Soft Matter

Deep Learning for Automated Classification and Characterization of Amorphous Materials

Journal:	*Soft Matter*
Manuscript ID	SM-ART-09-2019-001903.R1
Article Type:	Paper
Date Submitted by the Author:	15-Nov-2019
Complete List of Authors:	Swanson, Kirk; University of Chicago, Department of Computer Science; University of Chicago, Pritzker School of Molecular Engineering Trivedi, Shubhendu; MIT, Computer Science and Artificial Intelligence Laboratory; Brown University, Institute for Computational and Experimental Research in Mathematics Lequieu, Joshua; University of California Santa Barbara, Materials Research Laboratory; University of Chicago, Pritzker School of Molecular Engineering Swanson, Kyle; MIT, Computer Science and Artificial Intelligence Laboratory Kondor, Risi; University of Chicago, Department of Computer Science; University of Chicago, Department of Statistics; Flatiron Institute, Center for Computational Mathematics

Soft Matter

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

Deep Learning for Automated Classification and Characterization of Amorphous Materials†

Kirk Swanson,*[ab] Shubhendu Trivedi,[cd] Joshua Lequieu,[be] Kyle Swanson,[c] and Risi Kondor[afg]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

It is difficult to quantify structure-property relationships and to identify structural features of complex materials. The characterization of amorphous materials is especially challenging because their lack of long-range order makes it difficult to define structural metrics. In this work, we apply deep learning algorithms to accurately classify amorphous materials and characterize their structural features. Specifically, we show that convolutional neural networks and message passing neural networks can classify two-dimensional liquids and liquid-cooled glasses from molecular dynamics simulations with greater than 0.98 AUC, with no *a priori* assumptions about local particle relationships, even when the liquids and glasses are prepared at the same inherent structure energy. Furthermore, we demonstrate that message passing neural networks surpass convolutional neural networks in this context in both accuracy and interpretability. We extract a clear interpretation of how message passing neural networks evaluate liquid and glass structures by using a self-attention mechanism. Using this interpretation, we derive three novel structural metrics that accurately characterize glass formation. The methods presented here provide us with a procedure to identify important structural features in materials that could be missed by standard techniques and give us a unique insight into how these neural networks process data.

1 Introduction

Classifying material structures and predicting their properties are important tasks in materials science. The behavior of materials often depends strongly on their underlying structure, and understanding these structure-property relationships relies on accurately describing the structural features of a material. However, quantifying structure-property relationships and identifying structural features in complex materials are difficult tasks.

A variety of standard techniques have been developed to analyze material structures. Some of the most common techniques include the Steinhardt bond order parameters,[1] Bond Angle Analysis (BAA),[2] and Common Neighbor Analysis (CNA),[3] which are useful for detecting order-disorder transitions and differentiating between crystal structures in ordered samples.[4,5] Radial distribution functions, which measure a spatial average of interparticle distances, are also widely used for analyzing different materials and phases.[6]

However, these standard techniques have limitations, especially for analyzing weakly crystalline or amorphous materials. As discussed in Reinhart *et al.*,[4] the Steinhardt bond order parameters can be stymied by thermal fluctuations or ambiguous crystal structures. BAA relies on a small set of crystalline reference structures that may not be present in amorphous samples. CNA is more flexible than BAA, but it cannot provide accurate information about particles that do not exhibit known symmetries, making analysis of irregular structures challenging. Radial distribution functions rely on spatial averaging, which interferes with the ability to characterize complex anisotropic structures.

To overcome some of these limitations, recent studies have focused on developing machine learning algorithms to automate material structure characterization. Many of these studies have concentrated on using supervised machine learning for crystal structure identification and have shown improvements over standard techniques. Geiger and Dellago and Dietz *et al.*,[5,7] for ex-

[a] *Department of Computer Science, The University of Chicago, Chicago, IL 60637. E-mail: swansonk1@uchicago.edu*
[b] *Pritzker School of Molecular Engineering, The University of Chicago, Chicago, IL 60637.*
[c] *Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139. E-mail: shubhendu@csail.mit.edu, swansonk@mit.edu*
[d] *Institute for Computational and Experimental Research in Mathematics, Brown University, Providence, RI 02903.*
[e] *Materials Research Laboratory, University of California, Santa Barbara, CA 93106. E-mail: lequieu@mrl.ucsb.edu*
[f] *Department of Statistics, The University of Chicago, Chicago, IL 60637. E-mail: risi@cs.uchicago.edu*
[g] *Center for Computational Mathematics, Flatiron Institute, New York, NY 10010.*
† Electronic Supplementary Information (ESI) available. See DOI: 10.1039/cXsm00000x/

ample, classify crystalline structures in polymorphic and mixed phase systems more accurately by using a set of local structure functions that are fed into a neural network. Ziletti *et al.*[8] use convolutional neural networks to automate the process of classifying crystal symmetries, even in the presence of substantial defects, using images of computationally generated diffraction patterns. Similar supervised learning methods have been used to analyze a variety of other ordered systems, including graphene.[9–12]

Other studies have concentrated on crystal structure identification using unsupervised learning techniques, such as dimensionality-reduction algorithms. These unsupervised methods allow for the discovery of previously unidentified local structural features, a task that is often out of reach for standard methods and many supervised learning algorithms. For example, Reinhart *et al.*[4] use diffusion maps and clustering to identify new local structures in colloidal crystals, and Spellings and Glotzer[13] use spherical harmonic functions in conjunction with Gaussian mixture models to automatically identify different crystalline arrangements without relying on a set of reference structures. Unsupervised learning has also been used to study systems ranging from Lennard-Jones crystals to proteins.[14–19]

Although these machine learning approaches have shown substantial improvements over standard techniques when applied to a variety of systems, many of them still rely on the development of complex, hand-crafted descriptors of local particle environments. The results of machine learning analyses can be quite sensitive to the definitions of these descriptors.[14] Moreover, most of these studies have focused on characterization of crystalline or semi-crystalline structures, with little attention given to analysis of completely amorphous systems such as glassy materials.

The characterization of glassy systems is especially challenging because their lack of long-range order makes it difficult to define structural metrics. Nevertheless, several recent studies have been able to manually identify structural metrics for glassy systems. For example, Hu *et al.*[20] define a metric, the average degree of local fivefold symmetry, that can differentiate between configurations of metallic liquids and glasses and that has a quantitative relationship with dynamics during glass formation. Reid *et al.*[21] use a similar fivefold symmetry metric, based on the spherical harmonic functions, to compare the structures of two-dimensional liquid-cooled and vapor-deposited glasses.

There are also some recent studies that successfully used supervised machine learning to uncover previously unknown relationships between structure and dynamics in glassy materials,[22–30] in addition to some earlier work that explored unsupervised learning to that end.[31] In the supervised learning approaches, an algorithm, called the support vector machine, is used to define a metric, called "softness," that identifies populations of particles that are likely to dynamically rearrange. In this context, "softness" is used to link structure and dynamics, but it is not used to directly identify local structural features or to classify different material structures.

In this work, we use deep learning to accurately classify amorphous materials and to derive new metrics that characterize their structures. We use two-dimensional liquids and liquid-cooled glasses generated by molecular dynamics simulations as archetypal examples of amorphous materials. Our classification algorithms make no *a priori* assumptions about local relationships between particles and are not dependent on complex hand-crafted descriptors that define local particle environments.

We explore the application of two different types of deep learning algorithms: convolutional neural networks and message passing neural networks. Convolutional neural networks have been used in a variety of material classification tasks,[8–10,12] including the classification of ordered and disordered configurations from simulations of the Ising model,[32–34] but to the best of our knowledge they have never before been used to classify different amorphous material structures from molecular dynamics simulations. By rendering particle configurations of liquids and glasses as two-dimensional images, we are able to distinguish between them with high accuracy using convolutional neural networks. However, there are several limitations that accompany the use of convolutional neural networks, including the potential introduction of artifacts via image rendering and limited interpretability.

We overcome these issues by using message passing neural networks that operate directly on the Cartesian coordinates of particles by representing a configuration of particles as a graph. Message passing neural networks have been used previously to predict properties of molecules, such as toxicity and solubility,[35–42] and to predict properties of crystals, such as formation energy and shear modulus.[43] However, to the best of our knowledge message passing neural networks have never before been used to classify different amorphous material structures from molecular dynamics simulations. By representing particle configurations of liquids and glasses as graphs, we are able to distinguish between them with high accuracy using message passing neural networks. Moreover, by using a technique called self-attention, we are able to extract an interpretation of how message passing neural networks evaluate liquid and glass configurations. Using this interpretation, we derive three novel structural metrics that characterize glass formation and that can differentiate between liquid and glass configurations without the use of machine learning. This not only provides us with a general method for identifying important structural features in amorphous materials, but it also gives us a unique insight into how these neural networks process data. This result provides clear proof of concept that message passing neural networks could be used in more complex and demanding classification and characterization tasks that stymie standard techniques.

