**Title: [未完待续]**

**Abstract**

**[未完待续]**

**Introduction**

**[未完待续]**

**Materials and Methods**

**1.0 DFT algorithm**

**[Concept of DFT algorithm]**

**DFT的原理，为什么可以预测石墨烯的这些性质，速度是不是很慢**

**[Data-Preprocessing]**

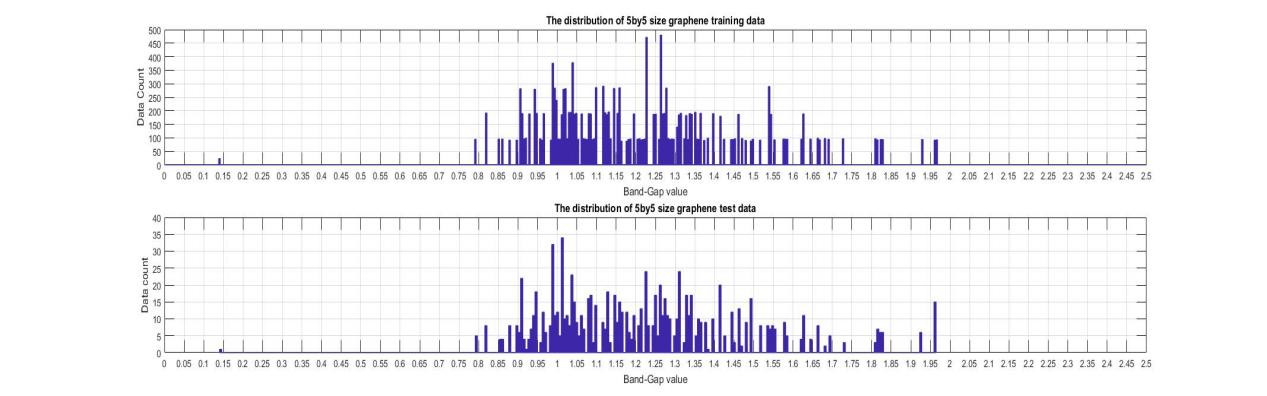
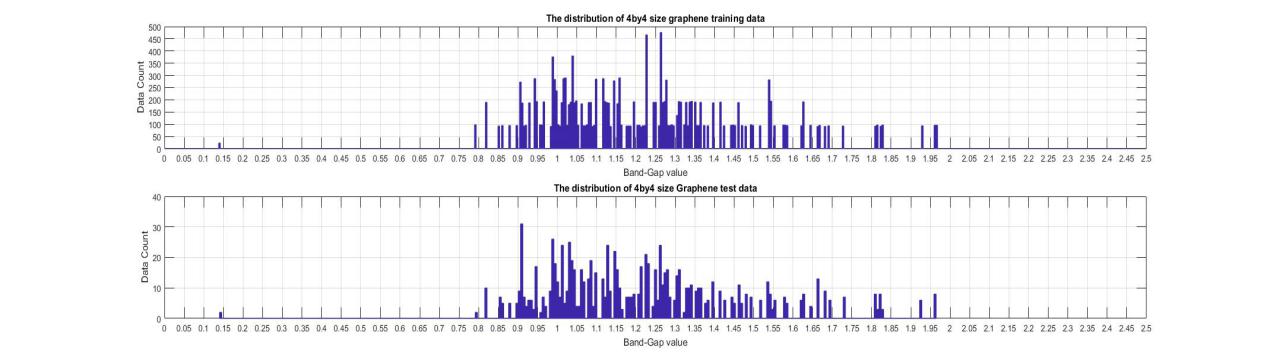
**我们命名的规则，0是碳原子，1是其他原子 ，怎么把一个4\*4或者5\*5的数据转换成一个数字，传递到后面的机器学习中来用.**

* 1. **Datasets**

Just like the method we have described. Based on our nomenclatures of how to name for a special type of graphene structure, we randomly selected the number among the ranges of each size (E.g for 4by4 structure, their have 0-2^16 kind of different structures in total ) them convert these number to special graphene structures data. Every value have the same probability to be selected which means that the original dataset is universality and it can cover almost all common structures that Graphene can form. It's very important for machine learning and it help us to verify it's performance conveniently.

