Supporting Information for

Bandgap Prediction by Deep Learning in Configurationally Hybridized Graphene and Boron Nitride

Yuan Dong[1ξ], Chuhan Wu[2ξ], Chi Zhang[1], Yingda Liu[2], Jianlin Cheng[2*], Jian Lin[1*]

[1]Department of Mechanical & Aerospace Engineering

[2]Department of Electrical Engineering & Computer Science, University of Missouri, Columbia, Missouri 65211, USA

[*]E-mail: LinJian@missouri.edu (J. L.) or chengji@missouri.edu (J. C.)
[ξ] Authors contributed equally to this work.

Supplementary Notes

Supplementary Note 1. Support Vector Machine (SVM)

The SVM used in this work was developed by Corinna Cortes and Vapnik in 1993 [1], and evolved into support vector regression machines later [2]. It was one of the hottest machine learning methods before the deep learning made a huge breakthrough recently. In this work, we first flatten the original input training data to a column vector (for example, the 4×4 data was flatten to be a 16×1 vector data). These column vectors were used as the SVM input. We selected Polynomial [3] as the regression kernel which can transform the data from low-dimension space to high-dimension space.

Supplementary Note 2. VGG16 Convolutional Network (VCN)

Convolutional neural networks (CNNs) [4], state-of-art deep learning methodologies which combine convolution layers and fully connected (FC) layers into one model, are used widely in the field of computer vision [5], natural language processing [6], self-driving cars [7], bioinformatics [8] and so on. For the raw 2D data, convolution can extract the feature not only from the element in

the input data but also their neighbor elements' information. This is a feature detector that's useful in full perspectives of the 2D data. Convolution process converts them into feature map and transforms it to later layers. For instance, when applied to image data study, it means detecting the edge or textures of the images. In this study, the structure information of the doped graphene we use can be set as 2D data (Channel [might: 1]*Length*Width). Each pair of atoms inside its structure might infect its neighbor atoms so they collectively determine bandgaps of the whole structure. We firstly built a neural network by following the VGG16 structure, which was widely used in computer vision. This model has 16 convolution layers, one global-flatten layer, three FC layers and an output layer. We call it VGG16 convolutional network (VCN). This network was further modified into other two novel networks, residual convolutional network (RCN), and concatenate convolutional network (CCN). We will elaborate them in the following section.

2.1 Convolution layer

The convolution layer of CNN that it does convolution operation toward its input and output forms a 2D feature map. The convolution process equation is shown in the following.

$$(h_k)_{ij} = (W_k * x)_{ij} + b_k \quad (1)$$

k is the index of the k^{th} feature map. W_k and b_k are the weight and bias of k^{th} feature map. $(h_k)_{ij}$ is the value of the output for the neuron in the k^{th} feature map in position of (i; j).

2.2 Global Max-Pooling layer

This layer extracts the biggest element, which is also the most "important" and "significant" feature, from each channel of the previous layer. It offers the models the most convincing

evidence to predict the bandgaps of the investigated systems. Since the feature map in each channel only outputs their biggest element, the output volume will “lost” one dimension from the input volume. Since this layer can convert 2D feature data into 1D vector feature data, it also acts as a "bridge layer" to link the previous convolution layer to the next Fully Connected Layer of the model.

2.3 Fully connected layer

The fully connected (FC) layer is a classical neural network layer where all the neurons from the previous layers are connected. It can be expressed as:

$$z_i = \sum_{k=1}^{N_1} w_{ki} x_k + b_i \qquad (2)$$

where Z_i is the i^{th} neuron’s output, N_1 is the neurons contained in its previous layer, w_{ki} is the weight of k^{th} neuron from previous layer, and b is the bias of the current layer.

