Data-Centric Engineering (2020), 1: e13
doi:10.1017/dce.2020.14

RESEARCH ARTICLE

Numerical simulation, clustering, and prediction of multicomponent polymer precipitation

Pavan Inguva[1,2], Lachlan R. Mason[3], Indranil Pan[3,4], Miselle Hengardi[2] and Omar K. Matar[2,*]

[1]Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge 02139, Massachusetts, USA.
[2]Department of Chemical Engineering, Imperial College London, London SW7 2AZ, United Kingdom.
[3]Data Centric Engineering Program, The Alan Turing Institute, London NW1 2DB, United Kingdom.
[4]Centre for Environmental Policy, Imperial College London, London SW7 2AZ, United Kingdom.
*Corresponding author. E-mail: o.matar@imperial.ac.uk

Received: 16 July 2020; **Accepted:** 7 October 2020

Keywords: Classification; image analysis; morphology; polymer blend; prediction

Abstract

Multicomponent polymer systems are of interest in organic photovoltaic and drug delivery applications, among others where diverse morphologies influence performance. An improved understanding of morphology classification, driven by composition-informed prediction tools, will aid polymer engineering practice. We use a modified Cahn–Hilliard model to simulate polymer precipitation. Such physics-based models require high-performance computations that prevent rapid prototyping and iteration in engineering settings. To reduce the required computational costs, we apply machine learning (ML) techniques for clustering and consequent prediction of the simulated polymer-blend images in conjunction with simulations. Integrating ML and simulations in such a manner reduces the number of simulations needed to map out the morphology of polymer blends as a function of input parameters and also generates a data set which can be used by others to this end. We explore dimensionality reduction, via principal component analysis and autoencoder techniques, and analyze the resulting morphology clusters. Supervised ML using Gaussian process classification was subsequently used to predict morphology clusters according to species molar fraction and interaction parameter inputs. Manual pattern clustering yielded the best results, but ML techniques were able to predict the morphology of polymer blends with $\geq$90% accuracy.

Impact Statement

By providing predictive tools to polymer engineers that assist in predicting morphological features from easily knowable (or measurable) input parameters, the need to perform time-consuming and costly experiments to obtain desired morphological behaviors in such complex systems is reduced. This could reduce the overall complexity of the research and development (R&D) process for a variety of industries. Applying data-driven approaches to analyzing simulation data may also help to identify trends and features that are important but might not otherwise be detected by humans.

1. Introduction

Multicomponent polymer systems are of industrial interest in a variety of applications such as high-performance plastics (Nauman and He, 1994), membrane systems (Ulbricht, 2006; Yang et al., 2018), nanoparticle and nano-colloidal systems (Lee et al., 2017; Li et al., 2017), and drug delivery (Lao et al., 2008;

Inguva et al., 2015). One of the key features considered during the research and development (R&D) and/or manufacturing process is the morphology of the polymeric particles/blends formed. The morphology can profoundly impact the final product's performance and usability; as for many applications, there is an optimal morphology that is desired. Understanding the relationship between polymer properties and their resultant blend morphology will therefore help guide product development from synthesis to manufacturing steps.

Computational methods provide an excellent tool for modeling various phenomena across various scales in polymeric systems (Gooneie et al., 2017). Modeling techniques can be applied to understand and evaluate a variety of thermodynamic and transport properties such as polymer-blend miscibility. They can also be used in engineering and manufacturing applications to understand how morphological structures form or how materials respond to processing conditions. On a molecular/atomic scale, techniques such as molecular dynamics have been widely used in polymer systems to evaluate properties such as the miscibility and interactions of polymers in a blend or with other species (Prathab et al., 2007; Luo and Jiang, 2010), diffusion coefficients and transport characteristics (Pavel and Shanks, 2005), composite elasticity (Han and Elliott, 2007), and nanoparticle morphology (Li et al., 2017). Recent, nonequilibrium molecular dynamics simulation studies have also considered the effect of shear on the morphology of anisotropic nanoparticles (Bianchi et al., 2015; Delacruz-Araujo et al., 2016) such as Janus nanoparticles which are an interesting case within the possibility space of multicomponent polymer systems.

Continuum-based techniques are typically applied at length and timescales orders of magnitude larger than discrete approaches. Phase-field models, such as the Cahn–Hilliard equation (Cahn and Hilliard, 1958), are useful in capturing the dynamic behavior of structures and morphologies in heterogenous systems. The Cahn–Hilliard model accounts for various thermodynamic driving forces for morphology evolution, such as homogenous free energy and interfacial energy, and can also be adapted to further account for other relevant transport phenomena such as convection (Wodo and Ganapathysubramanian, 2012). Previous continuum-scale simulations of multicomponent polymer systems have focused "uphill" diffusion, as described by the Cahn–Hilliard equation. Examples of previous applications include systems of two polymers and one solvent (denoted polymer–polymer–solvent [PPS]) (Alfarraj and Nauman, 2007; Shang et al., 2011), or ternary polymer systems (Nauman and He, 1994; Alfarraj and Nauman, 2007). Studies that have considered the influence of convective transport in multicomponent systems have only evaluated a single polymer species precipitating out of solution (Zhou and Powell, 2006; Tree et al., 2017). Consequently, there is a knowledge gap in the understanding of how systems containing more than one polymer, that is, polymer-polymer-solvent (PPS) and polymer-polymer-polymer (PPP) systems, behave in the presence of convective mass transport.

Machine learning (ML) has increasingly found use in physical simulations and computational modeling by complementing or replacing traditional modeling approaches. ML is noted in having strength in pattern recognition and data mining (Brunton et al., 2020), which correspondingly enables it to be used for many tasks relevant to physical simulations. Previous studies have applied ML to develop data-driven surrogate/reduced order models (ROMs) to improve the speed of computations and results generation (Peherstorfer et al., 2017; Janet et al., 2018; San and Maulik, 2018). ML has also been used to improve the accuracy of physical simulations by enabling the development of data-driven closures (Duraisamy et al., 2019) and models (Chmiela et al., 2017), which can capture more data than traditional first principles or empirical models. Within the material sciences, ML has similarly found increasing use in a variety of cases ranging from the prediction of macroscopic self-assembled structures using molecular properties (Inokuchi et al., 2018) to the prediction of novel permanent magnets (Möller et al., 2018) and in the optimization of alloy properties (Ward et al., 2018).

In polymer science specifically, ML has been used to optimize polymer-gel screening for injection wells (Aldhaheri et al., 2017) and solar cells (Jørgensen et al., 2018), to improve polymeric interfacial compatibilization (Meenakshisundaram et al., 2017), and to classify and predict the physical features of polymer systems. For instance, supervised feed-forward neural networks have been used to recognize configurations produced from Monte Carlo simulations of polymer models, distinguishing between differently ordered states (Wei et al., 2017). Self-folding mechanisms of polymer composite systems have also been modeled (Guo et al., 2013). There is however a knowledge gap in the use of ML as a tool for classifying and predicting polymer-blend morphology. ML techniques are well suited to this end as they are able to learn complex features from the data, which facilitates pattern recognition and dimensionality reduction (Cai et al., 2018).

Previous work in the broader material science field has typically considered only one aspect of the pipeline such as employing dimensionality reduction to identify important features that contribute to the outer structure of nanoparticles (López-Donaire et al., 2012) or applying supervised ML for classification of new carbon black samples (Fernandez Martinez et al., 2017). The present study hence applies a ML workflow, comprising the use of dimensionality reduction techniques with a clustering algorithm, to separate morphological data into clusters of distinct morphologies. Pattern prediction and design-space mapping are investigated via classification, using Gaussian process classification (GPC) techniques, in the low-dimensional transformed feature space. The present analysis is restricted to PPP systems and does not consider the effects of convective mass transfer. The value of the workflow developed in this study is twofold: (a) the approach is generalizable to additional engineering fields beyond polymer science; and (b) polymer blends can be studied more expediently as regions of interest (input parameters/morphology) can be first identified which allows resources (computational/experimental) to be focused.

2. Theory and Methodology

We address the problem set where physics-based simulation outputs need to be predicted from known input parameters: a workflow for integrating physics-based simulations and ML clustering is outlined in Figure 1.

2.1. Physics-based Cahn–Hilliard system

The Cahn–Hilliard equation is well suited for modeling polymer-blend precipitation at continuum length and timescales. Three different polymer species a, b, c are tracked; however, the following derivation is kept as general as possible to demonstrate the applicability of the model to n component mixtures. We use a modified Cahn–Hilliard system based on the work of (Petrishcheva and Abart, 2012), which has the advantage of being able to handle mixtures where the components may have orders of magnitude difference in diffusivities. The ability of the model to handle components with large differences in diffusivity is important in modeling systems with polymers and solvents or polymers of significantly different chain lengths (Alfarraj and Nauman, 2007).

The model used in the present work is also easier to implement than the earlier method of Alfarraj and Nauman (2007) for handling components with large difference in diffusivity. Alfarraj and Nauman (2007) introduced a “proportional flux method” within a finite difference scheme to ensure that the sum of fluxes into a point is zero. The proportional flux method introduces two issues: (a) the method itself is an heuristic solution without a robust theoretical foundation (Nauman and Savoca, 2001) and (b) many modern partial differential equation (PDE) solvers, e.g. FEniCS (Logg et al., 2012) and FiPy (Guyer et al., 2009), are primarily declarative, and ad hoc adjustments to standard discretizations are difficult to implement.