2 Methods

2.1 Simulation Details

Glasses are kinetically arrested states of matter which are generally prepared by cooling a liquid to temperatures below the glass transition, T_g, of the corresponding bulk material.[21] When cooling is sufficiently rapid, the system avoids crystallization and instead solidifies into a glass, an amorphous state which has an atomic structure similar to that of a liquid but with the mechanical properties of a solid. The specific properties of these liquid-cooled glasses depend on the rate at which they are cooled, as lower cooling rates lead to materials that lie deeper in the under-

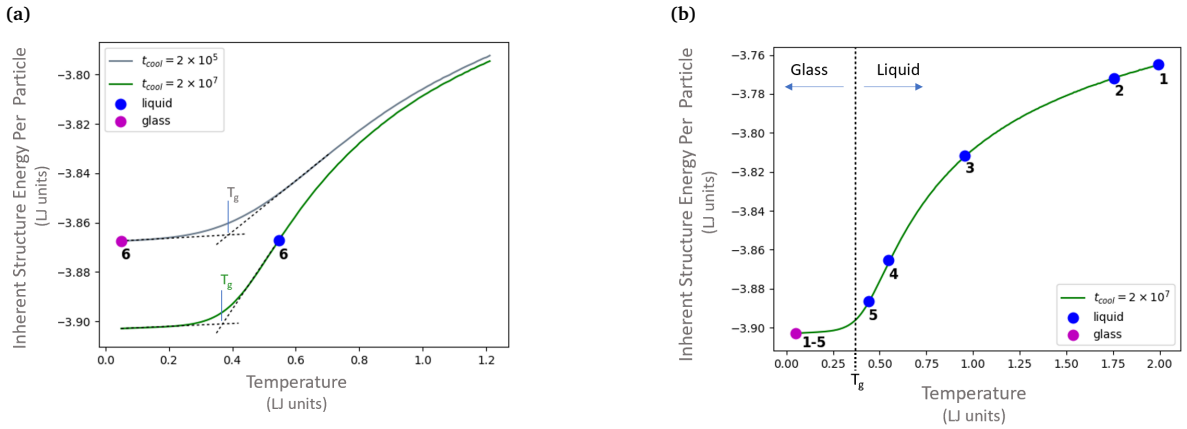

Fig. 1 (a) Average inherent structure energies per particle as a function of temperature for 10,000 configurations with $t_{cool} = 2 \times 10^5$ and 10,000 configurations with $t_{cool} = 2 \times 10^7$. T_g for each curve is calculated as the intersection of linear approximations to the supercooled liquid regime and the glass regime, as shown by the dotted black lines. We consider configurations at temperatures above T_g to be liquids and configurations below T_g to be glasses. The magenta and blue points indicate the temperatures at which we select glass and liquid configurations for dataset 6, respectively. Standard deviations of energy values, not shown here, are on the order of 10^{-3}. **(b)** Average inherent structure energy per particle as a function of temperature for the 10,000 configurations with $t_{cool} = 2 \times 10^7$. The magenta point indicates the temperature at which we select glass configurations for datasets 1 through 5. The blue points indicate the temperatures at which we select liquid configurations, labeled by dataset number.

lying potential energy landscape.

To simulate two-dimensional liquids and liquid-cooled glasses, we used the Kob-Andersen model, which consists of a binary mixture of spheres whose glass-forming behavior in the bulk has been studied extensively.[44] This binary mixture is comprised of 65% type A and 35% type B particles which have unit mass and which interact according to the pairwise Lennard-Jones potential,

$$V_{ij}(r) = \begin{cases} 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] & r \leq r_{cut,ij} \\ 0 & r > r_{cut,ij} \end{cases} \qquad i,j \in \{A,B\},$$

where r is the distance between a pair of particles, ε_{ij} characterizes the depth of the potential, and σ_{ij} characterizes the finite distance below $r_{cut,ij}$ at which the potential is zero. Specific parameter values for different values of i and j, given in Lennard-Jones units, are shown in Table 1.

Table 1 Lennard-Jones parameters for simulations of liquids and liquid-cooled glasses. Note that $\varepsilon_{ij} = \varepsilon_{ji}$, $\sigma_{ij} = \sigma_{ji}$, and $r_{cut,ij} = r_{cut,ji}$

ε	σ	r_{cut}
$\varepsilon_{AA} = 1.00$	$\sigma_{AA} = 1.00$	$r_{cut,\,AA} = 2.50$
$\varepsilon_{AB} = 1.50$	$\sigma_{AB} = 0.80$	$r_{cut,\,AB} = 2.00$
$\varepsilon_{BB} = 0.50$	$\sigma_{BB} = 0.88$	$r_{cut,\,BB} = 2.20$

We performed 20,000 independent simulations of this model using LAMMPS. ‡ Each simulation contained a fixed total of 4,320 particles in a simulation box of length $60\sigma_{AA}$ in the x- and y-directions.§ Each simulation was performed in the canonical NVT ensemble with periodic boundaries in the x- and y-directions, and the temperature was reduced linearly from an initial temperature of 2.0 to a final temperature of 0.05 in Lennard-Jones units using a Nose-Hoover thermostat for t_{cool} simulation steps. During each simulation we recorded the inherent structure energy per particle of the system using the FIRE minimization algorithm.[45] The inherent structure energy of a configuration, used to quantify a configuration's stability, is the potential energy brought to its local minimum.[21]

Each of these simulations models a liquid that cools and solidifies into a glass below T_g. For 10,000 of the simulations we used $t_{cool} = 2 \times 10^7$ cooling steps, and for the other 10,000 we used $t_{cool} = 2 \times 10^5$ cooling steps. We calculated the average glass transition temperature, T_g, for each cooling rate by identifying two linear regimes in each of the average inherent structure energy curves, one corresponding to the supercooled liquid regime and the other to the glass regime.[46] We then fit the data in these regimes and calculated the intersection point, as shown in Figure 1(a), which gives $T_g \approx 0.37$ for $t_{cool} = 2 \times 10^7$ and $T_g \approx 0.39$ for $t_{cool} = 2 \times 10^5$. The glass configurations generated using $t_{cool} = 2 \times 10^7$ are at a lower average inherent structure energy than those generated using $t_{cool} = 2 \times 10^5$ because $t_{cool} = 2 \times 10^7$ corresponds to a lower cooling rate.

2.2 Datasets

Our goal was to train machine learning algorithms to perform a binary classification task: identify a particle configuration as a liquid or a glass. To that end, we used the particle configurations generated by the simulations described in §2.1 to construct six datasets.

Datasets 1 through 5 are each composed of 10,000 glass configurations and 10,000 liquid configurations, all taken from the simulations with $t_{cool} = 2 \times 10^7$. The glass configurations in each dataset are all at a temperature of 0.05. In order of increasing dataset number, the liquid configurations are at temperatures

‡ See https://lammps.sandia.gov/.

§ We chose this system size to be consistent with the simulations in Reid *et al.*[21] The units used in these simulations are Lennard-Jones units.

1.99, 1.76, 0.96, 0.55, and 0.44, as enumerated in Figure 1(b).

In datasets 1 through 5, the liquid configurations are at higher inherent structure energies than the glass configurations. In order to compare liquid and glass configurations at the same average inherent structure energy, we constructed dataset 6, labeled in Figure 1(a). This dataset has 10,000 glass configurations at a temperature of 0.05 taken from simulations with $t_{cool} = 2 \times 10^5$ and 10,000 liquid configurations at a temperature of 0.55 taken from simulations with $t_{cool} = 2 \times 10^7$.

Each dataset serves as a test for a machine learning algorithm's ability to distinguish between amorphous material structures, and they are numbered roughly in order of increasing difficulty. Dataset 1 represents a relatively easier test, because the liquid and glass configurations in this dataset have a large difference in average inherent structure energy and distinctly different local structures, as exhibited by their average radial distribution functions (see Figure 1 in the Supplementary Information†). Datasets 2 through 5 represent increasingly difficult tests as the differences in average inherent structure energy decrease and the radial distribution functions converge. Dataset 6 represents one of the most difficult tests, because the liquid and glass configurations are, on average, at the same inherent structure energy, as shown in Figure 1(a), and have very similar average radial distribution functions (see Figure 2 in the Supplementary Information†). We chose to run simulations in the NVT ensemble so that the density for each configuration is constant, providing a machine learning algorithm with a more challenging task than if the densities of the liquids and glasses were significantly different.