2.4 Batch normalization layer

After each convolution and FC layer, there is a batch normalization layer. Batch normalization [9] is an useful process that it can erase the uncertainty of the hidden layer, and reduce the influence of internal covariate shift which changes the distribution of network activations during updating the parameters when training the network. Meanwhile it accelerates the training speed for the networks. The math of batch normalization is shown in the following:

$$\mu = \frac{1}{m}\sum_i z^{(i)},$$
$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2,$$
$$Z_{norm}^{(i)} = \frac{(z^{(i)} - \mu)}{\sqrt{\sigma^2 + \varepsilon}},$$
$$\tilde{Z}^{(i)} = \gamma \cdot Z_{norm}^{(i)} + \beta \qquad (3)$$

Where $z^{(1)}, z^{(2)}, \ldots, z^{(m)}$ are the outputs of each layer; ε is introduced to avoid dividing by zero; β, γ are the learnable parameters of the model. After we replace $z^{(i)}$ with $\tilde{z}^{(i)}$ and push it into forward and backward propagation, the original hyper parameter bias $b^{(i)}$ will be replaced with β.

2.5 Activation function of exponential linear units (ELU)

We used exponential linear units (ELU) [10] as the activation function in each layer. The ELU is shown as follows when α is larger than zero.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \le 0 \end{cases}, \quad f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ f(x) + \alpha & \text{if } x \le 0 \end{cases} \qquad (4)$$

ELU tolerates the negative values when they are close to zero, thus it decreases shifting effect of the bias. The rectified linear unit (ReLU) activation function is used in previous ML works [11]. However, in our network, the ReLU function could cause serious problem because it will always give zero if neuron units after ReLU activation converge to zero (called "dying ReLU" problem, http://cs231n.github.io/neural-networks-1/#actfun). So we choose to use ELU as our activation function and this problem is removed.

Supplementary Note 3. Residual convolutional network (RCN)

The deep training of the neural network helps it to think deeper and can extract more details and important features from the training data. However, as the network becomes deeper and deeper, it will reach to a maximum point of performance, and thereafter encounter with degradation problem. The prediction accuracy gets saturated and degrades rapidly [3]. The residual network is introduced which converts each convolution layers to residual blocks [3]. It could help to solve the degradation problem. **Fig. S2** shows the structure of a residual block. This block has one convolution block and two identity blocks. The equation is shown as:

$$a^{[l+2]} = g(Z^{[l+2]} + a^{[l]}) = g(\omega^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]}) \quad (5)$$

$Z^{[l]}$: the input of l^{th} activation layer
$\omega^{[l]}$: the parameters of l^{th} convolutional layer
$b^{[l]}$: the bias of l^{th} convolutional layer
$a^{[l]}$: the input of residual block

When degradation problem occurs, the term of $(\omega^{[l+2]} a^{[l+1]} + b^{[l+2]}) = 0$, and this formula will become $a^{[l+2]} = g(a^{[l]})$, it can help to prevent the network from getting hurt by degradation and keep the network's strong robustness and its stable performance. Because the size of input is equal to the output of the residual block, the original network adds the parameters from *a*[l] and Z[l+1] in element wise.

Supplementary Note 4. Concatenate Convolutional Network (CCN)

The CCN combined featured advantages from GoogleNet [12] and DenseNet [13]. Unlike RCN which "adds" feature maps in element-wise, this network concatenates the layer from input and output them pass though [l+α] activation layer together (**Fig. S3**). The equation of concatenate operation is:

$$\begin{aligned}\alpha^{[l+2]} &= C(g(Z^{[l+2]}), g(\alpha^{[l]}))_{\text{axis=filter\_num}} \\ &= C(g(\omega^{[l+2]}\alpha^{[l+1]} + b^{[l+2]}), g(\alpha^{[l]}))_{\text{axis=filter\_num}}\end{aligned} \quad (6)$$

where C(A,B) is the Concatenate function (Concatenate axis= Channels/filter numbers axis),$\alpha^{[l]}$ is the output of l^{th} convolutional layer. When degradation problem occurs, it means

$$(\omega^{[l+2]}\alpha^{[l+1]} + b^{[l+2]}) = 0 \quad (7)$$

and the equation becomes

$$\begin{aligned}\alpha^{[l+2]} &= C(g(0), g(\alpha^{[l]}))_{\text{axis=filter\_num}} \\ &= C(0, g(\alpha^{[l]}))_{\text{axis=filter\_num}}\end{aligned} \quad (8)$$