The Cahn–Hilliard equation, as previously mentioned, models uphill diffusion where the driving force is gradients in the chemical potential μ rather than concentration. Correspondingly, the flux $\boldsymbol{j}_i$ of species i can be represented as

$$\boldsymbol{j}_i = -\sum_j L_{ij} \nabla \mu_j, \tag{1}$$

where each L_{ij} is the species mobility coefficient and $\boldsymbol{L}$ is the square symmetric. The following constraints are imposed on the flux expression due to the Onsager reciprocal relations and that the total flux into a point is zero (Petrishcheva and Abart, 2012):

$$L_{ij} = L_{ji}, \quad \sum_i L_{ij} = 0, \quad \sum_i \boldsymbol{j}_i = \boldsymbol{0}. \tag{2}$$

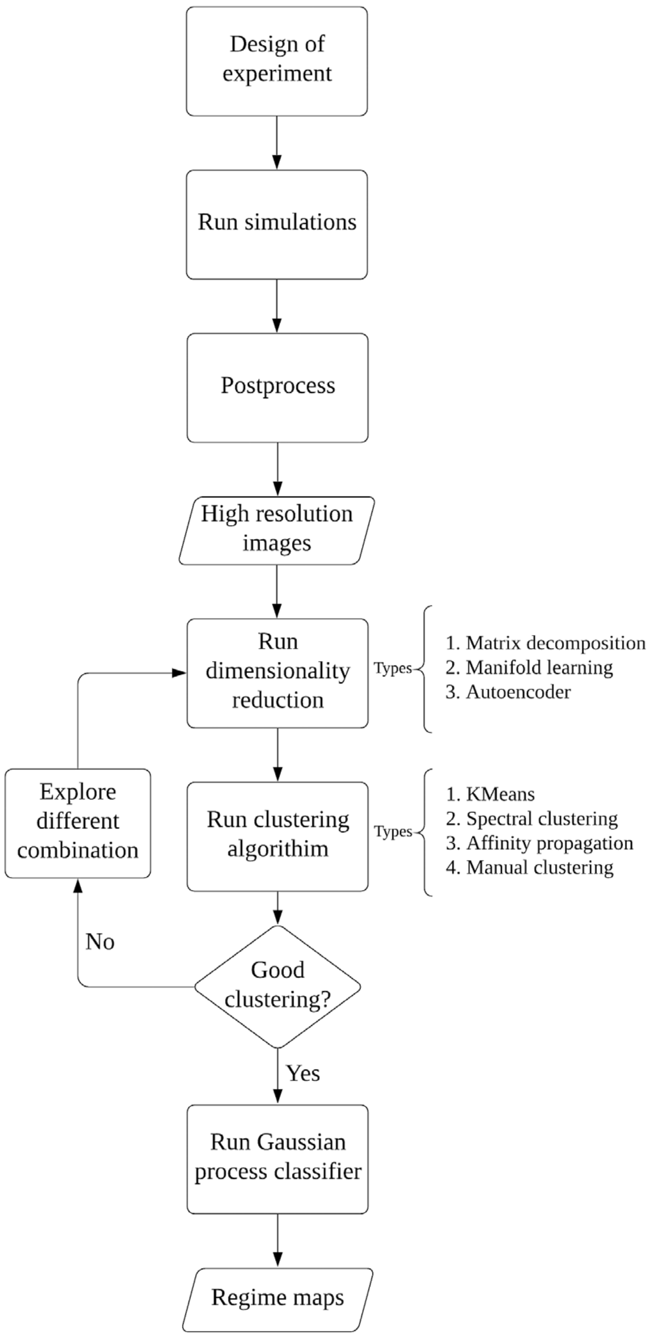


Figure 1. *Workflow for integrating the physics-based simulation set with machine learning (ML) dimensionality reduction, clustering, and prediction algorithms.*

Correspondingly as per Petrishcheva and Abart (2012), $\boldsymbol{j}_i$ can be expressed in terms of differences in chemical potentials,

$$\boldsymbol{j}_i = \sum_j L_{ij} \nabla \left(\mu_i - \mu_j \right). \tag{3}$$

As the Gibbs energy functional is scaled by RT, L_{ij} can be expressed with the following relationship:

$$L_{ij} = -D_{ij} x_i x_j, \tag{4}$$

where x_i and x_j are the mole fractions of species i and j, respectively. The mobility coefficients L_{ij} are composition dependent, but the D_{ij} effective diffusion coefficients can be constant.

To obtain the relevant transport equation for each species, we apply the continuity equation,

$$\frac{\partial x_i}{\partial t} + \nabla \cdot \boldsymbol{j}_i = 0, \tag{5}$$

where t is the time. To determine an expression for the chemical potential, a generalized Landau–Ginzburg free energy functional for N components, G_{system}, which accounts for inhomogeneity in the system is first considered (Cahn and Hilliard, 1958; Nauman and Balsara, 1989),

$$\frac{G_{\text{system}}}{RT} = \int_V \left[g(x_1, x_2, \ldots, x_N) + \sum_i^{N-1} \frac{\kappa_i}{2} (\nabla x_i)^2 + \sum_{j>i} \sum_i^{N-1} \kappa_{ij} (\nabla x_i)(\nabla x_j) \right] \mathrm{d}V, \tag{6}$$

where g is the homogenous free energy contribution, and κ_i and κ_{ij} are the self- and cross-gradient energy parameters, respectively. To evaluate whether the simulation has reached an equilibrium state, the Gibbs free energy was determined at each time step as per Equation (6). For polymeric systems, the homogenous free energy is well represented by the Flory–Huggins equation,

$$\frac{g(x_1, x_2, \ldots, x_N)}{RT} = \sum_i^N \frac{x_i}{n_i} \ln x_i + \sum_{j>i} \sum_i^{N-1} \chi_{ij} x_i x_j, \tag{7}$$

where n_i is the polymer chain length and χ_{ij} is the Flory–Huggins binary interaction parameter. The generalized chemical potential, applicable for inhomogenous systems, for each species i can be expressed as the variational derivative of the Gibbs energy functional (Nauman and Balsara, 1989; Cogswell, 2010),

$$\mu_i = \frac{\delta G_{\text{system}}}{\delta x_i} = \frac{\partial G}{\partial x_i} - \nabla \cdot \frac{\partial G}{\partial \nabla x_i}. \tag{8}$$

We replace x_i with a, b, c to represent the mole fractions of the three species of interest: A, B, and C, respectively. We can write the following equations for the differences in chemical potentials:

$$\mu_{\text{AB}} = \mu_{\text{A}} - \mu_{\text{B}} = \frac{\partial g}{\partial a} - \frac{\partial g}{\partial b} - (\kappa_{\text{A}} - \kappa_{\text{AB}}) \nabla^2 a + (\kappa_{\text{B}} - \kappa_{\text{AB}}) \nabla^2 b, \tag{9}$$

$$\mu_{\text{AC}} = \mu_{\text{A}} - \mu_{\text{C}} = \frac{\partial g}{\partial a} - \frac{\partial g}{\partial c} - \kappa_{\text{A}} \nabla^2 a - \kappa_{\text{AB}} \nabla^2 b, \text{ and} \tag{10}$$

$$\mu_{\text{BC}} = \mu_{\text{B}} - \mu_{\text{C}} = \frac{\partial g}{\partial b} - \frac{\partial g}{\partial c} - \kappa_{\text{B}} \nabla^2 b - \kappa_{\text{AB}} \nabla^2 a. \tag{11}$$

The gradient energy parameters for the PPP system (Nauman and He, 1994) can be evaluated as follows. We specifically consider the case of all polymer species having the same radius of gyration R_{G} and diffusivity,

$$\kappa_{\text{A}} = \frac{2}{3} R_{\text{G}}^2 \chi_{\text{AC}}, \tag{12}$$

$$\kappa_{\text{B}} = \frac{2}{3} R_{\text{G}}^2 \chi_{\text{BC}}, \text{ and} \tag{13}$$

$$\kappa_{\mathrm{AB}} = \frac{1}{3} R_{\mathrm{G}}^2 (\chi_{\mathrm{AC}} + \chi_{\mathrm{BC}} - \chi_{\mathrm{AB}}). \tag{14}$$

The compositional dependence of κ_i and κ_{ij} is neglected following an approach commonly used by similar simulation studies (Nauman and He, 1994; Zhou and Powell, 2006; Alfarraj and Nauman, 2007). This also simplifies the computations.

