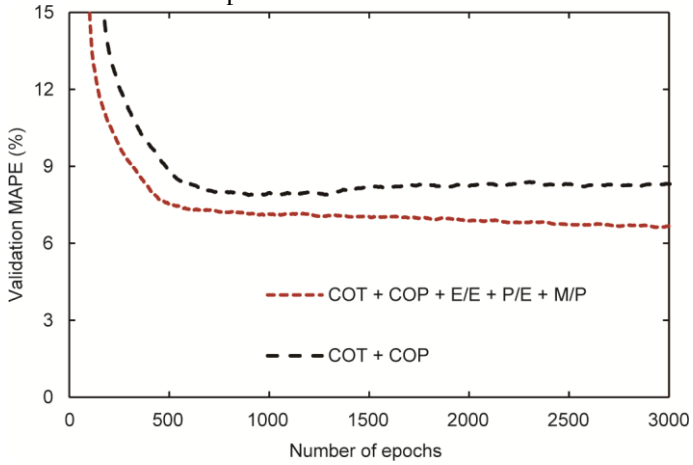

Fig. S16. Validation mean absolute percentage error (MAPE) of Network 3 when using two or five process descriptors as input. COT: coil outlet temperature; COP: coil outlet pressure; E/E: ethylene to ethane ratio; P/E: propylene to ethylene ratio; M/P: methane to propylene ratio.

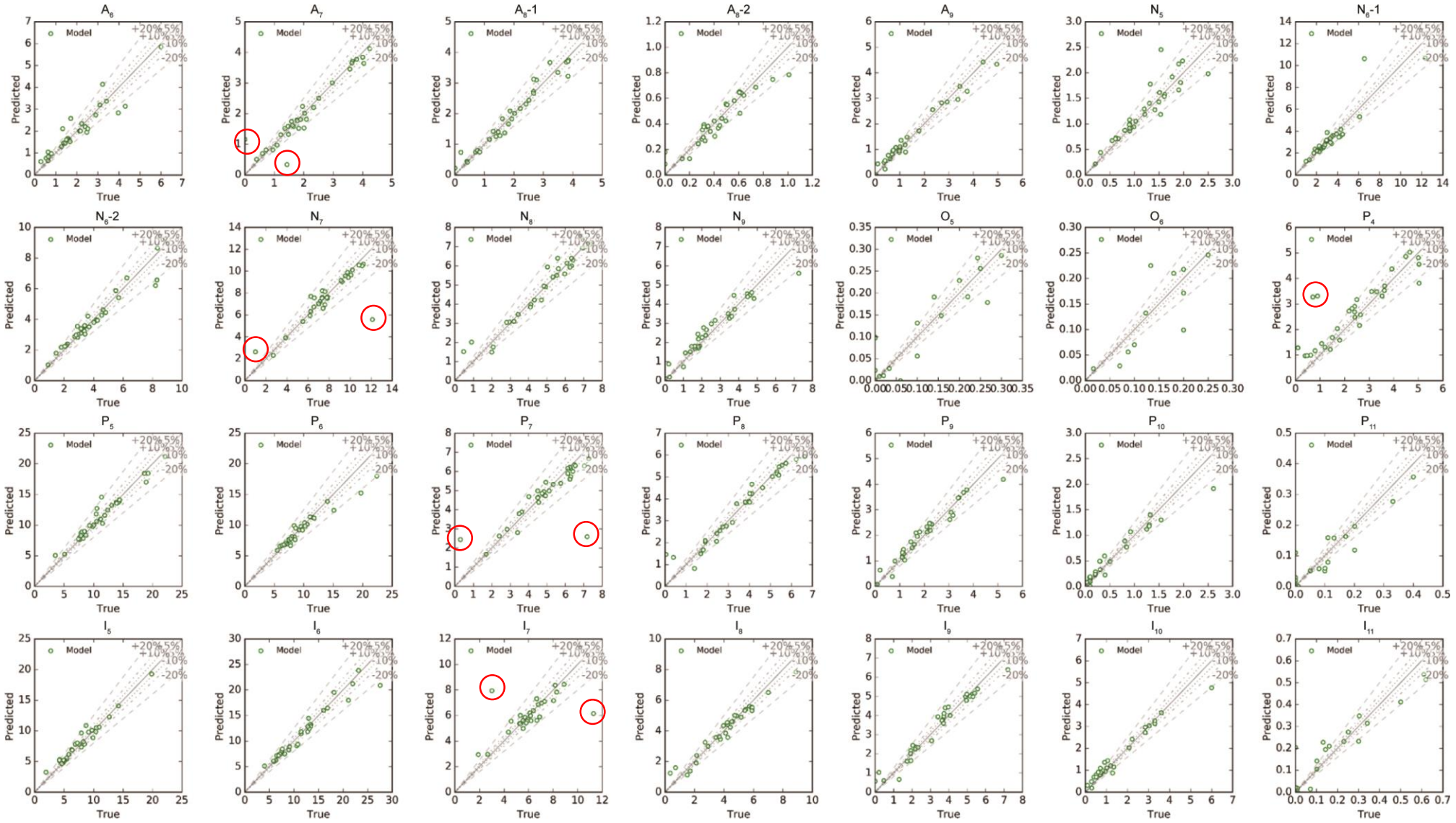

Fig. S17. Parity plots for all output component weight percentages of Network 2. The data points circled in red all correspond to the same two problematic naphthas.