2.3 Convolutional Neural Networks

In this section we give a brief overview of convolutional neural networks (CNNs) and a description of the architecture and training routine for CNNs used in this work.

Neural networks have been proven capable of approximating a wide set of functions.[47] One of the most fundamental types of neural network, called a fully connected feedforward network, is essentially a composition of affine transformations modified by nonlinear functions. These transformations can be represented as a series of interconnected neurons arranged in layers through which input data flows. The transformation corresponding to a single layer, f, is of the form $f(x) = \alpha(Wx+b)$, where x is a vector containing input data, W is a matrix of weights, b is a vector of weights, called the bias, and α is a nonlinear function, called the activation function. Given a loss function to be minimized, neural networks are typically trained via the backpropagation algorithm and gradient descent, in which the weights are iteratively adjusted to reduce their contribution to the loss. The amount by which they are adjusted in each step of training is partially controlled by a coefficient called the learning rate. One of the most common activation functions used in neural networks is the rectified linear unit (ReLU), given by $\alpha(x) = \max(0,x)$, because it has been shown to be effective for gradient-based learning.[48]

Convolutional neural networks, which use convolutional layers in addition to the fully connected layers that characterize a basic feedforward network, have been shown to excel at computer vision tasks ranging from assessing cancer risk in radiology scans to galaxy morphology classification in telescope images.[29,49,50] As explained in §1, CNNs have also been used to classify a variety of materials, including crystal structures and Ising model configurations.

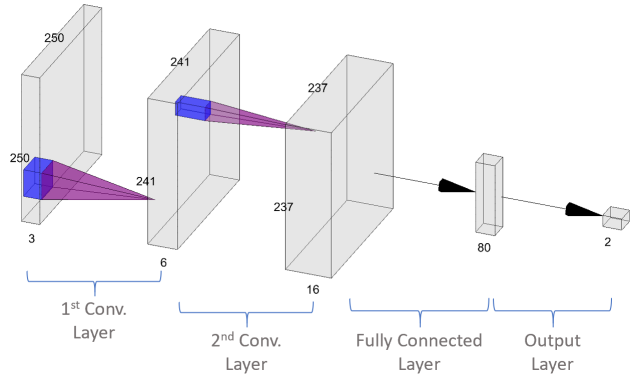

Fig. 2 Convolutional neural network (CNN) architecture used in this work. Two convolutional layers are followed by a fully connected layer and an output layer. In order to be analyzed by a CNN, particle configurations are rendered as images.

In a convolutional layer, a set of matrices, or kernels, is convolved with an input matrix to produce a set of output matrices, or feature maps.[48] Notably, convolutions are equivariant to translation, a property that underlies the effectiveness of CNNs in detecting features such as edges and shapes in different locations of an image, while also significantly reducing the number of parameters compared to a basic fully connected feedforward network.

2.3.1 Network Architecture

The CNN that we developed, shown in Figure 2,¶ has two convolutional layers. The first has 6 kernels with dimensions $10 \times 10 \times 3$ along with a bias vector and ReLU activation, with no zero padding and a stride of 1. Adding extra rows and columns of zeros to the boundaries of an input matrix, a technique called zero padding, is sometimes used to maintain the original dimensions of an input matrix as it flows through the layers of a CNN. A stride of 1 means that the kernels are shifted 1 pixel at a time as they are convolved with an input matrix. The second layer has 16 kernels with dimensions $5 \times 5 \times 6$ along with a bias vector and ReLU activation, also with no zero padding and a stride of 1. The 16 feature maps that are output by these layers, which have dimensions 237×237, are then flattened and fed into a fully connected layer with 80 neurons and a bias vector, followed by ReLU activation. Dropout, a regularization method that has been shown to reduce overfitting,[51] is applied to this fully connected layer followed by a final output layer with two neurons. We used TensorFlow to build and train these models, ‖ and all of our code is available in our GitHub repository.**

¶ This diagram was produced using the tools at http://alexlenail.me/NN-SVG/LeNet.html.

‖ See www.tensorflow.org.

** See https://github.com/ks8/glassML.

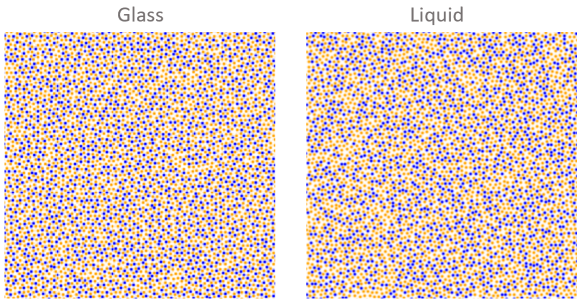

Fig. 3 Representative images of glass and liquid particle configurations from dataset 1 prepared for a CNN. Orange dots are type A particles and blue dots are Type B particles.

2.3.2 Training

To prepare a dataset for a CNN, we rendered each particle configuration as a 250 x 250 pixel PNG image, as shown in Figure 3. During training we use an on-the-fly data augmentation scheme whereby images are uniformly randomly rotated by 0, 90, 180, or 270 degrees and then flipped across the $y = 0.5$ axis (upside down) with a probability of 0.5, which effectively expands our dataset size. Each batch array is a tensor with dimensions $n_b \times 250 \times 250 \times 3$, where n_b is the batch size and the last dimension corresponds to the three color channels (red, green, blue) for color images.

Backpropagation is performed using cross-entropy loss with L^2 regularization (coefficient of 0.01) on all four layers of the network. L^2 regularization, which has been shown to reduce overfitting, refers to including the sum of the L^2 norms of the weights in the loss function. Weights are updated using the ADAM optimization procedure.[52] We used a learning rate schedule with a piecewise linear increase and exponential decay: the learning rate increases linearly from an initial learning rate of 1×10^{-4} to a maximum learning rate of 1×10^{-3} during the first two epochs, and then it decreases exponentially to a learning rate of 1×10^{-4} during the remaining epochs.[53]

2.4 Message Passing Neural Networks

In this section we give a brief overview of message passing neural networks (MPNNs) and a description of the architecture, training routine, and interpretation scheme for MPNNs used in this work.

As explained in §1, MPNNs have been used to predict properties of molecules and crystals by representing them as graphs, with atoms corresponding to nodes and bonds corresponding to edges. To the best of our knowledge, however, MPNNs have never before been used to classify and analyze large ensembles of particles or amorphous materials such as glasses.

As described in the message passing framework established by Gilmer *et al.*, MPNNs operate on undirected graphs G with node features x_v and edge features e_{vw}.[36] The MPNN processes these graphs in two phases: a message passing phase and a readout phase. In the message passing phase, the MPNN builds a representation of the input graph, and in the readout phase, the MPNN uses this representation to predict properties of interest.

The message passing phase runs for $\mathcal{T}$ steps. During each step t, hidden states h_v^t and messages m_v^{t+1} at each node v in the graph are updated using message function M_t and vertex update function U_t according to

$$\begin{aligned} m_v^{t+1} &= \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \\ h_v^{t+1} &= U_t(h_v^t, m_v^{t+1}), \end{aligned} \tag{1}$$

where $N(v)$ are the neighbors of v in graph G and h_v^0 is a function of the initial node features x_v. The readout phase uses some function R to make a property prediction of interest based on the final hidden states according to

$$\hat{y} = R\left(\left\{h_v^{\mathcal{T}} \,\middle|\, v \in G\right\}\right). \tag{2}$$

2.4.1 Network Architecture

Here we describe the particular variant of MPNN that we use in this work, as described in Yang *et al.* and proposed in Dai *et al.*[35,54] This variant, called Directed MPNN (D-MPNN), uses messages associated with directed edges rather than messages associated with nodes. Our motivation in using this particular architecture is that having messages passed along edges instead of nodes in the graph is more conducive to extracting an interpretation of the network using self-attention, as described in §2.4.3.