Combining the expressions for the chemical potential Equations (9)–(11), species flux Equations (3)–(4), and the continuity Equation (5), we arrive at the following transport equations tracking species A and B:

$$\frac{\partial a}{\partial t} = \nabla \cdot (D_{\mathrm{AB}} ab \nabla \mu_{\mathrm{AB}} + D_{\mathrm{AC}} ac \nabla \mu_{\mathrm{AC}}), \text{ and} \tag{15}$$

$$\frac{\partial b}{\partial t} = \nabla \cdot (-D_{\mathrm{AB}} ab \nabla \mu_{\mathrm{AB}} + D_{\mathrm{BC}} bc \nabla \mu_{\mathrm{BC}}). \tag{16}$$

Species C is obtained by using a material balance constraint,

$$c = 1 - a - b. \tag{17}$$

For symmetric PPP systems, where all species diffusivities can be assumed equal, the equation system given by Equations (15)–(17) reduces to that considered by Nauman and He (1994).

2.1.1. *Scaling*

We introduce the following scalings:

$$\boldsymbol{x} = d_{\mathrm{p}} \tilde{\boldsymbol{x}}, \text{ and} \tag{18}$$

$$t = \frac{n d_{\mathrm{p}}^2}{D_{\mathrm{AB}}} \tilde{t}, \tag{19}$$

where d_{p} is the characteristic length scale. The chemical potential and Gibbs energy functional are scaled by RT (denoted from here on as $\tilde{\mu}_i$ and $\tilde{g}$, respectively). We consider the case of a symmetric PPP system i.e. all the polymer species have the same chain length and diffusivity, thus resulting in the following equation system:

$$\tilde{\mu}_{\mathrm{AB}} = \tilde{\mu}_{\mathrm{A}} - \tilde{\mu}_{\mathrm{B}} = \frac{\partial \tilde{g}}{\partial a} - \frac{\partial \tilde{g}}{\partial b} - (\tilde{\kappa}_{\mathrm{A}} - \tilde{\kappa}_{\mathrm{AB}}) \tilde{\nabla}^2 a + (\tilde{\kappa}_{\mathrm{B}} - \tilde{\kappa}_{\mathrm{AB}}) \tilde{\nabla}^2 b, \tag{20}$$

$$\tilde{\mu}_{\mathrm{AC}} = \tilde{\mu}_{\mathrm{A}} - \tilde{\mu}_{\mathrm{C}} = \frac{\partial g}{\partial a} - \frac{\partial g}{\partial c} - \tilde{\kappa}_{\mathrm{A}} \tilde{\nabla}^2 a - \kappa_{\mathrm{AB}} \tilde{\nabla}^2 b, \tag{21}$$

$$\tilde{\mu}_{\mathrm{BC}} = \tilde{\mu}_{\mathrm{B}} - \tilde{\mu}_{\mathrm{C}} = \frac{\partial g}{\partial b} - \frac{\partial g}{\partial c} - \tilde{\kappa}_{\mathrm{B}} \tilde{\nabla}^2 b - \tilde{\kappa}_{\mathrm{AB}} \tilde{\nabla}^2 a, \tag{22}$$

$$\frac{\partial a}{\partial t} = \tilde{\nabla} \cdot \left(ab \tilde{\nabla} \tilde{\mu}_{\mathrm{AB}} + ac \tilde{\nabla} \tilde{\mu}_{\mathrm{AC}} \right), \text{ and} \tag{23}$$

$$\frac{\partial b}{\partial t} = \tilde{\nabla} \cdot \left(-ab \tilde{\nabla} \tilde{\mu}_{\mathrm{AB}} + bc \tilde{\nabla} \tilde{\mu}_{\mathrm{BC}} \right). \tag{24}$$

Equations (20)–(24) model the polymer-blend demixing dynamics and form the final set of equations to be solved.

2.1.2. Numerical implementation

Numerical solution of the Cahn–Hilliard equation is challenging due to the fourth-order derivative: hence, the equation is typically treated as a set of coupled second-order PDEs (Jokisaari et al., 2017) as shown in the equation system of Equations (20)–(24). The system was reformulated to variational form and solved with an open-source finite-element solver, FEniCS (Logg et al., 2012).

An unstructured square mesh of domain-length 40 dimensionless units was generated with a resolution of 80 cells along each domain boundary. Periodic boundary conditions were applied on the left and right boundaries and Neumann conditions were applied to the top and bottom domains. Unknown variables were treated implicitly and a backward Euler method was applied for time discretization. PPP cases were simulated for a duration of $\widetilde{t} = 400$ with a time step of $\Delta\widetilde{t} = 0.02$, corresponding to a physical duration of $t = 16\,\mathrm{s}$. Physical parameters were set to $R_{\mathrm{G}} = 200 \times 10^{-10}\,\mathrm{m}$ and $d_{\mathrm{p}} = R_{\mathrm{G}}$, $D_{\mathrm{AB}} = 10^{-11}\,\mathrm{m^2 s^{-1}}$. Values of χ_{ij} simulated ranged from 0.003 to 0.009 via automated batch scripting (Di Tommaso et al., 2017). In total, 1,140 simulation runs were performed, representing a total of five independent input parameters. The entire simulation process took ~80,000 core-hours. Benchmarking and model validation are discussed in the supplementary material.

2.2. Machine learning

2.2.1. Methodology and workflow

The ML workflow consisted of three steps: (a) dimensionality reduction, (b) clustering, and (c) supervised learning. Dimensionality reduction is required to address the curse of dimensionality (Kriegel et al., 2009), which can make clustering of high-dimensional data, such as raw simulation images, prohibitively expensive. Clustering on the low-dimensional processed data was used to identify morphologies. Supervised ML, using GPC, was then used to determine a relationship between the physical input parameters and the resultant morphology.

2.2.2. Dimensionality reduction

Each simulation generates a high-resolution color image of the physical morphology, where RGB image channels form a proxy for species molar fraction (i.e., subject to the material balance of Equation (17)). Prior to dimensionality reduction, each image was preprocessed to 200×200 pixel resolution. The dimensionality reduction itself was applied using three candidate techniques: (a) principal component analysis (PCA), (b) t-distributed stochastic neighbor embedding (t-SNE), and (c) autoencoder compression (Wang et al., 2016). The three techniques used are representative of the three broad categories of techniques for dimensionality reduction available: (a) matrix decomposition or linear techniques, (b) manifold learning or nonlinear techniques, and (c) artificial neutral network (ANN)-based techniques. PCA and t-SNE were implemented using *scikit-learn* (Pedregosa et al., 2011), while the autoencoder was implemented using *Keras* (Chollet et al., 2018).

For PCA and t-SNE pipelines, the set of species concentration fields was extracted as three grayscale images (200×200); image arrays were then flattened and concatenated into a one-dimensional array of length 120,000 ($3 \times 200 \times 200$) representing each simulation result. For the autoencoder, the preprocessed color images were used as direct inputs.

Principal component analysis (PCA). For a set of input arrays, the q principal components (PCs) form the orthonormal axes onto which the retained variance under projection is maximal. The PCA pipeline within *scikit-learn* was undertaken using the probabilistic model of Tipping and Bishop (1999): the number of retained PCs, q, was varied between 0 and 100, corresponding to the dimensionality of the embedding.

t-distributed stochastic neighbor embedding (t-SNE). Nonlinear dimensionality reduction was undertaken in *scikit-learn* using the t-SNE technique (Maaten and Hinton, 2008; Wattenberg et al., 2016). The number of embedding dimensions was varied from 2 to 10, while the perplexity was varied from 5 to 50.

Autoencoder compression. An autoencoder (Lecun et al., 1998) is a specific type of ANN that consists of two sections: (a) an encoder that compresses high-dimensional data to low-dimensional “bottleneck” representation and (b) a decoder that recovers the original data from the encoded data (Hinton, 2006). Autoencoders are highly suitable for dimensionality reduction, with performance often exceeding alternative techniques such as PCA (Hinton, 2006; Wang et al., 2016). Autoencoders are ideal for image analysis applications (Chen et al., 2017); moreover, the hidden-layer nodes at the ANN bottleneck can be exploited for downstream clustering pipelines.

Autoencoder architectures can be labeled using a convention *T*–*N*, where *T* denotes the layer type and *N* is the number of layers. Layer types explored presently include (a) densely connected (denoted “Dense”) and (b) convolutional (denoted “Conv”). The simplest autoencoder architectures consist of an input layer and an output layer with one or more Dense layers in a stacked fashion: Dense–1 has a single dense encoder layer which also serves as the bottleneck, while Dense–2 and Dense–3 contain additional layers. More complex architectures, such as LeNet–5 (Lecun et al., 1998), apply a combination of Dense and Conv layers.

All candidate autoencoders were trained on the full preprocess image data set, enabling autoencoder compression to be implemented into the same workflow, Figure 1, as PCA and t-SNE dimensionality reduction. Due to the small data set size, the conventional split into training and validation sets (Petscharnig et al., 2017; Chen and Huang, 2019) for evaluating clustering accuracy was not undertaken. A summary of all explored autoencoder types and hyperparameters is given in the supplementary material. Hyperparameter optimization for the dense autoencoders was performed using Talos Autonomio (2019) (fractional random search over 10–15% of the grid; 10 epochs), while convolutional autoencoders were tuned manually.

2.2.3. Clustering

To implement a purely unsupervised learning pipeline, the clustering method and/or use of an appropriate metric, heuristic, or algorithm should be able to estimate the optimal number of clusters. *Scikit-learn* implements a variety of clustering algorithms, such as affinity propagation, spectral clustering, hierarchical clustering, and *k*-means, which can be used to this end.