**[Train, Cross-validation, Test /ExtraTest(可能这么称呼)Dataset ]**

We have created 2 kinds of Graphene Structure: 4by4 and 5by5. 4by4 have 14018 data and 5by5 have 19017 data in total. The distribution of their Band Gap values has shown in Fig.1. The majority of their values converge in 0.8 - 2.0. While we randomly choose and split 1000 data among original dataset and combine them as test dataset, these datasets only use in the step of evaluating the performance of our method and don't throw into any training process. The distribution of test data and training data is shown in Fig.1. For neural network methods we also split 10% of remaining data as cross-validation data which helps us to know the training process of these models, then the other 90% set as training data. Meanwhile for SVM-Regression, the whole remain data set as training data. After we have trained models, we also created 300 data which haven't occurred in our original data, we set them as Extra-Test dataset. It gives us a convincing evidence when we test our models.



**Fig.1 The distribution of Train(when come into the neural network training, it will split to train/cross-validation dataset )/Test dataset by different size of data (1)4by4 data (2)5by5 data**

1. **Method**

**2.1 Support Vector Machine**

**[Graphene Support vector regression machine]**

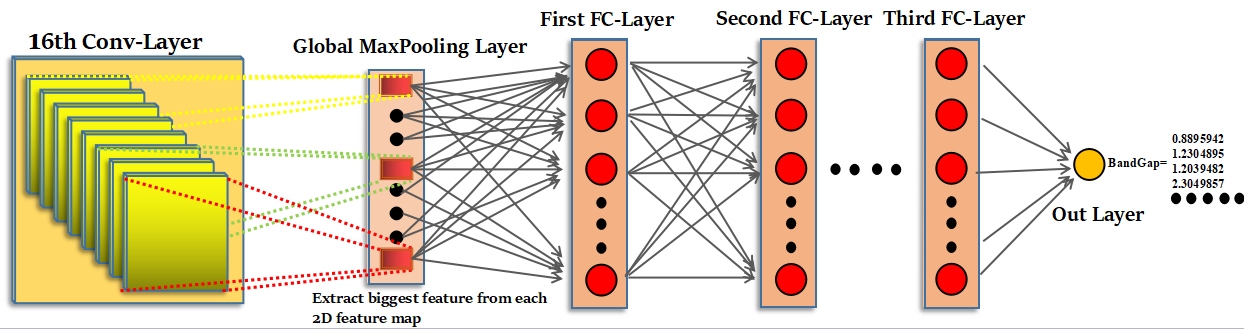
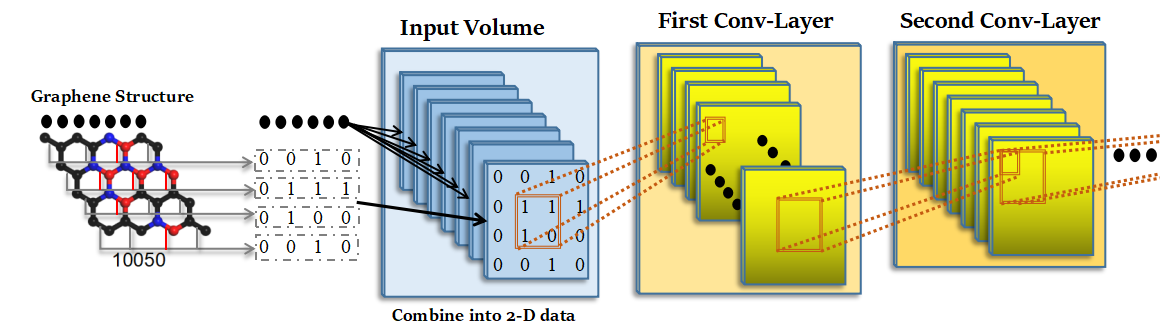
Support vector regression machine also named as SVM. Its original concept was first introduced by Vladmir N. Vapnik in 1963. It changed and developed as time when by and the most common and standard one we used now was developed by Corinna Cortes and Vapnik in 1993 [1], after a while some of them developed its application for continuous problem: SVR [2] which is one of the hottest machine learning methods before the deep learning made a huge breakthrough recently.