The D-MPNN works as follows, as shown in Figure 4. It operates on hidden states h_{vw}^t and messages m_{vw}^t at each edge connecting nodes v and w. The edges in graph G are directed, so that for any two connected nodes v and w, there is an edge from v to w and an edge from w to v. In this way, the messages are also directed: h_{vw}^t and m_{vw}^t are distinct from h_{wv}^t and m_{wv}^t. In the message passing phase, the hidden states and messages are updated according to

$$\begin{aligned} m_{vw}^{t+1} &= \sum_{k \in \{N(v) \setminus w\}} M_t(x_v, x_k, h_{kv}^t) \\ h_{vw}^{t+1} &= U_t(h_{vw}^t, m_{vw}^{t+1}). \end{aligned} \tag{3}$$

Note that the message m_{vw}^{t+1} is not a function of the reverse message m_{wv}^t from the previous step. Prior to the first step of message passing, edge hidden states are initialized according to

$$h_{vw}^0 = \alpha\left(W_i \; cat(x_v, e_{vw})\right), \tag{4}$$

where $W_i \in \mathbb{R}^{h \times h_i}$ is a learned matrix, $cat(x_v, e_{vw}) \in \mathbb{R}^{h_i}$ is the concatenation of the node features x_v for node v and the edge features e_{vw} for edge vw, and α is the ReLU activation function. The message passing functions M_t are given by

$$M_t(x_v, x_w, h_{vw}^t) = h_{vw}^t, \tag{5}$$

and the edge update functions are given by a neural network,

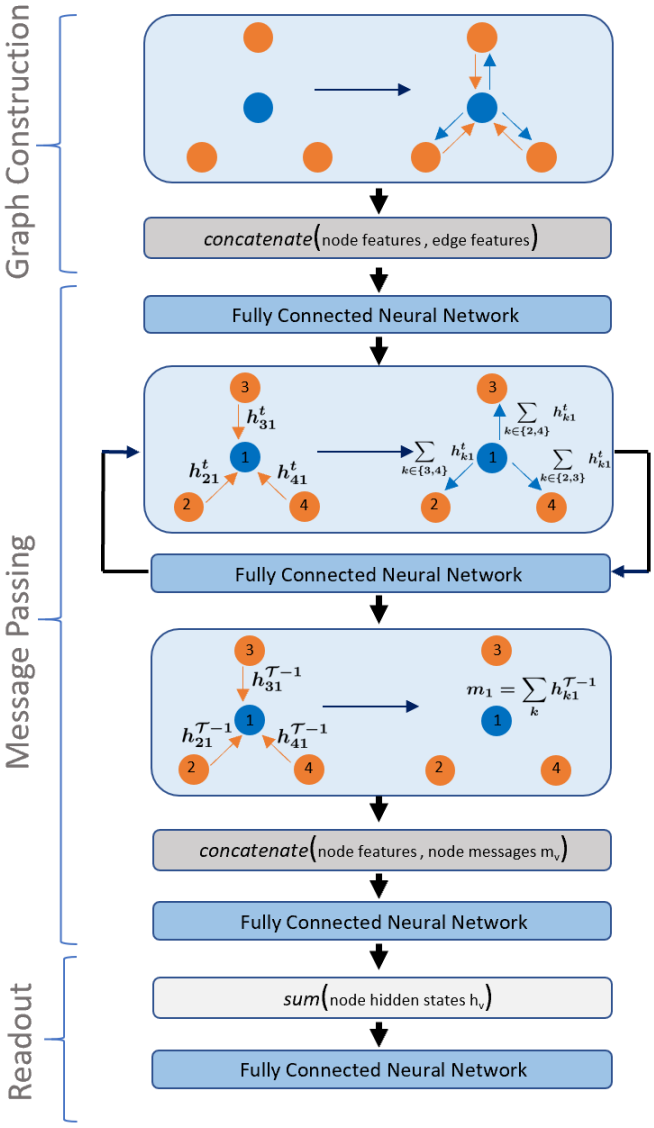

Fig. 4 Directed message passing neural network (D-MPNN) architecture used in this work. First, particle configurations are transformed into a graph representation by connecting each particle to its k nearest neighbors (graph construction). Then, after the edge hidden states are initialized via a fully connected layer, message passing along the edges is carried out, as in Eqns. 3 - 7 (message passing phase). Finally, property predictions are generated via a fully connected neural network, as in Eqns. 8 and 9 (readout phase).

$$U_t(h_{vw}^t, m_{vw}^{t+1}) = U(h_{vw}^t, m_{vw}^{t+1}) = \alpha\left(h_{vw}^0 + W_m m_{vw}^{t+1}\right), \tag{6}$$

where $W_m \in \mathbb{R}^{h\times h}$ is a learned matrix with hidden size h. Dropout is then applied. The presence of h_{vw}^0 in the above equation provides a skip connection to the original feature vector for that edge. After $\mathcal{T}-1$ steps of this message passing, a node representation of the graph is constructed by summing inbound edge features in a final message passing step according to

$$\begin{aligned} m_v &= \sum_{k\in N(v)} h_{kv}^{\mathcal{T}-1} \\ h_v &= \alpha\left(W_a \; cat(x_v, m_v)\right), \end{aligned} \tag{7}$$

where $W_a \in \mathbb{R}^{h\times h}$ is a learned matrix.

The readout phase is the same as for general MPNNs. For the readout function R, node hidden states are summed to obtain a feature vector for the graph

$$h = \sum_{v\in G} h_v. \tag{8}$$

Property predictions are then generated according to

$$\hat{y} = f(h), \tag{9}$$

where f is a feedforward neural network with ReLU activation and dropout at each layer.

2.4.2 Training

To prepare a dataset for a D-MPNN we extract a matrix of scaled particle coordinates $x, y \in [0,1]$ and particle types (A or B) for each configuration. In order to be processed by a D-MPNN, the raw coordinate data for a batch of configurations is transformed into a graph, which happens on-the-fly for each batch that is loaded for training.

To do this, we first implement a step for each configuration in a batch that effectively increases the size of the dataset. We specify a hyperparameter called the window length, $l_{win} \in (0,1)$, whose square is the area of a square window that we randomly select from each particle configuration. We select this window by drawing two random numbers uniformly from the interval [0, 1 - l_{win}], called x_{rand} and y_{rand}, and then select particles whose x, y coordinates satisfy $x_{rand} \le x \le x_{rand} + l_{win}$ and $y_{rand} \le y \le y_{rand} + l_{win}$. For example, if $l_{win} = 0.5$, we select a subset of particles in a random square window corresponding to about 25% of the original set of particles. This process effectively augments the size of the dataset.

We then transform this batch of modified configurations into a graph. Each individual configuration is transformed into a graph by connecting each particle to its k nearest neighbors, as illustrated in Figure 4. Each connection is comprised of two edges, one directed from particle (node) v to particle (node) w, and the other in the reverse direction. For each edge we compute the Euclidean distance between the connected nodes. Thus, the node

features x_v in this graph are the x-coordinate, y-coordinate, and particle type, t, for node v, and the edge feature e_{vw} is the distance between nodes v and w. The graphs corresponding to individual configurations in a batch are then concatenated into a single larger graph representing the entire batch. In this batch graph, the nodes corresponding to one configuration are connected to each other but not to nodes corresponding to other configurations.

During training, backpropagation is performed using cross-entropy loss. Weights are updated using the ADAM optimization procedure,[52] and we used the same learning rate schedule described in §2.3.2. Our code for D-MPNNs, written in PyTorch, [††] is built upon publicly available code. [‡‡] All of our code is available in our GitHub repository.[‡]

2.4.3 Interpretation

Currently, interpreting a neural network is very challenging. However, new techniques are beginning to provide avenues for accomplishing this task. One such technique is called self-attention, which is essentially a mechanism that allows us to examine which features of the data a neural network is paying "attention" to most.[53,55] Mathematically, self-attention is akin to a dot product that yields a set of weights for each feature, which are then interpreted as "attention" scores.

We place a self-attention mechanism on the edges of a graph and apply it during the message passing phase for the first $\mathcal{T}-1$ steps. This allows us to examine which edges of a graph the D-MPNN is paying the most "attention" to while training, which could give us insight into how the network is making its classification decisions.