The popular clustering algorithm *k*-means as implemented in *scikit-learn* was used to carry out clustering as a means of measuring “sign of life.” Sign-of-life in this case refers to there being initially positive results from a naive application of a clustering algorithm which then justifies further exploration. To this end, *k*-means was found to be as effective as any of the clustering algorithms outlined above and was used as the default clustering algorithm for all the embeddings generated by the various dimensionality reduction techniques.

The *k*-means algorithm works by clustering the datapoints into *k* groups by minimizing the within-cluster sum of squares (WCSS) (Yuan and Yang, 2019). The algorithm requires an initialization method (set to “k-means++”) and a predetermined number of clusters, *k*. Evaluating an appropriate number for *k* is one of the main challenges when using *k*-means and the elbow method was used. The elbow method involves running *k*-means for a range of *k* values and calculating the distortion or the total sum of square errors (Yuan and Yang, 2019). Ideally, when the number of clusters nears the real number of clusters, there is an inflexion in the distortion, showing as an “elbow” in a plot of distortions vs. *k*.

For the clustering of the t-SNE embedding of the Conv–3/4/5 autoencoder bottleneck values, *k*-means was ill-suited as it is unable to effectively identify clusters where the cluster shape is non-globular. In this case, affinity propagation was used (Frey and Dueck, 2007). Affinity propagation does not require the declaration of the number of clusters to perform clustering, but rather it performs clustering based on passing messages between the datapoints which represents the fitness of one sample to exemplify the other until a set of exemplars are identified, representing the final number of clusters (Frey and Dueck, 2007). The implementation of affinity propagation in *scikit-learn* was used and it has two main hyperparameters: the preference which refers to the strength of a datapoint to be an exemplar and the damping ratio which was set to .9 for stability.

2.2.4. Supervised learning

A *scikit-learn* GPC model was trained on the initial concentrations (a_0, b_0) and corresponding cluster label. A test set (20% of the data) was used to check if the classifier returned the correct cluster, given the specific parameters. Radial basis functions (RBFs) with a length scale of 1.0 were selected for the Gaussian process kernel. The data augmentation process is described in the results and discussion section. To perform the prediction, the initial species concentrations (a_0, b_0) were varied within a specific range ($a_0 \in [0.1, 0.8]$, $b_0 \in [0.1, 0.45]$) and passed into the trained GPC to evaluate the morphology maps as shown in Figure 10.

3. Results and Discussion

3.1. Polymer demixing simulation results

The numerical stability of Cahn–Hilliard solution methods is a known issue, especially when using the Flory–Huggins free-energy function at higher χ_{ij} values (Brunswick et al., 1998). Three possible simulation states were identified as shown in Table 1: simstate demonstrating the impact of numerical issues on the solution of the physical model, either the simulation would diverge prematurely due to numerical instability (State 3a), or even though the input parameters were selected such that the physical system is in the chemical spinodal, no demixing would occur (State 2). The present data set of 629 images was restricted to samples from States 1 to 3b. A representation of how the Gibbs energy evolved with time for each of the three cases is shown in Figure 2.

Table 1. Different states that a simulation could take. Images from states 1 to 3b are used in the data set.

State identity (State ID)	Description
1	G_{system} decreased over time and appeared to taper off: solutions converged for the full simulation period $\tilde{t}$ and a pattern formed.
2	Solutions converged for the full simulation period $\tilde{t}$, but G_{system} appeared to remain constant: simulations with this State ID did not generate any patterns.
3a	G_{system} initially decreased and a pattern began to emerge; however, the simulation diverged and was terminated early: the pattern was unusable.
3b	G_{system} decreased and started to plateau with the simulation generating a usable pattern; however, the solution did not converge for the full simulation period $\tilde{t}$.

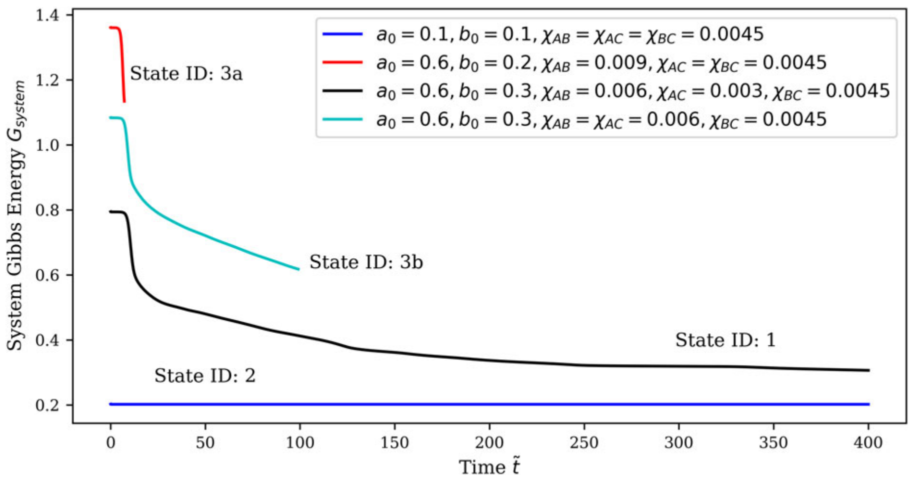


Figure 2. *Representative solutions for each Gibbs energy state. The state ID for each line is annotated next to the line for reference.*

3.2. Dimensionality reduction and clustering

The effectiveness of each dimensionality reduction and clustering technique is summarized in Figure 3: techniques were assessed.

3.2.1. Principal component analysis (PCA)

The image set could not be partitioned into distinct embedded-space clusters via PCA. As shown in Figure 4, the number of clusters evaluated by *k*-means consistently remained between 5 and 6 independent of the number of retained PCs, demonstrating that the application of PCA here is ineffective. PCA was not capable of learning distinguishing features of the system as a number of clusters, each with a distinct morphology, did not emerge.

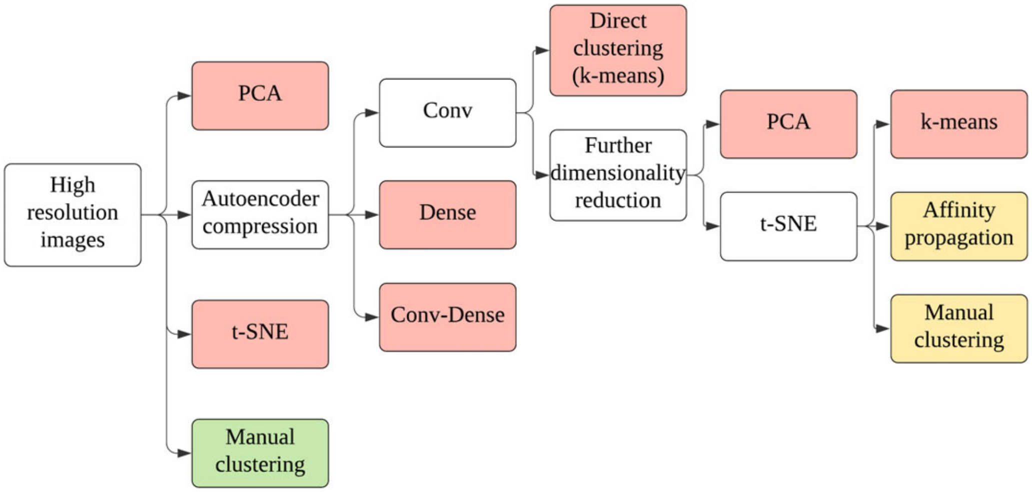


Figure 3. *Dimensionality reduction and clustering results—Red: Method is unable to yield useful results; Yellow: Method is able to yield results of some significance; however, the method is still inadequate; Green: Method that yielded the best results.*

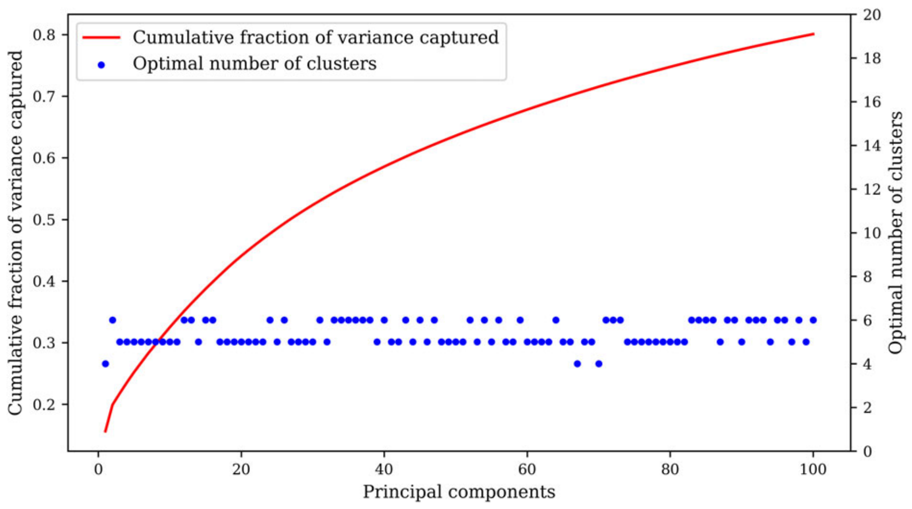


Figure 4. *Captured variance and optimal cluster number: the optimal number of clusters remained between 5 and 6 independent of the number of principal components (PCs) retained.*