For our problem, we first flatten the original input training data to a column vector (E.g for 4\*4 data, flatten them to be a 16\*1 vector data), them use these column vectors as SVR input. We selected Polynomial as our SVR kernel that it can transform data from low-dimension space to high-dimension space easily.

**2.2 Deep Network**

**[Graphene Convolutional Network]**

Convolutional neural networks (CNNs), a state-of-art deep learning method which combines convolution layer and FC (fully connected) layer into one model are used widely in the field of Computer Vision, bioinformatic and so on. It first introduced by LeCun et.al in 1998 and presented a famous model LeNet-5[8] which can use to recognize hand-written numbers data them many new models have developed and performance got better and better. For the raw 2-D data, convolution can extract the feature from not only the element from the input data but also their "neighbor elements' " information, a feature detector that's useful in one part of the 2D data is probably useful in another part of the 2D data.Convolution process converts them into feature map and transforms it to later layers. Mean. For image data, it might mean detecting edge or texture while for protein data it extracts the interaction from near-by amino-acid. In the meanwhile the structure of Graphene data we use can be set as 2-D data (Channel [might: 1]\*Length\*Width), each atom inside its structure might infect its neighbor atoms while the part of atom clusters have the huge influence toward the whole structure band-gap value. Compare to other neural network, the convolution network can match Graphene's learning process properly.First we use our own method which has been described before to convert the Graphene structure to 2-D 0/1 matrices, them combine all data as our input, normalize the data of setting 4by4 data as an example, the training data volume should be 13018\*4\*4. Them we push these 2-D data into the convolution layer in the back. The CNN model we use is shown in Fig.4.



**Fig.4: the structure of Graphene Convolutional network.(Setting 4by4 size or Graphene data as an example) , each layers’ output will have a batch normalization layer to do normalize the output. It isn’t**

**included in this image.**

Based on our experiment, We find that unlike using traditional famous CNN model like AlexNet, VGG16&VGG19 which are widely used in computer vision, the model build and configured by us has the best performance. This model has 16 Convolution layers, one global-flatten layer, three full-connected layer and an output layer. We called it Graphene convolutional network.

**[Convolution layer]**

The kernel layer of CNN that it does convolution operation toward its input and output a 2-d feature map . The convolution process equation shown like that:

 [9]

[ k : the index of the  feature map ;  ：the weight and the bias of  feature map .  : the value of the output for the neuron in the  feature map with position of (i; j). \* ：Convolution operation symbol]

The activation we useafter the layer is rectified-linear unit(ReLU) (Krizhevsky et al., 2012).

**[Global Max pooling layer]**

This layer extracts the biggest element from each channel of the former layer. Which means that it can extract the most "important" and "significant" feature. It gives models the most convincing evidence to predict the Band-gap of the Graphene data. Because each channel’s feature map only output the biggest element of them, the output volume will “lost” one dimension compare to the input volume. Since This layer can convert 2-D feature data into 1-D vector feature data. As a "bridge layer" to link former convolution layer to next fully-connected layer of the model.

[**Fully-Connected layer]**

It’s the classical neural network layer that all neuron from the former layer connects to the later FC layer. Its Equation shown like that:



[ means the  neuron’s output while  means the weight from former layer’s  neuron and means the bias of this layer. The former layer has  neurons ]

The activation we use after the layer is rectified-linear unit(ReLU) .