The input's original information still maintains in the output, so it can prevent the network from degradation. Meanwhile this process increases the volume and the trainable parameters of the data. They give more evidences and information for model to analyze and help it thinking "deep" and "thoroughly". Like the identity and convolution blocks of residual network, we introduced the small unit into the CCN: **Concatenation Block** (**Fig. S3B**). This block has three convolution layers: two normal convolution layers and one bottleneck layer inside of them. Bottleneck layer is a convolution layer with less filter number. Its filter size is 1×1 and convolution stride is 1. It reduces the number of parameters needed to update the neural without degrading the performance.

Supplementary Note 5. Evaluation metrics

The performance of prediction accuracy can be evaluated through many metrics. In this work we used coefficient of correlation (R), explained variance (R^2), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Their formulas are listed below:

$$R = \frac{\sum_{i=1}^{N}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sqrt{\sum_{i=1}^{N}\left(X_i - \bar{X}\right)^2}\sqrt{\sum_{i=1}^{N}\left(Y_i - \bar{Y}\right)^2}} \quad (9)$$

$$\mathrm{MAE} = \frac{1}{N}\left|X_i - Y_i\right| \quad (10)$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(X_i - Y_i\right)^2} \quad (11)$$

where X is the DFT value and Y is the ML predicted value. $\bar{X}$ is the mean value of X, and analogously for $\bar{Y}$. N is the number of values used from comparison. The MAE and RMSE could be further normalized by X to express them as the fractional error.

$$\mathrm{MAE_F} = \frac{1}{N}\left|\frac{X_i - Y_i}{X_i}\right| \quad (12)$$

$$\mathrm{RMSE_F} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{X_i - Y_i}{X_i}\right)^2} \quad (13)$$

Supplementary Figures

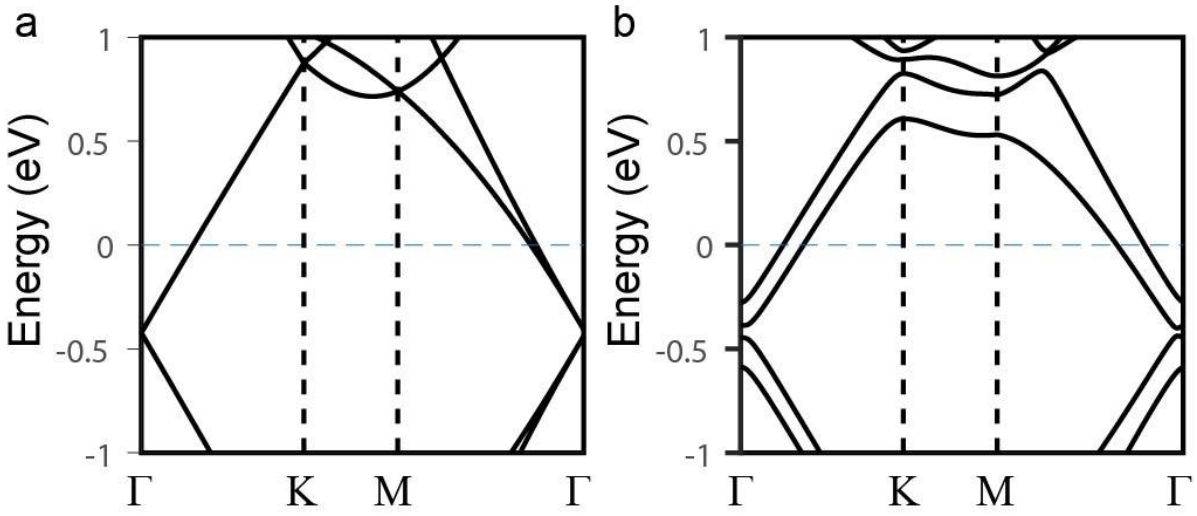

Fig. S1 Band structure of 6 × 6 system (a) pure graphene (b) with 3 at.% BN concentration.