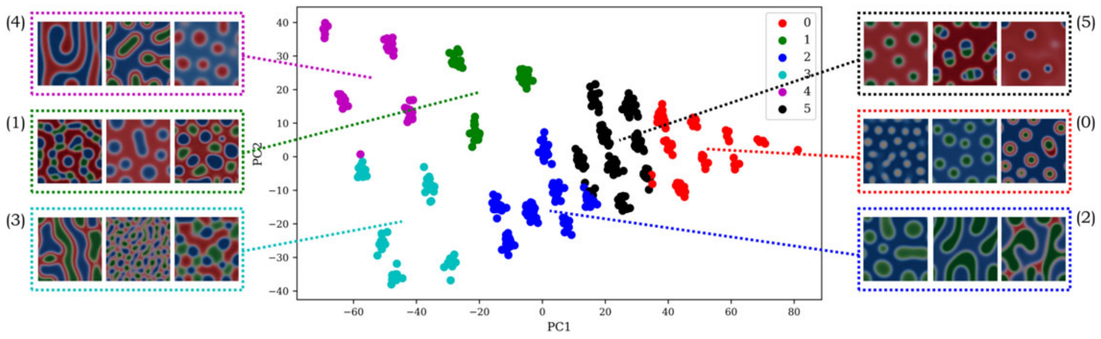


Figure 5. *Principal component analysis (PCA) dimensionality reduction (2 principal components retained) with k-means clustering (6 clusters). Sample images from each cluster are shown.*

Figure 5 shows how the clustering took place in two-dimensional (2D) space: each cluster contained a variety of morphologies and was therefore not distinct. Each cluster was, however, consistent in terms of the predominant continuous phase. This is evidenced in the sample images for each cluster. For example, cluster 0 contains globular dispersed-phase patterns; however, a mixture of core–shell (one component encapsulates the other), single-component (one component is present), and miscible (both components are mixed) patterns are present in the globular structure. A detailed description of the various observed morphologies and exemplar images is available in the supplementary material. Similar clustering behavior was observed when retaining higher numbers of PCs.

3.2.2. t-distributed stochastic neighbor embedding (t-SNE)

For t-SNE dimensionality reduction techniques, the variance in the optimal number of clusters was observed to increase with the number of embedding dimensions as shown in Figure 6. For almost all values of perplexity, the optimal number of clusters was 5 for two embedding dimensions and 7–8 for three embedding dimensions. A maximum in the number of clusters was observed for the combination of 7 embedding dimensions and perplexity 10.

Clustering in three or more embedding dimensions resulted in outlier clusters which contained only 1–2 datapoints (not shown). Clusters of outlying datapoints in the embedded space distorted the *k*-means process. The optimal number of clusters is also more sensitive to the perplexity values, which is reflected in the increasing variance in the number of optimal clusters. A visual inspection of the images from each cluster for sample cases revealed poorer clustering performance compared to PCA or t-SNE in two embedding dimensions with clusters having more variation in the types of morphologies and the species of the continuous phase present.

Use of two embedding dimensions resulted in more consistent clustering performance, with the optimal number of clusters mostly remaining at 5: a representative example can be seen in Figure 7. The dimensionality reduction and clustering performance was comparable to PCA; while there was significant mis-clustering within each cluster, each cluster was generally consistent with regards to the continuous-phase species present. Ultimately, both PCA and t-SNE techniques were unable to capture an adequate number of features to describe the diverse morphologies arising from ternary-polymer blends, prompting the exploration autoencoders as an alternative unsupervised ML workflow.

3.2.3. Autoencoders

The performance of each autoencoder architecture tested is shown in table: autoencoder together with reconstructed images and representative loss and accuracy values. Increasing the size and depth of the Dense autoencoders, which increases the number of tuneable parameters, did not improve the loss and accuracy values, which plateau at ~ 0.01 and ~ 0.6, respectively, for the optimal set of

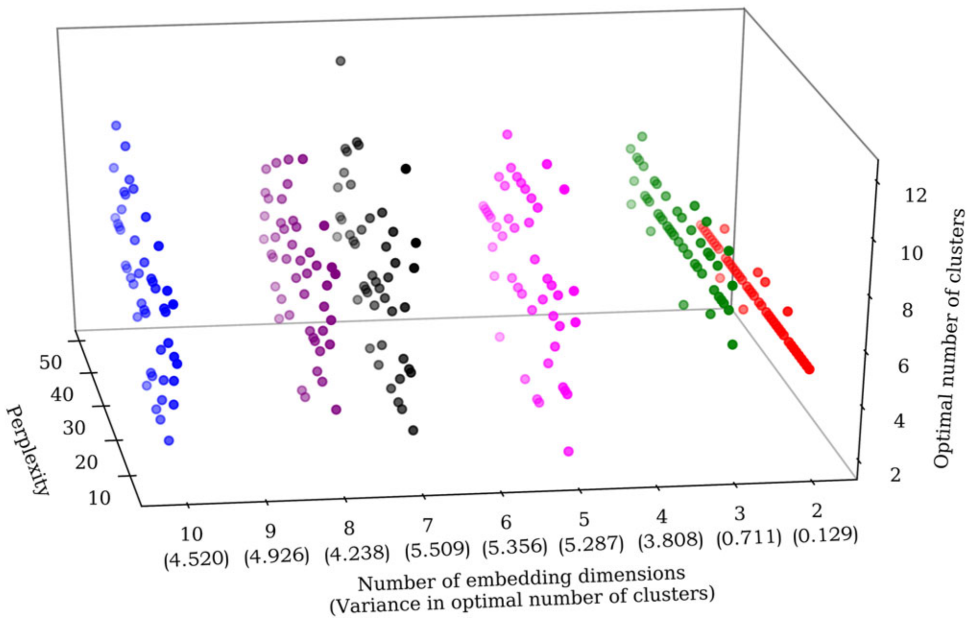


Figure 6. *Number of clusters as a function of number of embedding dimensions and perplexity. Configurations with 4, 6, and 9 embedding have been omitted for clarity. The variance in the optimal number of clusters is shown in parentheses below the x-axis. The variance generally increases with the number of embedding dimensions.*

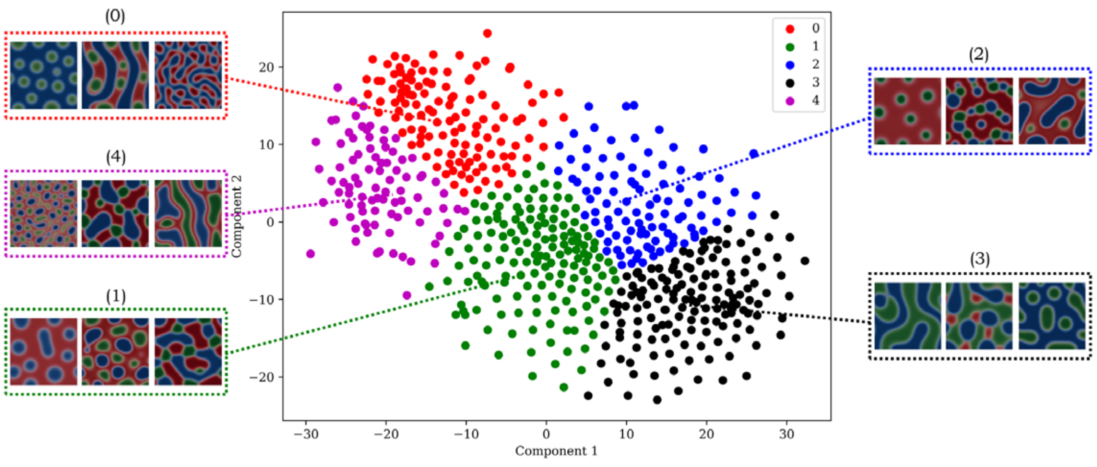
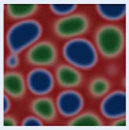
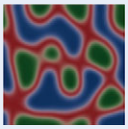
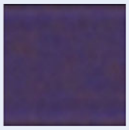
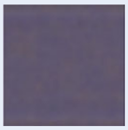
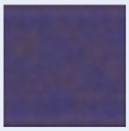
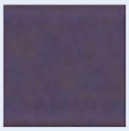
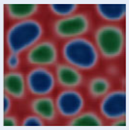
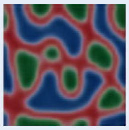
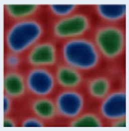
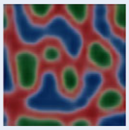
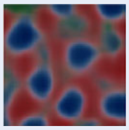
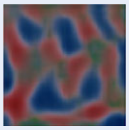


Figure 7. t-distributed stochastic neighbor embedding *(t-SNE) results in two-dimension (2D) with perplexity 30 and 5 clusters. Sample images from each cluster are shown. The clustering performance is similar to the results from using principal component analysis (PCA). There is a variety of different morphologies present in each cluster, but the species of the continuous phase is comparatively consistent.*

hyperparameters. The reconstructed image quality remains consistently poor. Autoencoders with Dense layers only generated poor embeddings of the images and were not further explored.