**[Batch Normalization layer]**

While after each convolution and FC layer, we introduce Batch Normalization after them. Batch Normalization, first introduced by Sergey Loffe et.al [10] is an useful process that it can erase the uncertainty of the hidden layer, reduce the influence of internal covariate shift which changes the distribution of network activations when the parameters are updating when we training the network. Meanwhile it accelerates the network’s training speed. The Equation is shown like that:

[If each layer output is ]

****

[ε : Introduce it to avoid deviding by zero ; β,γ : Leanable parameter of model ; after that we replace  with  and push it into forward & backward propagation, the original hyper parameter bias will be removed because subtract with the mean value, and replace it with  ]

**[Graphene Residual Network]**

CNN has huge advantage of training these data. If we training the network more deep, it might think deeper and can consider more details and extract more important features from the training data. However if network become too deep its performance will reach to a maximum point, and degradation problem which causes accuracy get saturated and degrade rapidly will occur [3]. However, if we introduced the concept of Residual Network and converted each convolution layers to Residual Block [3]. It will help to solve this problem, Fig4 show what the residual block look like, This block has two convolution layers. The equation shown like that:

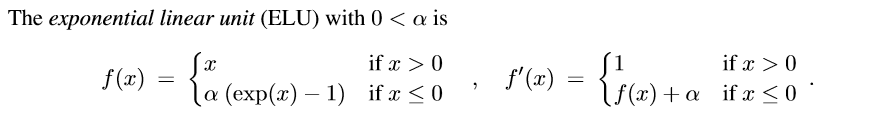




When “degrade” problem occur, the term of , and this formula will become , it can help to prevent the network from getting hurt by degradation and keep the network’s strong robustness and its stable performance. Because the size of input is equal to the output of the residual block, so the original network add the parameters from a[l] and Z[l+1] in element wise.

**[Exponential Linear Units activation function]**

We use Exponential Linear Units (ELU)(*Djork-Arne Clevert* *et al*., 2016) [7] as our activation layer. The Equation is shown like that:

 [7]

ReLU activation has a problem that its non-zero and its mean activation larger than zero, it might causes bias shift for the next layer and slow down the training process. However ELU allow its negative values close to zero and decrease the bias shift effect. Meanwhile, as layer become deeper and deeper, because ReLU always get zero when it lies on negative axis, more and more units after ReLU activation might “died”, this condition means their value converge to zero and can’t be updated after forward &backward propagation. This problem happened in our network and it seriously influences training performance. While if we use ELU as our function , this problem disappeared. So we set ELU as our activation.

The other part of Graphene Residual network we used is similar to ResNet50, but removed Max-Pooling layer, added Global Max pooling layer and keep the size of the data haven’t changed until the last FC layer. The network and the hyper-parameter of this network is shown in the attachment.