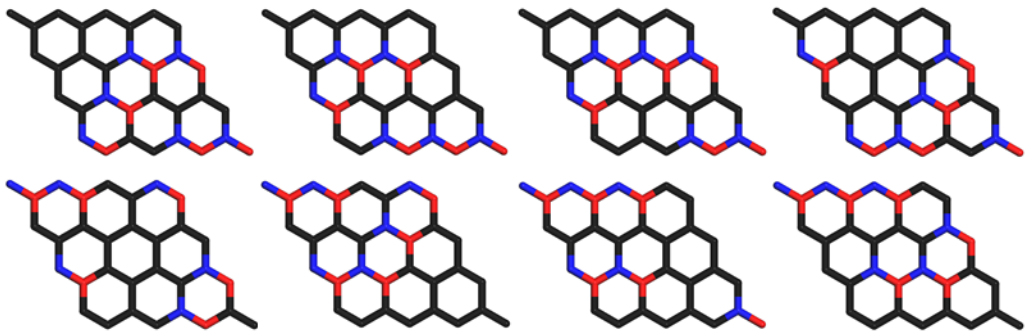

Figure S2. Examples of equivalent structures obtained by translating the particular structures along their lattice axis or inversion around their symmetry axis for 4 × 4 system.

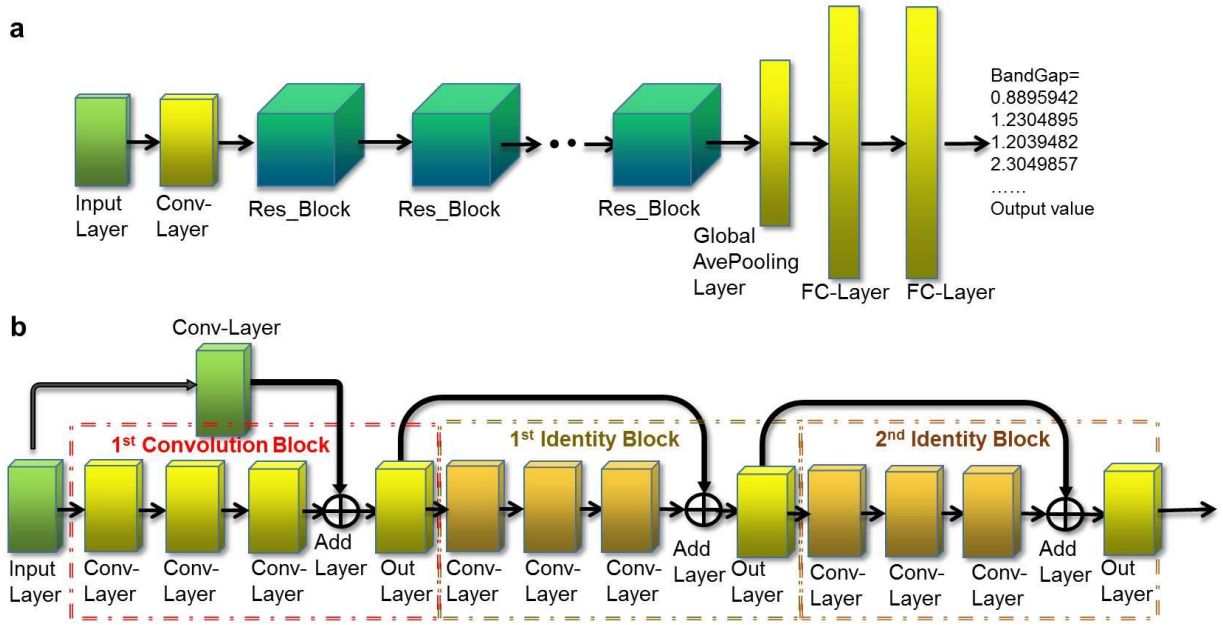

Figure S3. The structure of residual convolutional network (RCN). (a) The whole architecture of the RCN. (b) The detail about one example residual block (Res_Block).