Our self-attention mechanism for D-MPNNs works as follows. According to Eqn. 3, in round $t+1$ of message passing, the feature vector for each edge e_{vw} in a graph is updated to h_{vw}^{t+1}. Each of these vectors has length h (hidden size), and we assume that there are n edges in the graph. After Eqn. 3, we insert the following steps into the D-MPNN algorithm. The feature vectors h_{vw}^{t+1} are concatenated as row vectors in a matrix H, where $H \in \mathbb{R}^{n \times h}$. We then apply the following transformation to H:

$$\sigma = \text{softmax}\left(\alpha\left(HW_{attn}\right)v_{attn}\right), \tag{10}$$

where α is the ReLU transformation, $W_{attn} \in \mathbb{R}^{h \times h}$ is a learned parameter matrix, and $v_{attn} \in \mathbb{R}^{h}$ is a learned parameter vector. The weights σ are then dotted with the initial row vectors to yield a new set of hidden edge states:

$$H' = \sigma H. \tag{11}$$

Finally, the row vectors of H' replace the corresponding values of the edge features from Eqn. 3 and are applied as such to the next round of message passing. The weights σ represent the "attention" that the network is giving to each edge in the graph. As discussed in §3, we then used the *networkx* package in Python to quantify attributes of the graph structure of these self-attention weights for glass and liquid particle configurations.

2.5 Hyperparameter Optimization and Cross Validation

There are seven hyperparameters in our D-MPNN: window length l_{win}, number of nearest neighbors k, number of message passing steps $\mathcal{T}$, dropout probability p, hidden size h, number of feedforward layers f, and batch size n_b. There are two hyperparameters in our CNN: dropout probability and batch size. To discover optimal values for these hyperparameters, we used a Bayesian optimization scheme called a Tree-structured Parzen Estimator (TPE), implemented in the *hyperopt* package. [§] As described in Bergstra *et al.*, TPE is a type of sequential model-based global optimization algorithm which discovers optimal hyperparameters by modeling the loss function with a surrogate probability model and making increasingly well-informed guesses for a specified number of iterations (see §3 in the Supplementary Information[†] for more details).[56] These algorithms have been shown to exceed the performance of grid search and random search when optimizing for multiple hyperparameters. Table 2 shows the range of hyperparameter values that we explored using *hyperopt*.

Table 2 Ranges of values used for hyperparameter optimization. To discover optimal values in these ranges, we used a Bayesian optimization scheme called a Tree-structured Parzen Estimator (TPE).[56] TPE discovers optimal hyperparameters by modeling the loss function with a surrogate probability model and making increasingly well-informed guesses for a specified number of iterations. We found that batch sizes greater than five for D-MPNNs sometimes exceeded GPU memory, so we fixed $n_b = 5$ for D-MPNNs

Hyperparameter	Low	High	Step Size
# nearest neighbors (k)	1	5	1
window length (l_{win})	0.1	0.3	0.05
hidden size (h)	300	2400	100
# message passing steps ($\mathcal{T}$)	2	6	1
dropout probability (p)	0.0	0.5	0.05
# feedforward layers (f)	1	3	1
batch size (n_b)	10	100	10

To measure the performance of a CNN or D-MPNN on a dataset, we used a three-fold nested cross-validation scheme, which works as follows. We select three disjoint subsets of the dataset, each containing 20% of the data, which we call outer test sets. For a given outer test set, we label the remaining 80% of the data as the outer train set. We then split the outer train set into an 80% inner train set, 10% inner validation set, and 10% inner test set. We apply 15 iterations of the TPE algorithm, where each iteration consists of 10 epochs of training using the inner train and validation sets, to identify an optimal set of hyperparameters on the inner test set. We train a model with these optimal hyperparameters on the outer train set and report performance on the outer test set. The outer train sets consist of 80% of a dataset, which is 16,000 configurations. Therefore, each CNN is trained on 16,000 different samples. Since each D-MPNN is trained on

†† See https://pytorch.org/.

‡‡ See https://github.com/rusty1s/pytorch_geometric and https://github.com/wengong-jin/chemprop.

‡ See https://github.com/ks8/glassML.

§ See https://hyperopt.github.io/hyperopt.

random square windows of these configurations for 10 epochs, each D-MPNN is trained on 160,000 different samples. This process is repeated for each of the three outer test sets, or folds, giving us three independent measures of performance for which we can report the mean and standard deviation.

The primary performance metric that we report, which we also use to select optimal sets of hyperparameters, is AUC, or area under the ROC (receiver operating characteristic) curve. Our machine learning models output a continuous value between 0 and 1 when making a prediction, so a threshold value is used to binarize the prediction. For example, if the threshold value is 0.5, outputs greater than 0.5 correspond to glass while outputs less than 0.5 correspond to liquid. The ROC curve is created by plotting the true positive rate against the false positive rate at different thresholds. AUC ranges from 0 to 1, where 0.5 indicates that classification is no better than random guessing and 1 indicates that classification is perfect across all thresholds. For comparison, we also report accuracy, or fraction of configurations labelled correctly using a single default threshold.

We trained the CNNs and D-MPNNs and performed hyperparameter optimization on NVIDIA Tesla V100 GPUs using Amazon Web Services.¶ As described in §7 of the Supplementary Information,† the training time for CNNs was less than 7.5 milliseconds per data sample, while the training time for D-MPNNs ranged from approximately 28 milliseconds to 260 milliseconds per data sample, depending on the values of the hyperparameters. Since one epoch of training corresponded to 16,000 data samples, the training time per epoch for CNNs was less than 2 minutes, while the training time per epoch for D-MPNNs ranged from approximately 7.5 minutes to 70 minutes. Since we trained each model for 10 epochs, training a CNN took approximately 20 minutes while training a D-MPNN ranged from approximately 75 minutes to 6.5 hours. Figure 8 in the Supplementary Information† shows D-MPNN training time for three representative sets of hyperparameters as a function of input graph size, which we varied by adjusting the window length l_{win} while keeping the other hyperparameters fixed.

3 Results and Discussion

3.1 CNNs

The CNNs classified liquid and glass configurations in datasets 1 through 6 with greater than 0.98 AUC, as shown in Figure 5 (see Figure 5 in the Supplementary Information† for a complementary accuracy report). These results indicate that CNNs are capable of classifying amorphous materials with no *a priori* information about particle interactions and no hand-crafted descriptors of local particle environments, even when the material structures are only subtly different. The CNNs are simply given basic geometric information - particle coordinates, rendered as an image - and they are able to classify configurations very accurately.

Moreover, CNNs optimized on one dataset were able to make accurate classifications of configurations from other datasets, as exhibited in Figure 6(a). All of the models generalized successfully to other datasets, but the models trained on the more difficult tasks (i.e. datasets 5 and 6) performed best across the board.

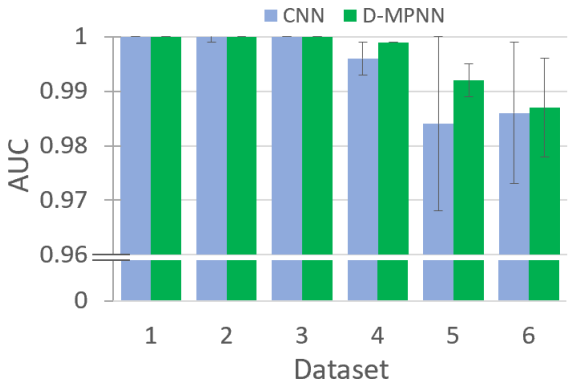

Fig. 5 CNN and D-MPNN average classification AUC for datasets 1 through 6 with error bars showing standard deviation. These average values were computed using the three-fold nested cross-validation scheme described in §2.5.

A central question that arises is how the CNNs are making classification decisions. Since a CNN can accurately identify liquid and glass configurations prepared at the same inherent structure energy (dataset 6), these algorithms are not simply computing energy as a means for classification. We tried several methods for interpreting the CNNs, including visualizing feature maps and examining whether specific geometrical patterns activated components of the network (see §4 in the Supplementary Information† for more details). These methods have been used in previous work to facilitate interpretation of neural networks.[32,34] However, none of these methods were successful at providing an interpretation. Further work is needed to answer this question.

Besides the challenge of interpreting the CNNs, there were several other issues. The CNNs do not incorporate rotational invariance, a symmetry that is present in our data, and they require each input image to have the same size, limiting their flexibility. Moreover, rendering particle configurations as images introduces an artificial radius to these particles which is not present in the underlying system. These issues motivated our study of message passing neural networks. As discussed below, we found that message passing neural networks overcome all of these challenges.