**[Graphene Concatenate Network]**

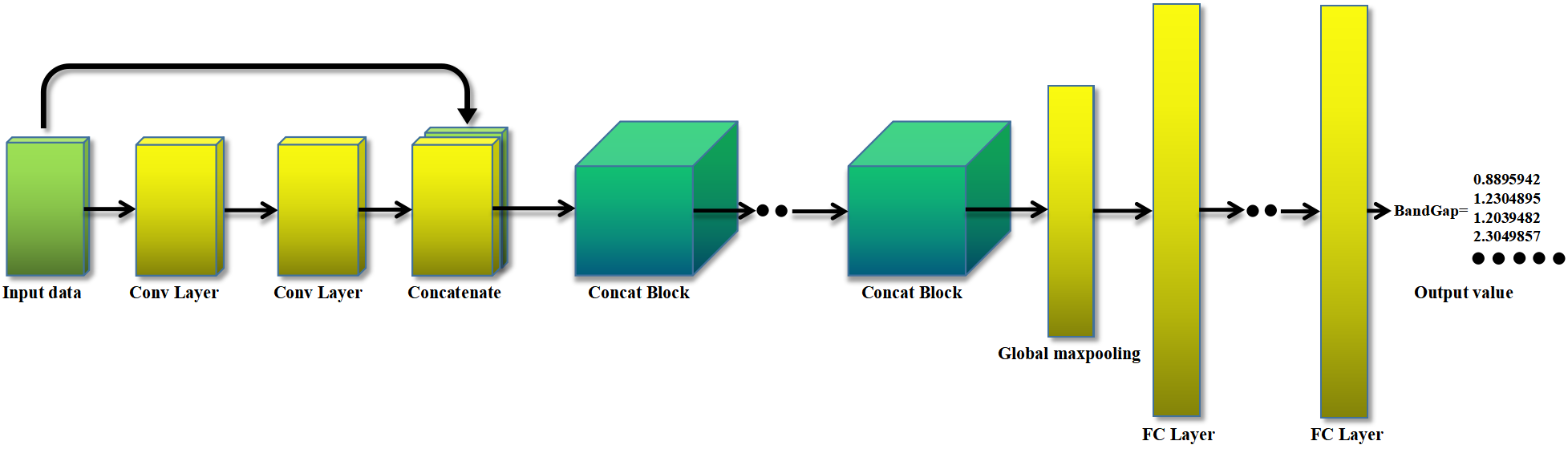
Concatenate Network, while we call it ConcatNet is a special network which we combined some advantages from GoogleNet [4] and DenseNet [5] together. Unlike Residual network which “adds” feature maps in element-wise, this network concatenates the layer from input and output them pass though [l+a] activation layer together. Setting one process as an example. The equation of concatenate operation shown as below:

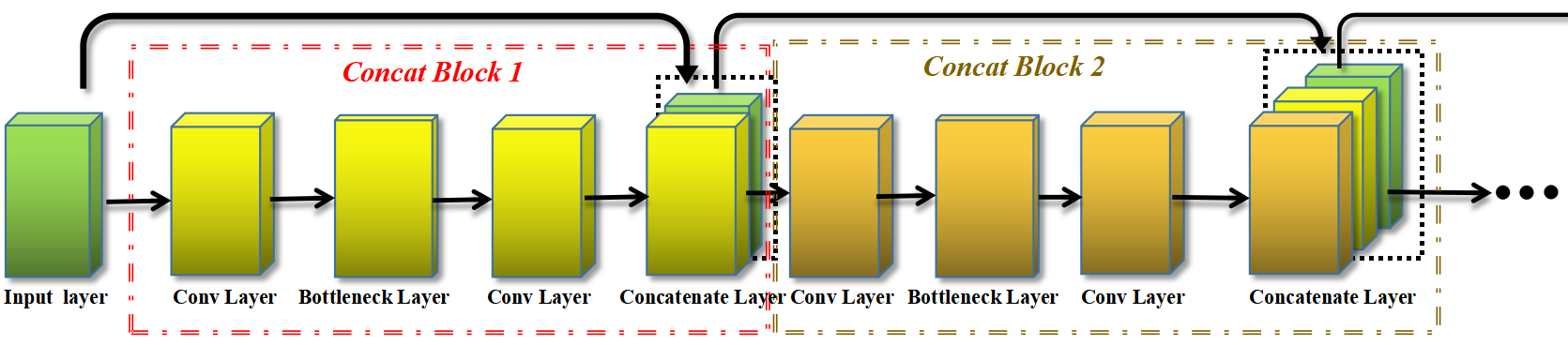


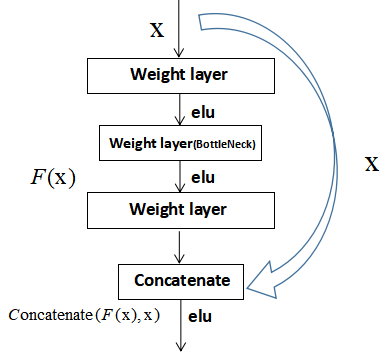
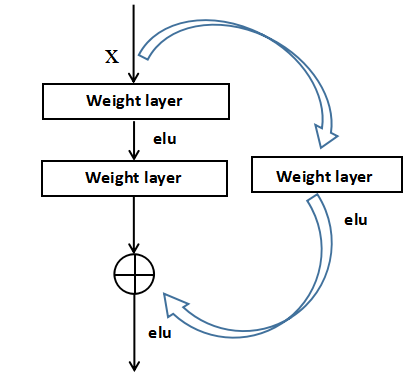
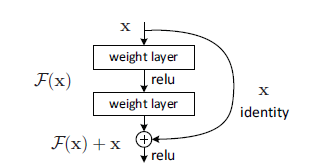
 is the Concatenate function (Concatenate axis= Channels/filter numbers axis)

 is the output of convolution layer

When “degrade” problem occur, the termand the equation becomes . The input’s original information still maintains in the output, since it also can prevent network from degrading. 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 block of residual network, we introduced the small unit of the ConcateNet : Concat Block. This block has three convolution layers, two normal convolution layers and one bottleNet (bottleneck layer) layers inside of them. Bottleneck layer is a convolution layer that it has less filter number compare to the former convolution layer and it’s filter size is 1\*1 and convolution stride is 1. It helps to reduce the number of parameters we need to updated while doesn’t influence any performance of the model. Based on our experiment, ConcateNet also has the problem of “died” neuron unit when we use ReLU as our activation function. So we also use ELU as ConcateNet convolution and fully-connected layers’ activation function.



**Fig.5: the structure of ConcatNet. (a) the whole architecture of the ConcatNet. (b) the detail about the connection between two Concat Block.**



(a) (b) (c)

**Fig.4: the structure of ResNet Identity block/Convolution block and ConcateNet Concat block. (a)Identity block. (b)Convolution block. (c) Concat block.**

1. **Performance**

**3.1 How we train**

Because hyper-parameters selection will effect the performance of all models. SVR is a easy one, and only thing we want to considered is its soft-margin and kernel size. Neural networks have many hyper-parameters such as learning rate, loss function, convolution layers’ number and filters’ size and so on. For ResNet and ConcatNet, meanwhile the number of Convolution/Identity block and Concat-Block also is the think we need to consider. Therefore we first text many kinds of hyper-parameter combos, compare their performance. Selecting the optimum one them we can use it to compare the other kind of method. The training condition, the detail of these hyper-parameters are attached in the implement file. **[未完待续]**

After that, we test 4\*4 and 5\*5 data and the method we use to evaluate their performances which are described before. We evaluate 1000 test data which haven’t be use.The profess is shown as below,

**3.2 Evaluation**

We test our method on Test and Extra test datasets. Once the model has been trained completely, pass though these data to the model and gain their band-gap prediction values. Band-gap value is continuous. Therefore we use approximation error estimation to judge their performances. We use three methods to test our models performance : [1] Correlation matrix [2]Root-mean-square error [3] Approximate error.

**[Correlation coefficient]**

Corr(X,Y) which we call it Pearson product-moment correlation coeffcient is a famous way to measure dependence. The equation is shown like that:

****

When both two data distribution are correlation / anti-correlation, their coefficient values are close to one/minus one, while if they are independent between each other, the coefficient value will close to zero. It’s worse in reverse so it also is an unnecessary and sufficient condition. If two data match each other perfectly, their distribution will show in a “linear curve”, while it they are fully independence, their distribution will show in “cluster cloud” or other shape of curve.

**[Root mean square error]**

The root mean square error is a famous method that it used to measure the difference between two values or two dataset. The equation shown like that:

****  is an estimated parameter and  is an estimator

It observe and summarize all magnitudes of error and represented it in a value. If the value is large that it means two data are independent. Otherwise if the value is close to zero, both dataset distribution are quite similar.