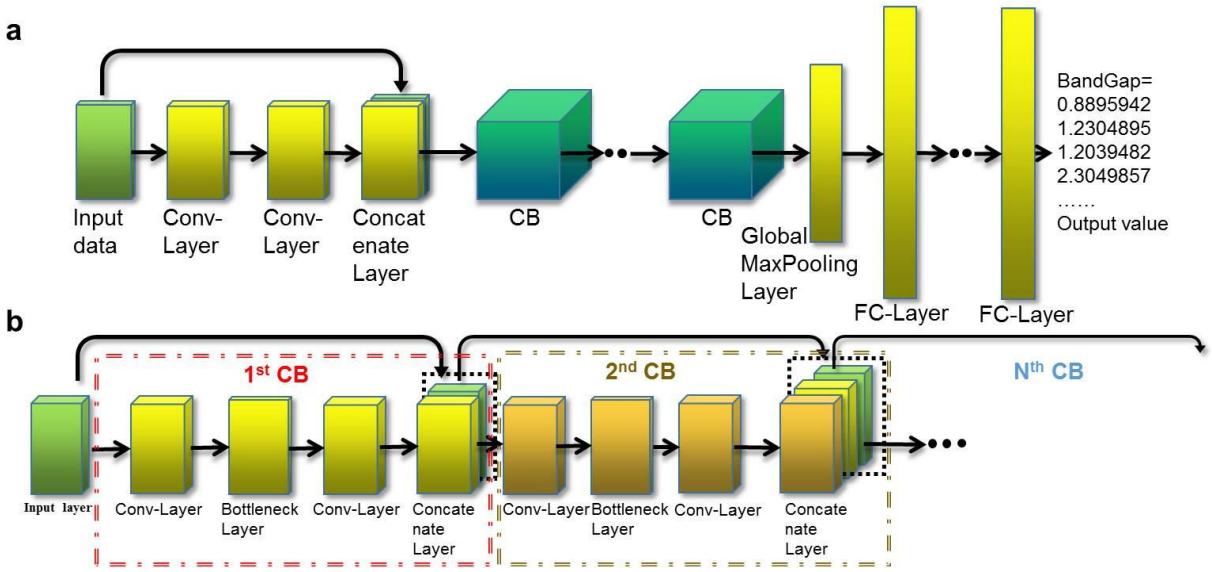

Figure S4. The structure of concatenate convolutional network (CCN). (a) Overall architecture of the CCN. (**b**) Details about the connection between two concatenation blocks (CBs) in the CCN.

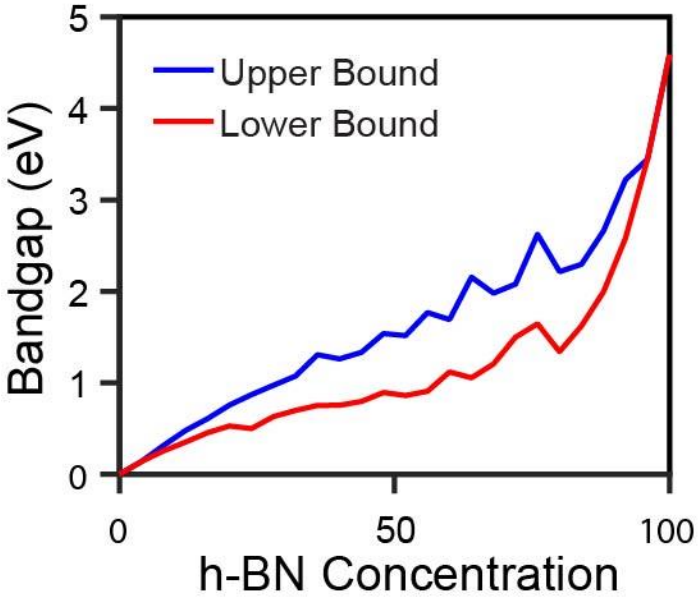

Fig. S5 Bandgaps vs. h-BN concentration in training dataset of 5 ×5 systems.