Conv autoencoder architectures performed significantly better than Dense architectures as evidenced by the reconstructed images in Table 2. The accuracy and reconstructed image quality increased with the number of bottleneck embedding dimensions. A dimensionality of $\gtrsim 500$ is necessary to reconstruct both

Table 2. Autoencoder performance for each architecture. Results for Dense–2 and Conv–Dense architectures have been omitted due to similarity with other Dense autoencoders. Dense and Conv–Dense autoencoders were observed to have lower accuracies and produce poorer image reconstructions than Conv autoencoders.

Autoencoder architecture	Encoding dimensionality	Loss / Accuracy	Sample 1	Sample 2
Original images	$(200, 200, 3) = 120,000$	0 / 1.0		
Dense–1	500	0.0111 / 0.5982		
Dense–1	5,000	0.0119 / 0.6000		
Dense–3	20	0.0204 / 0.6039		
Conv–3 (4 filters)	$(25, 25, 4) = 2,500$	4.5042×10^{-4} / 0.9654		
Conv–4 (4 filters)	$(13, 13, 4) = 676$	0.0012 / 0.9280		
Conv–5 (4 filters)	$(7, 7, 4) = 196$	0.0050 / 0.7949		

the morphology and continuous-phase species identity. A set of reconstructed images and loss/accuracy results for Conv–3, 4, 5 autoencoders is presented in the supplementary material for a range of bottleneck filter values. The addition of a Dense layer at the bottleneck of the Conv–Dense autoencoders increased the number of required parameters ($\geq 10^9$): hence, only Conv–4, 5 Dense–2, 1 autoencoders were tested due to their tractable memory requirements. Conv–Dense performance was comparable to Dense–1, 2, 3 architectures and was not further explored.

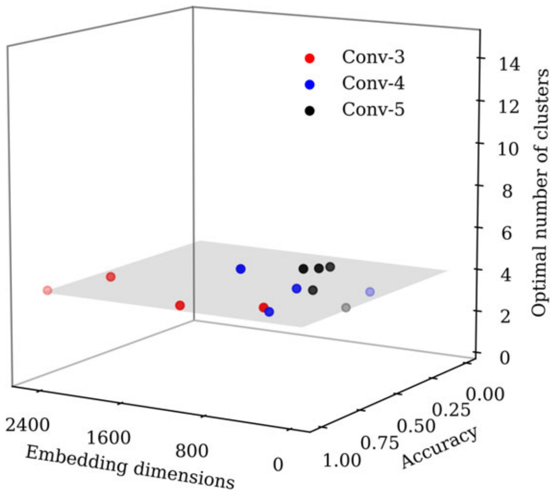


Figure 8. *k-means clustering on Conv–4 (4 filters) embedding (a plane of k = 4 clusters is shown for reference). Performing k-means clustering directly on the Conv autoencoder bottleneck values consistently resulted in 4–6 clusters.*

Clustering via *k*-means was performed directly on the Conv autoencoder bottleneck as shown in Figure 8. Performance was found to be comparable with the other dimensionality reduction techniques tested (PCA and t-SNE): clustering on the full embedding yielded a similar number of clusters as the previous approaches.

As the dimensionality of the bottleneck was comparatively high (~ 100—~ 1,000), stacking of additional dimensionality reduction techniques to further reduce the data dimensionality was performed (Maaten and Hinton, 2008). Applying PCA and retaining two PCs was not effective, yielding clustering performance comparable to applying PCA or t-SNE directly. Further dimensionality reduction was applied to the bottleneck values using t-SNE with a final two-dimensional embedding. The resulting datapoint distribution shown in Figure 9a almost exactly coincides with the composition a_0 and b_0: by following the "S" shape curve from the bottom and the value of a_0 increases. This result was consistent across the various Conv autoencoders and the results from the combination of the Conv–4 (4 Filters) bottleneck and t-SNE is presented in Figure 9.

Clustering via *k*-means techniques is ill-suited due to the shape of the embedded data shown in Figure 9b,c: the clusters cannot be ellipsoidal/spherical. Two alternative options were tested as shown in Figure 3: (a) manual clustering following the composition trend and (b) affinity propagation.

Affinity propagation (with optimal preference −250) resulted in 24 clusters each of comparable size (Figure 9b), while manual clustering following the composition trend yielded 21 clusters of varying sizes (Figure 9c). The clustering carried out by affinity propagation and manual clustering following the composition trend had ~ 30% of ~ 17.6% of the datapoints within each cluster not corresponding to the majority morphology of that cluster. Furthermore, both clustering techniques resulted in non-unique clusters as different clusters would have the same majority morphology: this reduces the inherent value of each cluster as a bin to capture a distinct morphological class, adversely impacting the usability of the cluster labels for the subsequent supervised ML tasks. The majority pattern of each cluster for both manual clustering following the composition trend and affinity propagation clustering from the Conv–4(4 Filters)—t-SNE dimensionality reduction is presented in the supplementary material together with a separate direct manual clustering of the high-resolution images based on the morphology.

We speculate that it may be possible to obtain further improvements in the clustering performance for this case by adopting classical image analysis and feature-engineering approaches. However, such approaches are beyond the scope of the present work as the premise of applying unsupervised ML is

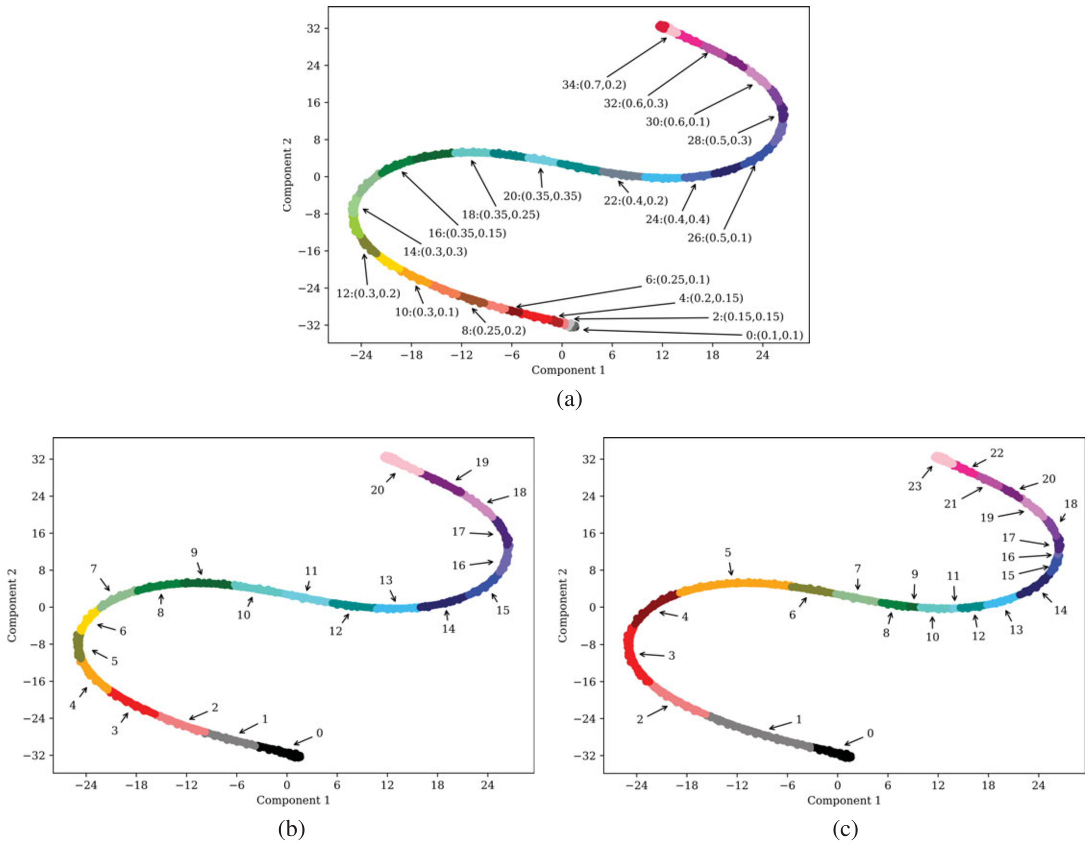


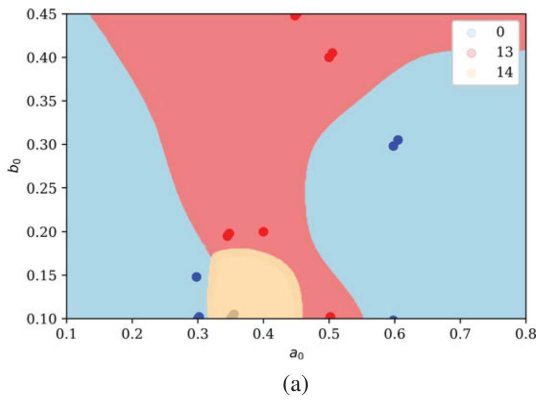
Figure 9. *(a) Reduced datapoints labeled by initial composition: each group of constant* (a_0, b_0) *contains mulitple simulations with varying* χ_{ij} *(half of the cluster labels are omitted for clarity); (b) affinity propagation clustering on Conv–4 (4 filters) with t-distributed stochastic neighbor embedding (t-SNE); (c) manual clustering on Conv–4 (4 filters) with t-SNE embedding: applying t-SNE to the bottleneck values of Conv autoencoders arranged the datapoints based on initial composition. Affinity propagation yielded* $k = 21$, *while manual clustering following the trend yielded* $k = 24$ *clusters.*

defeated when feature engineering is performed. Classical feature engineering and extraction identify defined features such as edges or shapes; hence, classical image analysis is conceptually similar to manual clustering. In addition, while the dimensionality reduction and clustering sections of the proposed workflow had limited success in the present study, the workflow is generalizable and can be implemented with minimal modification. However, performing feature engineering constrains the workflow as it requires the features to be re-engineered for each new problem.