3.2 D-MPNNs

D-MPNNs classified liquid and glass configurations in datasets 1 through 6 with greater than 0.98 AUC. As shown in Figure 5, the D-MPNNs performed at least as well as the CNNs on all six datasets and had a higher average AUC than the CNNs on datasets 4 through 6 (see Figure 5 in the Supplementary Information† for a complementary accuracy report). These results show that, like CNNs, D-MPNNs are capable of classifying amorphous materials with no *a priori* information about particle interactions and no hand-crafted descriptors of local particle environments, even when the material structures are only subtly different. It is true that we created the particle graphs by connecting nearest neighbors, but we do not restrict the D-MPNN to focus on any specific local environments in the graph (different values for number of

¶ See https://aws.amazon.com/ec2/instance-types/p3/.

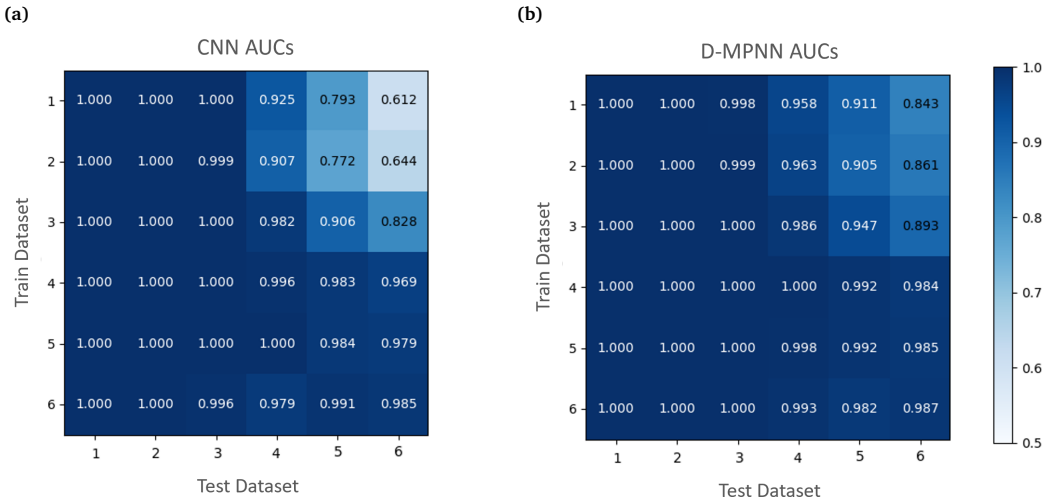

Fig. 6 These plots show how models trained on one dataset perform on all other datasets. In each of the plots in this figure, the number in row i, column j is the average classification AUC on dataset j of the optimal models trained on dataset i. For $i = j$ we report the average three-fold nested cross validation AUC. For $i \neq j$, we report average AUC using an outer test set of dataset j. The optimal models, three CNNs and three D-MPNNs for each dataset, were generated from the three-fold nested cross validation procedure described in §2.5. **(a)** shows average CNN AUCs and **(b)** shows average D-MPNN AUCs.

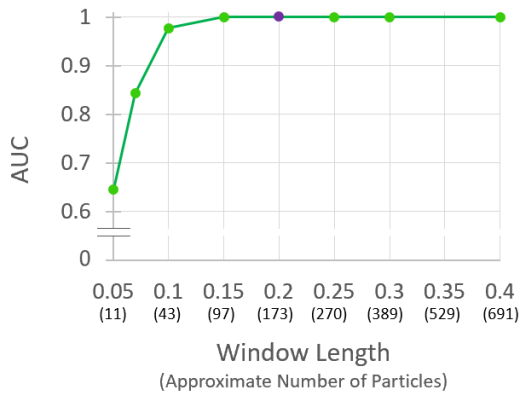

Fig. 7 D-MPNN AUC on an outer test set of dataset 1 with different values of l_{win} (number of particles). This model was trained on dataset 1 using only $l_{win} = 0.2$ (approximately 173 particles), which is highlighted in purple.

nearest neighbors (k) are explored during optimization).

Similar to the CNNs, the D-MPNNs trained on one dataset were able to make accurate classifications on configurations from other datasets, as exhibited in Figure 6(b). Again, the models trained on the more difficult tasks perform best across the board. All of the D-MPNN models, on average, generalize better than the CNN models.

Besides their superior performance, the D-MPNNs have several distinct advantages over the CNNs. Because information from neighboring particles in the graph is summed, the D-MPNNs are invariant to permutations and rotations of the graph, taking advantage of a natural symmetry in the system. The D-MPNNs also operate directly on the particle coordinates without introducing any unnecessary artifacts that might appear in an image representation. In addition, they are flexible and can process different graph sizes (number of particles). We took one of the optimal D-MPNNs trained on dataset 1, which was trained using $l_{win} = 0.2$ (approximately 173 particles per graph), and successfully performed inference on both smaller and larger graphs from dataset 1, as shown in Figure 7.

Perhaps the most significant advantage of D-MPNNs is that, when imbued with the self-attention mechanism described in §2.4.3, they are able to produce a clear interpretation of how they are making classification decisions. We illustrated this by training a D-MPNN on one of the outer train sets of dataset 1 with the self-attention mechanism in place and using an optimal set of hyperparameters. The model achieved an AUC of 0.995 on the corresponding outer test set. We then visualized the attention weights from configurations in the outer test set. Representative visualizations for glass and liquid configurations are shown in Figures 8(a) and 8(b), respectively.

There are clear differences in the graph structure of self-attention weights in these visualizations. The high attention weight edges form one large connected graph in the glass but multiple smaller disjoint graphs in the liquid. Several type B (blue) particles are connected to each other with high attention edges in the liquid, but none are connected in the glass. There are a larger number of type A (orange) particles that are isolated from high attention weight graphs in the liquid compared to the glass. Isolated type A particles and pairs of type B particles connected with high attention edges are highlighted in Figures 8(c) and 8(d). Also, all of the high attention edges in both liquid and glass are connected to at least one type B particle (there appear to be no isolated type B particles).

We also tested this model, which was trained using $l_{win} = 0.2$ (approximately 173 particles per graph), on larger configurations generated with $l_{win} = 0.3$ (approximately 389 particles per graph). The model successfully generalized to these larger configurations,

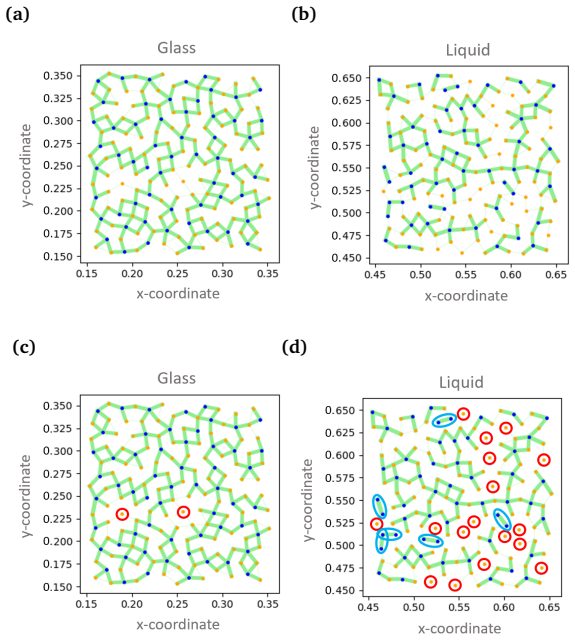

Fig. 8 Self-attention visualizations, with the attention weights computed on the $\mathcal{T}-1^{th}$ step of message passing. All connected particles in the graph are joined with a green line whose width is proportional to the magnitude of the attention weight. Some of the edges have such small attention weights that these lines are just barely visible. Note that each connected pair of particles actually has two edges, because the graph is directed. Here, we visualize the edge with the higher weight. **(a)** Glass configuration generated with $l_{win} = 0.2$. **(b)** Liquid configuration generated with $l_{win} = 0.2$. **(c)** and **(d)** show the same attention weight visualizations as **(a)** and **(b)**, respectively, but with isolated type A particles highlighted with red circles and pairs of type B particles connected by high attention edges highlighted with blue ellipses.