Supplementary Tables

Table S1. Structure and hyperparameters of VGG16 convolutional network (VCN).

Input (Graphene Structure)
Convolution Layer 1 [filter number=50, filter size=(3,3), stride=(1,1), activation='elu'] Batch Normalization Layer 1
Convolution Modules 2-12 Convolution Modules= {Convolution Layer [filter number=50, filter size=(3,3), stride=(1,1), activation='elu'] Batch Normalization Layer}
Global-Max Pooling Layer
Dropout Layer 1 (keep probability=0.6)
Fully-Collected Layer 1 (neuron unit=64, activation="elu")
Fully-Collected Layer 2 (neuron unit=64, activation="elu")
Fully-Collected Layer 3 (neuron unit=48, activation="elu")
Dropout Layer 2 (keep probability=0.7)
Output Layer (neuron unit=1, activation="elu")
Output (Bandgap Value)

Table S2. Structure and hyperparameters of residual convolutional network (RCN).

Input (Graphene Structure)
Convolution Layer 1 [filter number=24, filter size=(3,3), stride=(1,1), activation='elu'] Batch Normalization Layer 1
Residual Block 1: {Convolution Block 1 [filter number 1=24, filter number 2=24, filter number 3=48] Identity Block 1 [filter number 1=24, filter number 2=24, filter number 3=48] Identity Block 2 [filter number 1=24, filter number 2=24, filter number 3=48]}
Residual Block 2: {Convolution Block 2 [filter number 1=48, filter number 2=48, filter number 3=64] Identity Block 3 [filter number 1=48, filter number 2=48, filter number 3=64] Identity Block 4 [filter number 1=48, filter number 2=48, filter number 3=64]}
Residual Block 3: {Convolution Block 3 [filter number 1=64, filter number 2=64, filter number 3=72] Identity Block 5 [filter number 1=64, filter number 2=64, filter number 3=72] Identity Block 6 [filter number 1=64, filter number 2=64, filter number 3=72]}
Residual Block 4: {Convolution Block 4 [filter number 1=72, filter number 2=72, filter number 3=84] Identity Block 7 [filter number 1=72, filter number 2=72, filter number 3=84] Identity Block 8 [filter number 1=72, filter number 2=72, filter number 3=84]}
Global Max-Pooling Layer
Fully-Collected Layer 1 (neuron unit=48, activation="elu")
Fully-Collected Layer 2 (neuron unit=24, activation="elu")
Output Layer (Neuron unit:1, activation="elu")
Output (Bandgap Value)

Table S3. Structure and hyperparameters of convolution blocks inside residual convolutional network (RCN).

Input	Input
Convolution Layer 1 (filter number 1, filter size=(1,1), stride=(1,1), activation='elu')	Convolution Layer 1 (filter number 3, filter size=(1,1), stride= (1,1), activation='elu')
Convolution Layer 2 (filter number 2, filter size=(3,3), stride=(1,1), activation='elu')	
Convolution Layer 3 (filter number 3, filter size=(1,1), stride=(1,1), activation='elu')	
Add Layer (activation='elu')	
Output Layer (Neuron unit=1, activation="elu")	

Table S4. Structure and hyperparameters of identity blocks inside residual convolutional network (RCN).

Input	Input
Convolution Layer 1 (filter number 1, filter size:(1,1), stride=(1,1), activation='elu')	
Convolution Layer 2 (filter number 2, filter size:(3,3), stride=(1,1), activation='elu')	
Convolution Layer 3 (filter number 3, filter size:(1,1), stride=(1,1), activation='elu')	
Add Layer (activation='elu')	
Output Layer (Neuron unit=1, activation="elu")	

Table S5. Structure and hyperparameters of concatenate convolutional network (CCN).