3.3. Morphology prediction

The manual cluster identities obtained by performing manual clustering directly on the high-resolution images were used to train a prediction model. Even though affinity propagation clustering and manual clustering following the composition trend on the Conv–t-SNE embedding as outlined in Figure 3 was able to identify clusters with reasonable accuracy (Figure 9b), each cluster did not represent an intrinsic morphology. Therefore, the manual labels were deemed to be more appropriate for downstream supervised learning.

The simulation has a total of eight parameters: the composition which is controlled by a_0, b_0, the polymer chain lengths N_i for $i = 1,2,3$, and the binary interaction parameters χ_{ij} for $(i,j) = (1,2)$, $(1,3)$, and $(2,3)$. Polymers of the same chain length, $N_i = 1,000$ for all i, were considered in this study, which



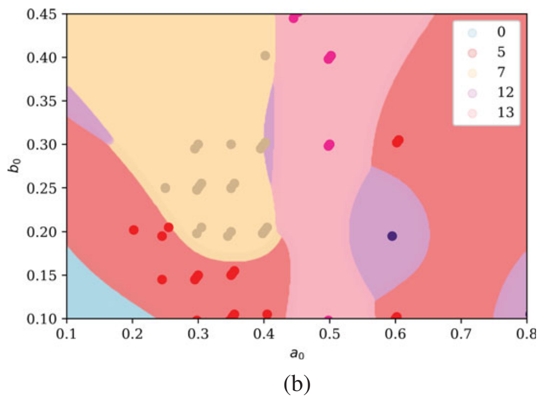


Figure 10. *Prediction of blend morphology for (a)* $\chi_{ij} = \chi_{ik} = \chi_{jk} = 0.003$ *and (b)* $\chi_{ij} = \chi_{jk} = 0.006, \chi_{ik} = 0.003$.

resulted in a total of five independent variables. For ease of visualization, 2D slices of the five-dimensional (5D) space are presented when we consider how the composition affects the morphology for different cases of interaction parameters. The entire image data set together with corresponding simulation parameters is available in the supplementary material.

The quantity of raw data for a given slice was approximately 25–50 datapoints. The low count was deemed inadequate for stable predictions (whereby the prediction quality becomes independent of the data quantity), and data augmentation was hence applied. Performing additional simulation runs was not considered due to computational limitations and anticipated numerical instability issues. Data augmentation was performed under the assumption that slight perturbations in a_0 and b_0 do not change the morphology of the polymer blend. The size of the data set was increased threefold by considering $(a_0 \pm \varepsilon, b_0 \pm \varepsilon)$ for $\varepsilon \in \{0.002, 0.005\}$. The associated uncertainty introduced to the prediction was bounded by 5%. This comes about when considering the smallest datapoint $a_0 = 0.1$, $b_0 = 0.1$ and the largest change of ± 0.005. The ℓ_2 norm changes by 5% in this case and is lower for all other datapoints.

As shown as Figure 10, the prediction process was able to yield maps whereby input values of a_0 and b_0 could be mapped to distinct morphological clusters using the cluster numbers from the manual clustering process. The test data have been overlaid onto each plot for reference. The prediction accuracy for the first case presented where $\chi_{ij} = \chi_{ik} = \chi_{jk} = 0.003$ was found to be 100%. The second case where $\chi_{ij} = \chi_{jk} = 0.006$ and $\chi_{ik} = 0.003$ had a prediction accuracy of 93%. The high performance of the GPC in mapping the regimes demonstrates the capability of supervised ML techniques for morphology prediction.

4. Conclusion

Physics-based simulations of ternary polymer demixing were implemented using Cahn–Hilliard theory and a numerical solver. Despite difficulties due to numerical instabilities, a comprehensive data set was generated that covers meaningful parameter ranges, including Flory–Huggins interaction parameters and molecular size.

The performance of conventional dimensionality reduction techniques (PCA and t-SNE) when clustering the simulation results into distinct categories was inadequate for use in downstream supervised learning tasks. Application of ML to the present simulation set remains a challenging task: the techniques struggled to identify unique polymer-blend features that are important for morphology characterization. It may well be possible to apply more sophisticated clustering techniques; the time and cost investments may significantly exceed those of direct manual labeling and yield comparatively poorer results. Supervised ML using GPC was used to predict the polymer-blend morphology to within $\geq 93\%$ accuracy; the accuracy is anticipated to increase with the addition of further simulation training data.

The data set (included in the supplementary material) enables users to obtain reasonable first predictions of polymer-blend morphologies for polymers with comparable physical parameters, bypassing computationally expensive simulations: resources can hence be targeted at regions of interest in the physical parameter space. The present framework can be readily extended for ternary polymer blends with modified physical properties. Extension is also envisioned for entirely different systems, including polymer-polymer (PP) and PPS systems, and the coupling of Navier–Stokes models for prediction of shear on polymer-blend morphology.

Acknowledgment. We acknowledge Funding from the UK Research and Innovation, and Engineering and Physical Sciences Research Council through the PREdictive Modeling with QuantIfication of UncERtainty for MultiphasE Systems (PREMIERE) programme, grant number EP/T000414/1, the Alan Turing Institute AI for Science and Government programme. O.K.M. acknowledges the Royal Academy of Engineering Research Chair in Multiphase Fluid Dymamics. I.P. acknowledges funding from the Imperial College Research Fellowship scheme (ICRF).

Competing Interests. The authors declare no competing interests exist.

Author Contributions. Conceptualization: O.K.M. and L.R.M.; Methodology, L.R.M. and P.I.; I.P.; Software, L.R.M., I.P., and P.I.; Investigation, L.R.M., P.I., and M.H.; Supervision, O.K.M. and L.R.M.; Visualization, P.I. and M.H.; Writing-original draft, P.I. and M.H.; Writing-review & editing, P.I., L.R.M., and I.P. All authors approved the final submitted draft.

Data Availability Statement. The full data set used in this study along with the scripts and environment configuration files needed to run the simulations and machine learning tasks can be found in the following repository: https://github.com/ImperialCollegeLondon/polymer_blend_morphology

Supplementary Materials. To view supplementary material for this article, please visit http://dx.doi.org/10.1017/dce.2020.14.

References

Aldhaheri M, Wei M, Bai B and Alsaba M (2017) Development of machine learning methodology for polymer gels screening for injection wells. *Journal of Petroleum Science and Engineering 151*(August 2016), 77–93.

Alfarraj AA and Nauman EB (2007) Spinodal decomposition in ternary systems with significantly different component diffusivities. *Macromolecular Theory and Simulations*, *16*(6), 627–631.

Autonomio (2019) Autonomio talos: Hyperparameter optimization for Keras.

Bianchi E, Panagiotopoulos AZ and Nikoubashman A (2015) Self-assembly of Janus particles under shear. *Soft Matter*, *11*(19), 3767–3771.

Brunswick A, Cavanaugh TJ, Mathur D, Russo AP and Nauman EB (1998) Experimental confirmation of computer-aided polymer blend designs. *Journal of Applied Polymer Science*, *68*(2), 339–343.

Brunton SL, Noack BR and Koumoutsakos P (2020) Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, *52*(1), 477–508.

Cahn JW and Hilliard JE (1958) Free energy of a nonuniform system I. Interfacial free energy. *The Journal of Chemical Physics*, *28*(2), 258–267.

Cai J, Luo J, Wang S and Yang S (2018) Feature selection in machine learning: A new perspective. *Neurocomputing*, *300*, 70–79.

Chen M, Shi X, Zhang Y, Wu D and Guizani M (2017) Deep features learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data*, *7790*(c), 1–1.

Chen P-Y and Huang J-J (2019) A hybrid autoencoder network for unsupervised image clustering. *Algorithms*, *12*(6), 122.

Chmiela S, Tkatchenko A, Sauceda HE, Poltavsky I, Schütt KT and Müller K-R (2017) Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, *3*(5), e1603015.

Chollet F. *et al.* (2018) Keras.

Cogswell DA (2010) *A phase-field study of ternary multiphase microstructures.* PhD thesis. Massachusetts Institute of Technology.