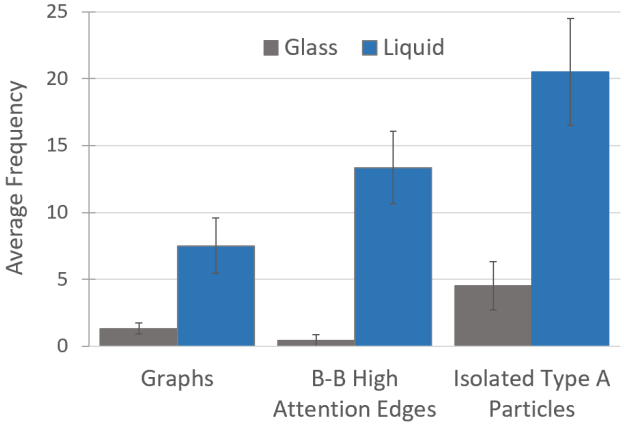

Fig. 9 Average number of disjoint graphs, B-B high attention edges, and isolated type A particles in an outer test set of dataset 1. High attention edges were determined with a hard cutoff. Here, a disjoint graph is a subset of the particles in a configuration connected by high attention edges. B-B high attention edges are edges connecting pairs of type B particles. Isolated type A particles are those not connected to a high attention edge.

achieving an AUC of 0.957, and yielded similar self-attention features, as shown in Figure 6 in the Supplementary Information.†

We quantified these features by computing the average number of disjoint graphs, average number of high attention edges connecting pairs of type B particles, and average number of isolated type A particles in dataset 1 configurations generated with $l_{win} = 0.3$, as shown in Figure 9. We confirmed that the differences in these values for glasses and liquids are statistically significant by using one-sided Wilcoxon signed-rank tests,‖ which returned p-values less than 1×10^{-4} .The number of isolated type B particles, not shown in the figure, was approximately 0 for all configurations, confirming that type B particles are nearly always adjoined to high attention edges. These features provide a clear interpretation of how the D-MPNN is making classification decisions: the network is focusing on type B particles and their relationship to nearby neighbors.

Not only can we clearly interpret the D-MPNN – a task that is typically very challenging in neural networks – but we can also derive three novel structural metrics from this interpretation that characterize glass formation. For any configuration of particles, we construct a graph by connecting every type B particle to its two nearest neighbors, and then we count: the number of disjoint graphs, the number of edges connecting pairs of type B particles, and the number of isolated type A particles. We use two nearest neighbors in this procedure because $k = 2$ was the optimal value used for the D-MPNN with self-attention, and we only connect type B particles to nearest neighbors because the D-MPNN with self-attention focused on edges connected to type B particles. Note that this new graph construction procedure does not necessarily yield graphs equivalent to the attention graphs generated by a D-MPNN. It is also different from the graph construction procedure described in §2.3.2 because only type B particles are connected to nearest neighbors.

Average values of the three metrics are plotted as a function of temperature for configurations from simulations with $t_{cool} = 2 \times 10^7$ in Figure 10. All three of these structural metrics exhibit a similar dependence on temperature. They decrease at rapid rates above T_g, but immediately below T_g, these rates change and the metrics decrease more slowly. We fit each metric to a linear regression model of the form

$$y_i = \alpha_0 + \alpha_1 T_i + \alpha_2 \mathbb{1} + \alpha_3 \mathbb{1} T_i + \varepsilon_i \tag{12}$$

using ordinary least squares, where T is temperature, $\mathbb{1}$ is an indicator variable with a value of 0 below T_g and a value of 1 above T_g, and ε is a random error term. The regressions returned positive-valued interaction coefficients α_3 with p-values less than 0.05, confirming that the difference in slopes above and below T_g is statistically significant for all three metrics. Figure 7 in the Supplementary Information† shows these metrics plotted separately as a function of temperature along with predictions from the linear regression models. Similar to the five-fold symmetric metric in Hu *et al.*, which was derived by hand, these metrics have a temperature dependence that describes the structural evolution of a liquid during glass formation.

‖ We used the function *wilcoxon* in the Python *scipy* package.

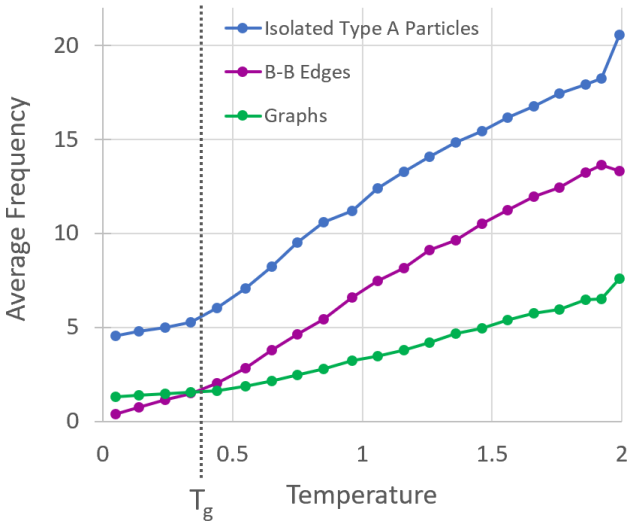

Fig. 10 Average number of disjoint graphs, edges connecting pairs of type B particles, and isolated type A particles in configurations from simulations with $t_{cool} = 2 \times 10^7$ at a variety of temperatures. Graphs of configurations were generated by connecting every type B particle to its two nearest neighbors. Averages were computed over 1,000 configurations at each temperature.

We were able to derive novel structural metrics for two-dimensional liquid and glass configurations directly from the self-attention features generated by a D-MPNN. The neural network generated these features based on raw particle coordinates alone and did not rely upon complex local descriptors, extensive spatial averaging, or a set of reference structures. This provides clear proof of concept that D-MPNNs are an effective tool not only for classifying amorphous materials but also for identifying structural features in complex systems.

4 Conclusions

In this work, we showed that CNNs and D-MPNNs are both effective tools for amorphous materials classification, as they can classify two-dimensional liquids and liquid-cooled glasses with greater than 0.98 AUC. We also demonstrated ways in which D-MPNNs are superior to CNNs in this context, including their ability to operate on raw particle data without introducing artifacts, to achieve better classification performance, to process configurations with different numbers of particles, and to provide a clear and quantifiable interpretation of the classification process. Using the interpretation that we extracted from D-MPNNs with a self-attention mechanism, we derived three novel structural metrics that characterize glass formation.

Moving forward, we believe that D-MPNNs could be applied to more difficult classification tasks that elude standard techniques. Specifically, we are interested in applying D-MPNNs to analyze other types of glassy materials, such as vapor-deposited glasses. A natural next step also includes using D-MPNNs to classify three-dimensional liquids and glasses, which could provide new insight into the local structure of these materials. We would be able to test the feasibility of classifying three-dimensional materials without changing the D-MPNN architecture by adding a third node feature representing the z-coordinate of particles and by changing the edge feature to be the Euclidean distance between pairs of particles in three dimensions. Depending on the outcome of this experiment, which would also require generating new datasets with particles in three dimensions, modifications to the neural architecture might be necessary. Recent studies have used graph-based neural networks to successfully classify three-dimensional shapes represented by a set of points, which suggests that classifying three-dimensional materials using D-MPNNs or similar graph-based architectures is feasible.[57] These architectures are invariant to rotations and translations, which are desirable properties for classifying three-dimensional structures. Moreover, new methods are being developed that are tailored specifically to operate on sets of points in three-dimensional space while respecting Euclidean symmetries.[58–60]

In our work we performed graph-level predictions, but it is also possible to use D-MPNNs to perform node-level predictions, which could be used to extract even more granular information about local structures around specific particles or to identify defect sites in materials. And finally, further steps can be taken to improve the interpretability of attention mechanisms in these networks, including incorporating node attention in addition to edge attention and using multi-headed attention or other attention pooling techniques. In our work, we manually analyzed self-attention visualizations, but as the field of neural network interpretation advances,[61] future work on developing tools that improve and automate this process would make deep learning analysis of amorphous material structures even more effective.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

The authors are grateful to Ashley Guo and Cody Bezik for useful discussions and comments and to Juan J. de Pablo for support. Shubhendu Trivedi's work was supported by the National Science Foundation under Grant No. DMS-1439786 while the author was in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during the non-linear algebra and computer vision programs. Risi Kondor was partially supported by DARPA HR00111890038 and this project used computational resources provided by NSF MRI 1828629.

Notes and references

1 P. J. Steinhardt, D. R. Nelson and M. Ronchetti, *Phys. Rev. B*, 1983, **28**, 784–805.
2 G. J. Ackland and A. P. Jones, *Phys. Rev. B*, 2006, **28**, 054104.
3 J. D. Honeycutt and H. C. Andersen, *J. Phys. Chem.*, 1987, **91**, 4950–4963.
4 W. F. Reinhart, A. W. Long, M. P. Howard, A. L. Ferguson and A. Z. Panagiotopoulos, *Soft Matter*, 2017, **13**, 4733–4745.
5 C. Dietz, T. Kretz and M. H. Thoma, *Phys. Rev. E*, 2017, **96**, 011301.
6 M. E. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation*, Oxford University Press, 2010.