**[Average approximate error]**

Average approximate error can be described as three indexes: [1]Average relative error [2]Average absolute error [3]Average percent error (same as relative error but described in the form of percentage). Dataset has m number of data in total, predict values and real values we notated as. The Equation of these indexes are shown as below:



Provided that the model is trained very well. Its approximate error values should in a very low level. we merge the value of absolute error and relative error together and name this combined error as . The smaller value of its value, the better model we got, it's shown as below. We use Alpha and Beta to control relative and absolute error weight toward the results. Now we set alpha and beta to 0.5. So their influences are equal in the result of 

**3.3 Result [三种Evaluation的方法分别都描述下结果，目前只有一个Evaluation 的结果展现出来，未完待续！]**

**[Model only training the same size of data]**

First we only pop the training data with same size to these models. For 4\*4 data and 5\*5 data, the result showed that neural network methods get better performance than traditional machine learning method: SVR, this result matched our expectation because we have huge number of training data so deep learning methods should get decent result than other machine learning method. Meanwhile it shows that ConcateNet get the best result in all evaluate indexes. ResNet peformances are similar to CNN performance, because the evaluation result has a slightly fluctuate by different test dataset, so I think the performance of ResNet and CNN are equal in this part . The results are Shown in Chart.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4by4 Test Data Evaluation** | **Absolute Error** | **Relative Error** | **Percent Error(%)** | **Total Error** |
| **SVR** | **0.1633** | **0.0698** | **7.00%** | **0.1166** |
| **Traditional CNN** | **0.0677** | **0.0506** | **5.07%** | **0.0592** |
| **Residual network** | **0.0614** | **0.0481** | **4.81%** | **0.0523** |
| **ConcatNet** | **0.0157** | **0.0197** | **1.97%** | **0.0177** |

**(a)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5by5 Test Data Evaluation** | **Absolute Error** | **Relative Error** | **Percent Error(%)** | **Total Error** |
| **SVR** | **0.0712** | **0.0609** | **6.09%** | **0.0660** |
| **Traditional CNN** | **0.0691** | **0.0570** | **5.70%** | **0.0630** |
| **Residual network** | **0.0463** | **0.0329** | **3.29%** | **0.0396** |
| **ConcatNet** | **0.0192** | **0.0208** | **2.08%** | **0.0200** |

**(b)**

**Chart.1 The TOP-1 model performance of these methods after predicted each size’s test datasets. (a)4by4 size test graphene data. (b)5by5 size test graphene data.**

**[Model training variable size of data]**

Second we train all data we have to our models (5by5,4by4 size). After training is completed, we predicted all test data from these datasets and evaluate their performances. Because SVR can’t train variable size of data, so it doesn’t do anything in this part. Because 4by4 and 5by5 Size of Graphene data, their distributions are different though their band-gap values field are quite similar. The performances are slightly inferior than the first one and the different between different methods increase quite a lot. In this part, the residual and concatenate part shows its superiority in training variable size of data. The ConcatNet still shown its excellent performance and do well in all evaluate indexes. Then ResNet sit in the second rules make a huge improve toward the tradition CNN model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable Size Evaluation** | **Absolute Error** | **Relative Error** | **Percent Error(%)** | **Total Error** |
| **SVR** | **None** | **None** | **None** | **None** |
| **Traditional CNN** | **0.1573** | **0.1164** | **11.64%** | **0.1369** |
| **Residual network** | **0.1004** | **0.0753** | **7.53%** | **0.0879** |
| **ConcatNet** | **0.0727** | **0.0561** | **5.61%** | **0.0644** |

**Chart.1 The TOP-1 model performance of these methods test datasets.**

**[未完待续]**

**Conclusion**

**[未完待续]**

**Reference**

[1] *[Cortes, Corinna](https://en.wikipedia.org/wiki/Corinna_Cortes" \o "); Vapnik,Vladimir N*. (1995). "Support-vector networks". *[Machine Learning](https://en.wikipedia.org/wiki/Machine_Learning_(journal)" \o "Machine Learning (journal))*. **20** (3): 273–297.[doi](https://en.wikipedia.org/wiki/Digital_object_identifier" \o "Digital object identifier):[10.1007/BF00994018](https://doi.org/10.1007/BF00994018).