Delacruz-Araujo RA, Beltran-Villegas DJ, Larson RG and Córdova-Figueroa UM (2016) Rich Janus colloid phase behavior under steady shear. *Soft Matter*, *12*(18), 4071–4081.

Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E and Notredame C (2017) Nextflow enables reproducible computational workflows. *Nature Biotechnology*, *35*(4), 316–319.

Duraisamy K, Iaccarino G and Xiao H (2019) Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, *51*(1), 357–377.

Fernandez Martinez R, Iturrondobeitia M, Ibarretxe J and Guraya T (2017) Methodology to classify the shape of reinforcement fillers: Optimization, evaluation, comparison, and selection of models. *Journal of Materials Science*, *52*(1), 569–580.

Frey BJ and Dueck D (2007) Clustering by passing messages between data points. *Science*, *315*(5814), 972–976.

Gooneie A, **Schuschnigg S**, **Holzer C**, **Gooneie A**, **Schuschnigg S and Holzer C** (2017) A review of multiscale computational methods in polymeric materials. *Polymers*, *9*(12), 16.

Guo W, **Li M and Zhou J** (2013) Modeling programmable deformation of self-folding all-polymer structures with temperature-sensitive hydrogels. *Smart Materials and Structures*, *22*(11), 115028.

Guyer JE, **Wheeler D and Warren JA** (2009) FiPy: Partial differential equations with Python. *Computing in Science & Engineering*, *11*(3), 6–15.

Han Y and Elliott J (2007) Molecular dynamics simulations of the elastic properties of polymer/carbon nanotube composites. *Computational Materials Science*, *39*(2), 315–323.

Hinton GE (2006) Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Inguva PK, **Ooi SM**, **Desai PM and Heng PW** (2015) Encapsulation of volatiles by homogenized partially-cross linked alginates. *International Journal of Pharmaceutics*, *496*(2), 709–716.

Inokuchi T, **Li N**, **Morohoshi K and Arai N** (2018) Multiscale prediction of functional self-assembled materials using machine learning: High-performance surfactant molecules. *Nanoscale*, *10*(34), 16013–16021.

Janet JP, **Chan L and Kulik HJ** (2018) Accelerating chemical discovery with machine learning: Simulated evolution of spin crossover complexes with an artificial neural network. *The Journal of Physical Chemistry Letters*, *9*(5), 1064–1071.

Jokisaari A, **Voorhees P**, **Guyer J**, **Warren J and Heinonen O** (2017) Benchmark problems for numerical implementations of phase field models. *Computational Materials Science*, *126*, 139–151.

Jørgensen PB, **Mesta M**, **Shil S**, **García Lastra JM**, **Jacobsen KW**, **Thygesen KS and Schmidt MN** (2018) Machine learning-based screening of complex molecules for polymer solar cells. *The Journal of Chemical Physics*, *148*(24), 241735.

Kriegel H-P, **Kröger P and Zimek A** (2009) Clustering high-dimensional data. *ACM Transactions on Knowledge Discovery from Data*, *3*(1), 1–58.

Lao LL, **Venkatraman SS and Peppas NA** (2008) Modeling of drug release from biodegradable polymer blends. *European Journal of Pharmaceutics and Biopharmaceutics*, *70*(3), 796–803.

Lecun Y, **Bottou L**, **Bengio Y and Haffner P** (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee VE, **Sosa C**, **Liu R**, **Prud'homme RK and Priestley RD** (2017) Scalable platform for structured and hybrid soft nanocolloids by continuous precipitation in a confined environment. *Langmuir*, *33*(14), 3444–3449.

Li N, **Panagiotopoulos AZ and Nikoubashman A** (2017) Structured nanoparticles from the self-assembly of polymer blends through rapid solvent exchange. *Langmuir*, *33*(24), 6021–6028.

Logg A, **Mardal K-A and Wells G** (eds) (2012) *Automated Solution of Differential Equations by the Finite Element Method*, Vol. *84* of *Lecture Notes in Computational Science and Engineering.* Berlin Heidelberg: Springer.

López-Donaire ML, **Sussman EM**, **Fernández-Gutiérrez M**, **Méndez-Vilas A**, **Ratner BD**, **Vázquez-Lasa B and San Román J** (2012) Amphiphilic self-assembled "polymeric drugs": Morphology, properties, and biological behavior of nanoparticles. *Biomacromolecules*, *13*(3), 624–635.

Luo Z and Jiang J (2010) Molecular dynamics and dissipative particle dynamics simulations for the miscibility of poly(ethylene oxide)/poly(vinyl chloride) blends. *Polymer*, *51*(1), 291–299.

Maaten LVD and Hinton G (2008) Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Meenakshisundaram V, **Hung J-H**, **Patra TK and Simmons DS** (2017) Designing sequence-specific copolymer compatibilizers using a molecular-dynamics-simulation-based genetic algorithm. *Macromolecules*, *50*(3), 1155–1166.

Möller JJ, **Körner W**, **Krugel G**, **Urban DF and Elsässer C** (2018) Compositional optimization of hard-magnetic phases with machine-learning models. *Acta Materialia*, *153*, 53–61.

Nauman EB and Balsara NP (1989) Phase equilibria and the Landau–Ginzburg functional. *Fluid Phase Equilibria*, *45*(2–3), 229–250.

Nauman EB and He DQ (1994) Morphology predictions for ternary polymer blends undergoing spinodal decomposition. *Polymer*, *35*(11), 2243–2255.

Nauman EB and Savoca J (2001) An engineering approach to an unsolved problem in multicomponent diffusion. *AIChE Journal*, *47*(5), 1016–1021.

Pavel D and Shanks R (2005) Molecular dynamics simulation of diffusion of O_2 and CO_2 in blends of amorphous poly(ethylene terephthalate) and related polyesters. *Polymer*, *46*(16), 6135–6147.

Pedregosa F, **Weiss R and Brucher M** (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peherstorfer B, **Kramer B and Willcox K** (2017) Combining multiple surrogate models to accelerate failure probability estimation with expensive high-fidelity models. *Journal of Computational Physics*, *341*, 61–75.

Petrishcheva E and Abart R (2012) Exsolution by spinodal decomposition in multicomponent mineral solutions. *Acta Materialia*, *60*(15), 5481–5493.

Petscharnig S, **Lux M and Chatzichristofis S** (2017) Dimensionality reduction for image features using deep learning and autoencoders. In *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing—CBMI '17*, Vol. Part F1301. New York: ACM Press, pp. 1–6.

Prathab B, **Subramanian V and Aminabhavi TM** (2007) Molecular dynamics simulations to investigate polymer-polymer and polymer-metal oxide interactions. *Polymer*, *48*(1), 409–416.

San O and Maulik R (2018) Machine learning closures for model order reduction of thermal fluids. *Applied Mathematical Modelling*, *60*, 681–710.

Shang Y, **Fang L**, **Wei M**, **Barry C**, **Mead J and Kazmer D** (2011) Verification of numerical simulation of the self-assembly of polymer-polymer-solvent ternary blends on a heterogeneously functionalized substrate. *Polymer*, *52*(6), 1447–1457.

Tipping ME and Bishop CM (1999) Mixtures of probabilistic principal component analyzers. *Neural Computation*, *11*(2), 443–482.

Tree DR, **Delaney KT**, **Ceniceros HD**, **Iwama T and Fredrickson GH** (2017) A multi-fluid model for microstructure formation in polymer membranes. *Soft Matter*, *13*(16), 3013–3030.

Ulbricht M (2006) Advanced functional polymer membranes. *Polymer*, *47*(7), 2217–2262.

Wang Y, **Yao H and Zhao S** (2016) Auto-encoder based dimensionality reduction. *Neurocomputing*, *184*, 232–242.

Ward L, **O'Keeffe SC**, **Stevick J**, **Jelbert GR**, **Aykol M and Wolverton C** (2018) A machine learning approach for engineering bulk metallic glass alloys. *Acta Materialia*, *159*, 102–111.

Wattenberg M, **Viégas F and Johnson I** (2016) How to use t-SNE effectively. *Distill*, *1*(10), e2.

Wei Q, **Melko RG and Chen JZY** (2017) Identifying polymer states by machine learning. *Physical Review E*, *95*(3), 032504.

Wodo O and Ganapathysubramanian B (2012) Modeling morphology evolution during solvent-based fabrication of organic solar cells. *Computational Materials Science*, *55*, 113–126.

Yang HC, **Xie Y**, **Hou J**, **Cheetham A K**, **Chen V and Darling SB** (2018) Janus membranes: Creating asymmetry for energy efficiency. *Advanced Materials*, *1801495*, 1–11.

Yuan C and Yang H (2019) Research on K-value selection method of K-means clustering algorithm. *J*, *2*(2), 226–235.

Zhou B and Powell AC (2006) Phase field simulations of early stage structure formation during immersion precipitation of polymeric membranes in 2D and 3D. *Journal of Membrane Science*, *268*(2), 150–164.

Cite this article: Inguva, P. R. Mason, L. Pan, I. Hengardi, M. and Matar, O. K. 2020. Numerical simulation, clustering, and prediction of multicomponent polymer precipitation. *Data-Centric Engineering*, 1: e13. doi:https://doi.org/10.1017/dce.2020.14