7 P. Geiger and C. Dellago, *J. Chem. Phys.*, 2013, **139**, 164105.

8 A. Ziletti, D. Kumar, M. Scheffler and L. M. Ghiringhelli, *Nature Communications*, 2018, **9**, 2775.

9 J. Madsen, P. Liu, J. Kling, J. B. Wagner, T. W. Hansen, O. Winther and J. Schiotz, *Nature Communications*, 2018, **9**, 2775.

10 R. Kondo, S. Yamakawa, Y. Masuoka, S. Tajima and R. Asahi, *Acta Materialia*, 2017, **141**, 29–38.

11 Y. Liu, Q. Ye, L. Wang and J. Peng, *Bioinformatics*, 2018, **34**, 773–780.

12 L. Pu, G. Govindaraj, J.-M. Lemoine, H.-C. Wu and M. Brylinski, *PLoS Comput. Biol.*, 2019, **15(2)**, e1006718.

13 M. Spellings and S. C. Glotzer, *AIChE J*, 2018, **64**, 2198–2206.

14 P. Gasparotto, R. H. Meisner and M. Ceriotti, *J. Chem. Theory Comput.*, 2018, **14**, 486–498.

15 N. E. R. Zimmermann, M. K. Horton, Z. Jain and M. Haranczyk, *Front. Mater.*, 2017, **4**, 1–13.

16 N. Laanait, M. Ziatdinov, Q. He and A. Borisevich, *Adv. Struct. Chem. Imag.*, 2016, **2**, 14.

17 B. A. Helfrecht, P. Gasparotto, F. Giberti and M. Ceriotti, *Front. Mol. Biosci.*, 2019, **6**, 24.

18 Y. Liu, Q. Ye, L. Wang and J. Peng, *Bioinformatics*, 2018, **34**, 773–780.

19 M. Giulini and R. Potestio, *Interface Focus*, 2019, **9**, 20190003.

20 Y. C. Hu, F. X. Li, M. Z. Li, H. Y. Bai and W. H. Wang, *Nature Communications*, 2015, **6**, 8310.

21 D. R. Reid, I. Lyubimov, M. D. Ediger and J. J. de Pablo, *Nature Communications*, 2016, **7**, 13062.

22 D. M. Sussman, S. S. Schoenholz, E. D. Cubuk and A. J. Liu, *PNAS*, 2017, **114**, 10601–10605.

23 E. D. Cubuk, S. S. Schoenholz, E. Kaxiras and A. J. Liu, *J. Phys. Chem. B*, 2016, **120**, 6139–6146.

24 E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, B. D. Malone, J. Rottler, D. J. Durian, E. Kaxiras and A. J. Liu, *Phys. Rev. Lett.*, 2015, **114**, 108001.

25 X. Ma, Z. S. Davidson, T. Still, R. J. S. Ivancic, S. S. Schoenholz, A. J. Liu and A. G. Yodh, *Phys. Rev. Lett.*, 2019, **122**, 028001.

26 E. D. C. *et al.*, *Science*, 2017, **358**, 1033–1037.

27 S. S. Schoenholz, E. D. Cubuk, D. M. Sussman, E. Kaxiras and A. J. Liu, *Nature Physics*, 2016, **12**, 469–471.

28 S. S. Schoenholz, E. D. Cubuk, E. Kaxiras and A. J. Liu, *PNAS*, 2017, **114**, 263–267.

29 R. J. S. Ivancic and R. A. Riggleman, *Soft Matter*, 2018, **15**, 4548.

30 M. Harrington, A. J. Liu and D. J. Durian, *Phys. Rev. E*, 2019, **99**, 022903.

31 P. Ronhovde, S. Chakrabarty, D. Hu, M. Sahu, K. K. Sahu, K. F. Kelton, N. A. Mauro and Z. Nussinov, *Scientific Reports*, 2012, **2**, 1–6.

32 P. Suchsland and S. Wessel, *Phys. Rev. B*, 2018, **97**, 174435.

33 H. Munoz-Bauza, F. Hamze and H. G. Katzgraber, *Learning to find order in disorder*, e-print arXiv:cond-mat.dis-nn/1903.06993, 2019.

34 K. Mills and I. Tamblyn, *Phys. Rev. E*, 2018, **97**, 032119.

35 K. Y. *et al.*, *J. Chem. Inf. Model*, 2019, **59**, 3370–3388.

36 J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, *Neural Message Passing for Quantum Chemistry*, e-print arXiv:cs.LG/1704.01212, 2017.

37 P. C. S. John, C. Phillips, T. W. Kemper, A. N. Wilson, Y. Guan, M. F. Crowley, M. R. Nimlos and R. E. Larsen, *J. Chem. Phys.*, 2019, **150**, 234111.

38 T. S. Hy, S. Trivedi, H. Pan, B. M. Anderson and R. Kondor, *J. Chem. Phys.*, 2018, **148**, 241745.

39 R. Kondor, H. T. Son, H. Pan, B. Anderson and S. Trivedi, *Covariant Compositional networks for learning graphs*, e-print arXiv:cs.LG/1801.02144, 2018.

40 K. T. Schutt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko and K.-R. Muller, *J. Chem. Phys.*, 2018, **148**, 241722.

41 P. B. Jorgensen, K. W. Jacobsen and M. N. Schmidt, *Neural Message Passing with Edge Updates for Predicting Properties of Molecules and Materials*, e-print arXiv:stat.ML/1806.03146, 2018.

42 S. Kearnes, K. McCloskey, M. Berndl, V. Pande and P. Riley, *Molecular graph convolutions: moving beyond fingerprints*, e-print arXiv:stat.ML/1603.00856, 2016.

43 T. Xie and J. C. Grossman, *Phys. Rev. Lett.*, 2018, **120**, 145301.

44 W. Kob. and H. C. Andersen, *Phys. Rev. E*, 1995, **51**, 4626–4641.

45 E. Bitzek, P. Koskinen, F. Gahler, M. Moseler and P. Gumbsch, *Phys. Rev. Lett.*, 2006, **97**, 170201.

46 J. Helfferich, I. Lyubimov, D. Reid and J. J. de Pablo, *Soft Matter*, 2016, **12**, 5898–5904.

47 G. Cybenko, *Math. Control Signal Systems*, 1989, **2**, 303.

48 I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

49 C. D. Lehman, A. Yala, T. Schuster, B. Dontchos, M. Bahl, K. Swanson and R. Barzilay, *Radiology*, 2018, **290**, 1.

50 S. Dieleman, K. W. Willett and J. Dambre, *Rotation-invariant convolutional neural networks for galaxy morphology prediction*, e-print arXiv:astro-ph.IM/1503.07077, 2015.

51 N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, *Journal of Machine Learning Research*, 2014, **15**, 1929–1958.

52 D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, e-print arXiv:cs.LG/1412.6980, 2017.

53 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention Is All You Need*, e-print arXiv:cs.CL/1706.03762, 2017.

54 D. Duvenaud, D. Maclaurin, J. A.-I. R. Gomez-Bombarelli, T. Hirzel, A. Aspuru-Guzik and R. P. Adams, *Convolutional Networks on Graphs for Learning Molecular Fingerprints*, e-print arXiv:cs.LG/1509.09292, 2015.

55 K. Swanson, L. Yu, J. W. C. Fox and T. Lei, *Building a Production Model for Retrieval-Based Chatbots*, e-print arXiv:cs.CL/1906.03209, 2019.

56 J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, *Algorithms for Hyper-Parameter Optimization*, Advances in Neural Infor-

mation Processing Systems 24, 2011.

57 Y. Zhang and M. Rabbat, *A Graph-CNN for 3D Point Cloud Classification*, e-print arXiv:cs.CV/1812.01711, 2018.

58 R. Kondor and S. Trivedi, *On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups*, e-print arXiv:stat.ML/1802.03690, 2018.

59 C. R. Qi, L. Yi, H. Su and L. J. Guibas, *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*, e-print arXiv:cs.CV/1706.02413, 2017.

60 M. Weiler, M. Geiger, M. Welling, W. Boomsma and T. S. Cohen, *3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data*, Advances in Neural Information Processing Systems 31, 2018.

61 R. Ying, D. Bourgeois, J. You, M. Zitnik and J. Leskovec, *GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks*, e-print arXiv:cs.GL/1903.03894, 2019.

We use deep learning to automatically classify liquid and glass structures and to derive novel metrics that describe glass formation.