[2] *Drucker, Harris; Burges, Christopher J. C.; Kaufman, Linda; Smola, Alexander J.; and Vapnik, Vladimir N.* (1997); "[Support Vector Regression Machines](http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf)", in *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161, MIT Press

[3] *Kaiming He; Xiangyu Zhang; Shaoqing Ren; Jian Sun*. Deep Residual Learning for Image Recongnition; In: CVPR 2016, url: https://www.cv-foundation.org/openaccess/content\_cvpr\_2016/papers/He\_Deep\_Residual\_Learning\_CVPR\_2016\_paper.pdf

[4]Chiristian Szegedy; Wei Liu ; Yangqing Jia et.al. Going Deeper With Convolutions. In: CVPR 2015, url: <https://arxiv.org/pdf/1409.4842.pdf>

[5] *Gao Huang, Zhuang Liu, Laurens van der maaten et.al*. Densely Connected Convolutional Networks; published in CVPR 2017, url: <https://arxiv.org/pdf/1608.06993.pdf>

[6] *Krizhevsky, A. et al* (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*. pp.1097-2105

[7] *Djork-Arne Clevert, Thomas Unterthiner, Sepp Hochreiter* (2016) Fast and Deep Network Learning by Exponential Linear Units (ELUs). Published in ICLR 2016, url: <https://arxiv.org/pdf/1511.07289.pdf>

[8] LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). ["Gradient-based learning applied to document recognition"](http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf). *Proceedings of the IEEE*. **86** (11): 2278–2324. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier" \o "Digital object identifier):[10.1109/5.726791](https://doi.org/10.1109/5.726791). Retrieved October 7, 2016.

[9] *Seyed Majid; Azimi1; Dominik Britz et.al.* Advanced Steel Microstructural Classification by Deep Learning Methods published in Nature-Scientific Report Journal. Cited as [arXiv:1706.06480](https://arxiv.org/abs/1706.06480) [cs.CV]

[10] *Sergey Ioffe ,Christian Szegedy.* Accelerating Deep Network Training by Reducing Internal Convariate Shift. Published in arXiv. Url: <https://arxiv.org/pdf/1502.03167.pdf.> Cited as [arXiv:1502.03167](https://arxiv.org/abs/1502.03167) [cs.LG]

**[Transfer Learning method of predicting 6by6 data ]**

As we all know, DFT algorithm is an algorithm that it has a huge size of time complexity, so we need a fair amount of time to calculate the Band-gap of each graphene structure and it's hard for us to get enough data which have complicated and large size. Now we introduce the latest technique of machine learning initializes method -- Transfer Learning. When two machine learning problems are quite similar , for the predicting model, the feature and the way it thinks that how to solve these problems are similar too. Therefore inside these models, they have almost the same values and distribution shape among the serial of front or even the whole layers' parameter (weights and bias). Thus when one problem is accomplished and gets the descent result, we can transform some part of its machine learning model's parameters values to the model we use in the other problem. It has the huge improvement compare to randomly initialize parameters' value.For the Graphene Learning problem, we use this method to promote the predicting performance of the large size of data. Set 6by6 graphene data as an example, we only have **(\*\*\*) 6by6数量** data, so the performance is poor if we only use these data. However we have almost **(\*\*\*) 5by5数据数量**data of 5by5 size. So after we finished training the 5by5 data predicting model, we save its parameters (weights and bias) values and transform these values to the model which we use in predicting 6by6 data (the structure of the model is the same one), then we start to learn. The result show that it's worth to do it.