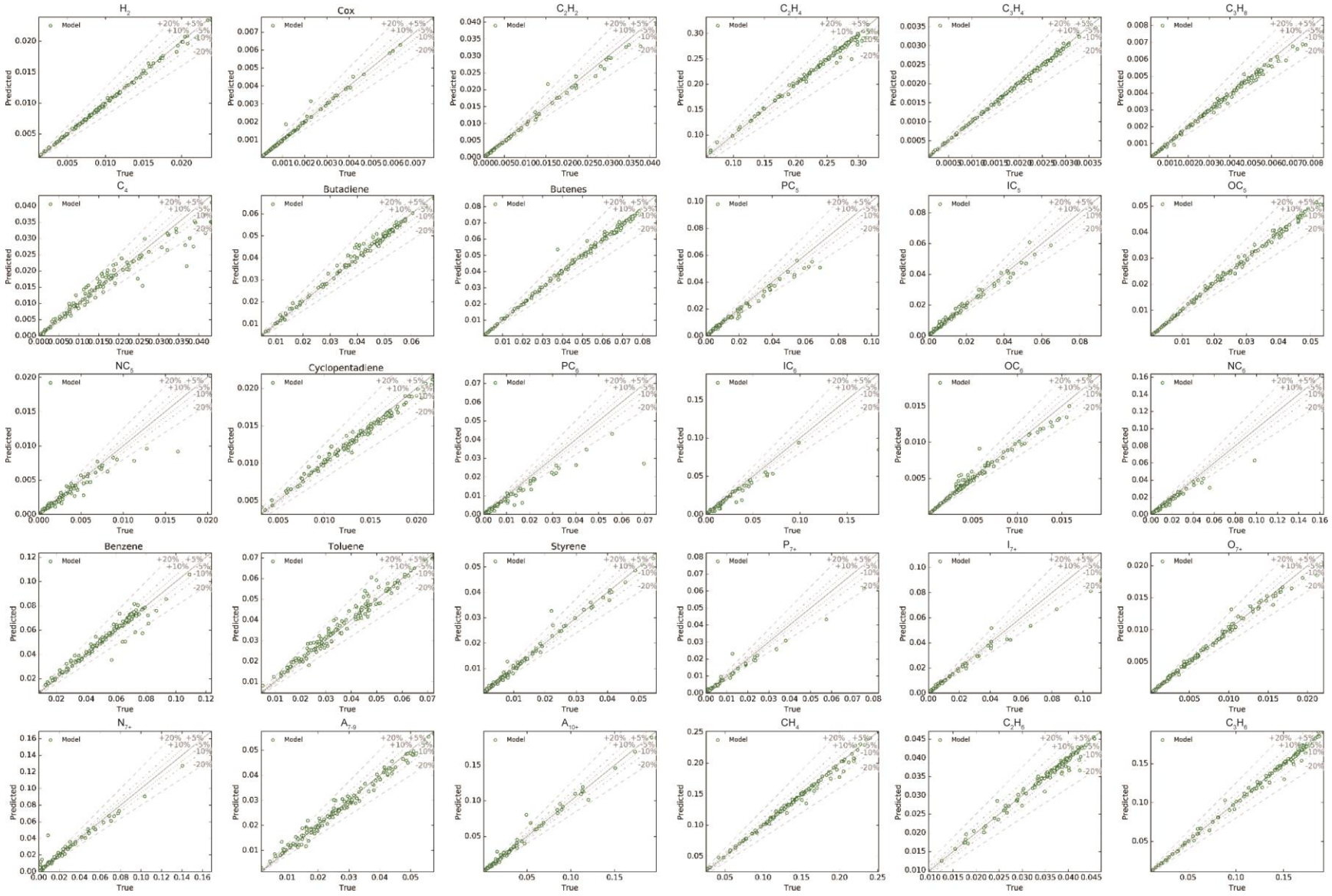

Fig. S18. Parity plots for all output component mass fractions of Network 3.