Input (Graphene Structure)	Input (Graphene Structure)
Convolution Layer 1 [filter number=60, filter size=(3,3), stride=(1,1), activation='elu'] Batch Normalization Layer 1	
Convolution Layer 2 [filter number=60, filter size=(3,3), stride=(1,1), activation='elu'] Batch Normalization Layer 2	
Concatenate Layer (activation='elu')	
Concatenation Blocks 1-15	
Global Max-Pooling Layer	
Drop Out Layer1 (keep probability=0.6)	
Fully-Collected Layer 1 (neuron unit=64, activation="elu")	
Fully-Collected Layer 2 (neuron unit=48, activation="elu")	
Fully-Collected Layer 3 (neuron unit=24, activation="elu")	
Drop Out Layer2 (keep probability=0.8)	
Output Layer (neuron unit=1, activation="elu")	
Output (Bandgap Value)	

Table S6. Structure and hyperparameters of concatenation blocks inside concatenate convolutional network (CCN).

Input	Input
Convolution Layer 1 (filter number=60, filter size=(3,3), stride=(1,1), activation='elu')	
Convolution Layer 2 ("Bottleneck Layer") (filter number=32, filter size=(1,1), stride=(1,1), activation='elu')	
Convolution Layer 3 (filter number=60, filter size=(3,3), stride=(1,1), activation='elu')	
Concatenate Layer (activation='elu')	
Output Layer (Neuron unit=1, activation="elu")	

Table S7. Statistics of predicted bandgaps at different BN concentration for 5 × 5 supercell systems. .

	MAE (eV)	MAE_F	RMSE (eV)	$RMSE_F$	R^2
Low Concentration (< 33 at.%)					
VCN	0.06	7.04%	0.07	9.87%	0.7990
RCN	0.07	8.4%	0.08	11.1%	0.7030
CCN	0.13	14.9%	0.15	18.6%	0.6407
Medium Concentration (33 at.% ~ 66 at.%)					
VCN	0.06	5.27%	0.08	6.56%	0.7924
RCN	0.06	5.62%	0.08	7.17%	0.7899
CCN	0.14	12.9%	0.16	14.6%	0.8117
High Concentration (>66 at.%)					
VCN	0.09	5.13%	0.12	6.29%	0.8921
RCN	0.10	6.15%	0.13	7.70%	0.7752
CCN	0.12	7.67%	0.14	9.01%	0.8451

Supplemental References

1. Cortes C, Vapnik V, Support-vector networks, Machine Learning, 20(3), 273-297, (1995).

2. Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V, Support vector regression machines, Advances in Neural Information Processing Systems 9, 9, 155-161, (1997).

3. Opsomer JD, Ruppert D, Fitting a bivariate additive model by local polynomial regression, The Annals of Statistics, 186-211, (1997).

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

5. Krizhevsky A, Sutskever I, Hinton GE, ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems, 1097-1105, (2012).

6. Collobert R, Weston J, A unified architecture for natural language processing: deep neural networks with multitask learning, Proceedings of the 25th International Conference on Machine learning, 160-167, (2008).

7. Bojarski M, *et al.*, End to end learning for self-driving cars, arXiv preprint arXiv:1604.07316, (2016).

8. Zeng H, Edwards MD, Liu G, Gifford DK, Convolutional neural network architectures for predicting DNA–protein binding, Bioinformatics, 32(12), i121-i127, (2016).

9. Ioffe S, Szegedy C, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167, (2015).

10. Clevert D-A, Unterthiner T, Hochreiter S, Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint arXiv:1511.07289, (2015).

11. Carrasquilla J, Melko RG, Machine learning phases of matter, Nature Physics, 13(5), 431, (2017).

12. Szegedy C, *et al.*, Going deeper with convolutions, 2015 IEEE Conference on Computer Vision and Pattern Recognition, 1-9, (2015).

13. Huang G, Liu Z, van der Maaten L, Weinberger KQ, Densely connected convolutional networks, 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), 2261-2269, (2017).