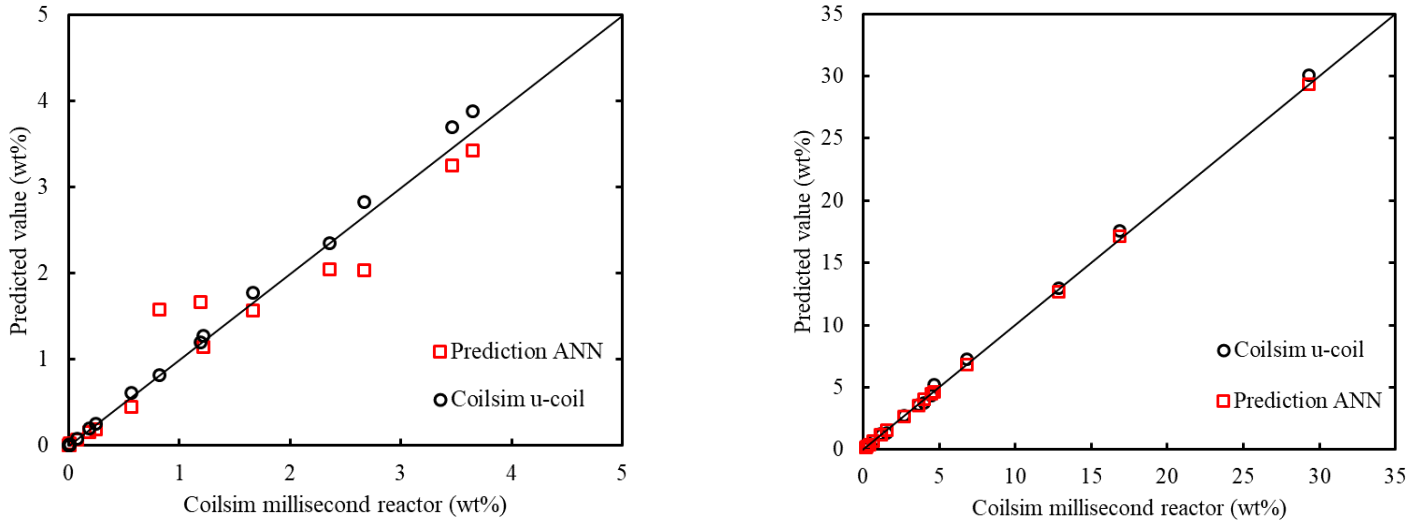

Fig. S19. Illustration of the independence of the reactor configuration of the simulations/predictions. (Left) COT: 950K, COP: 1.5 bara; (Right) COT: 1175K, COP: 2.0 bara. When fixing the reactor severity indices, the Coilsim® simulations are nearly identical for both the millisecond reactor and the u-coil reactor. Therefore, the artificial neural network (ANN) performance is (nearly) independent of the reactor type.

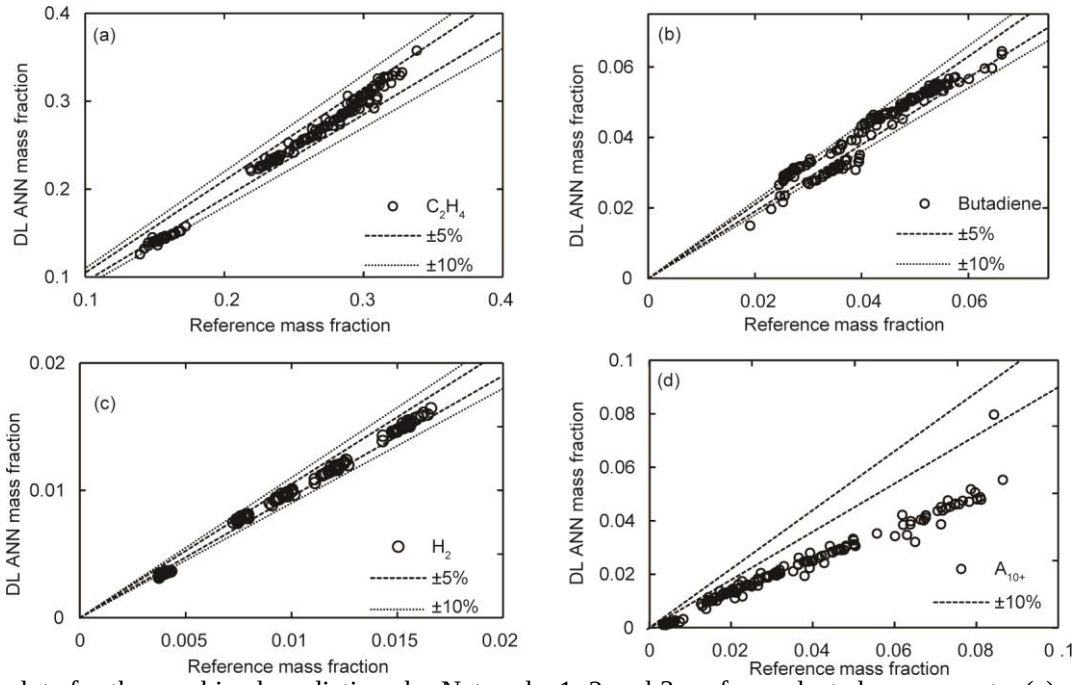

Fig. S20. Parity plots for the combined predictions by Networks 1, 2 and 3 on four selected components: (a) ethylene; (b) 1,3-butadiene; (c) hydrogen and (d) C_{10+} aromatics. 158 of the 1587 test set data points are displayed.

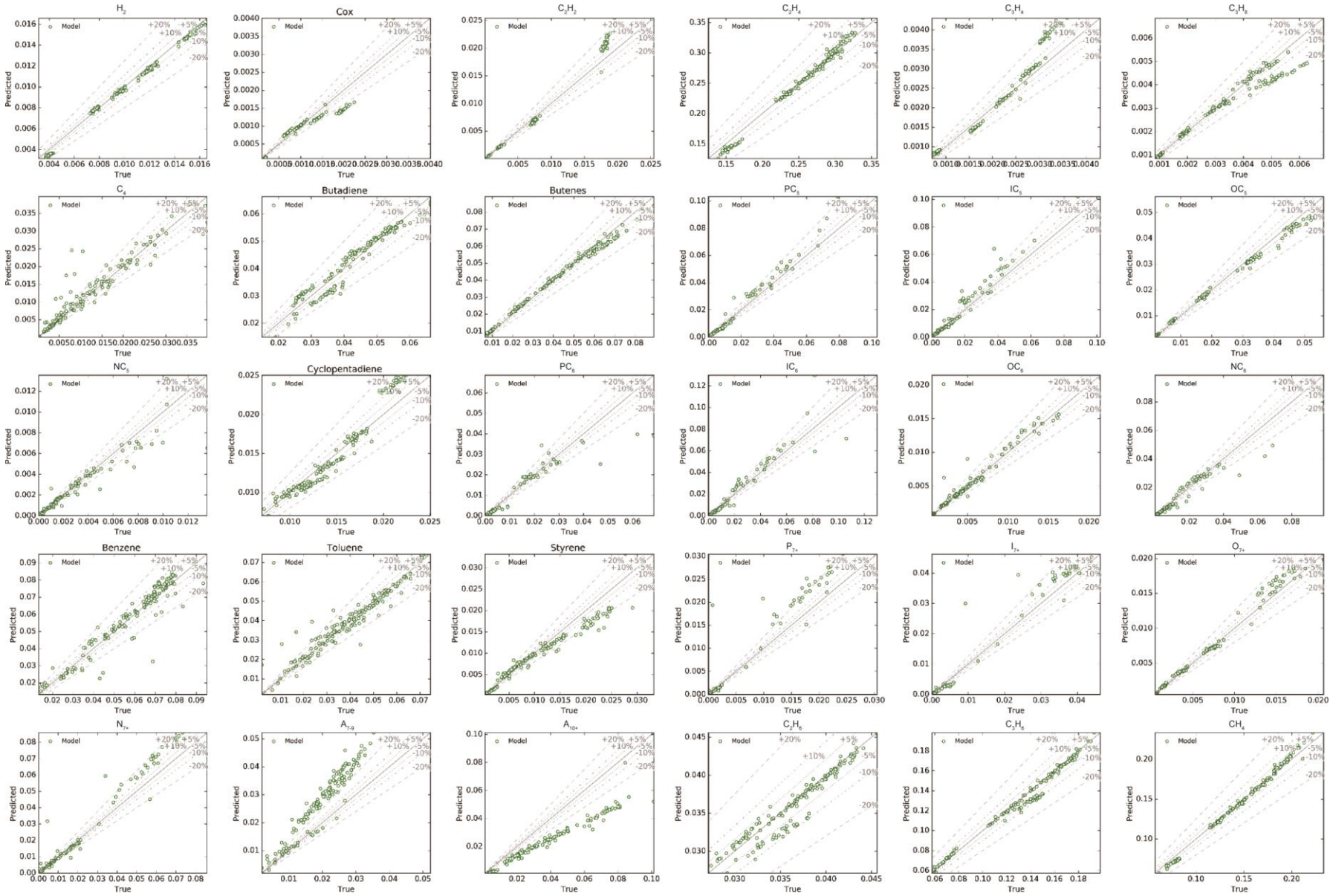

Fig. S21. Parity plots for all output component mass fractions of the combined prediction of Networks 1, 2 and 3.