Electronic Supplementary Information

A Transfer Learning Approach for Improved Classification of Carbon Nanomaterials from TEM Images

Qixiang Luo[1], Elizabeth A. Holm[2], Chen Wang[3]*

[1]Department of Material Science and Engineering, Pennsylvania State University, University Park, PA 16802, United States
[2]Department of Material Science and Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States
[3]Health Effects Lab Division, National Institute for Occupational Safety and Health, Centers for Disease Control and Prevention, Cincinnati, OH 45226, United States

*Author to whom correspondence should be addressed (xli7@cdc.gov)

S1. Hyperparameter Optimization

When applying image processing with the deep CNN learning, a large amount of intermediate information extracted from the features require optimization. The optimization process was applied to three main hyperparameters: hypercolumn density and channel, K-means size, and boosting parameters.

We first optimized two hyperparameters associated with the hypercolumn settings: density and channel. The hypercolumn density is defined as how many pixels are taken into account when multiple intermediate images through vgg16 architecture are extracted and summarized to create a standard size hypercolumn. In our model, each image at different levels of Conv block has different shapes (224, 112, 56, 28, and 14 pixels, respectively). A fully-considered or 100% density hypercolumn contains all channels in-depth, and fully-normalized size of pixels in width and length, represented by a total of 50,176 pixels. The density can be set as10% (5,000 times 1,472 channels), 20% (10,000 times 1,472 channels), etc., where 5 Conv block layers are included in the 1,472 channels. The results showed that optimization for the hypercolumn density had little impact on classification performance in the range from 5% to 60%, while the typical settings were selected as10% and 20%.

Hypercolumn channel, another important hyperparameter, is defined as the depth of the hypercolumn. The total number of channels is 1,472 including the depth of convolutional layers 2, 4, 6, 9, and 12 (see Fig. 6 in the main content). We attempted to utilize 5 blocks to count features from shallow to deep. Each extracted image feature was then standardized to a certain shape and counted to obtain the number of channels. Even extraction from 5 Conv blocks was a balanced strategy to combine deep filtered information output from the bottom LinearSoftmax layer of the VGG16 architecture, and shallow features processed through the Conv matrices. Future efforts will be made to consider a different combination of extracted layers to fit image datasets from a wide range of interests.

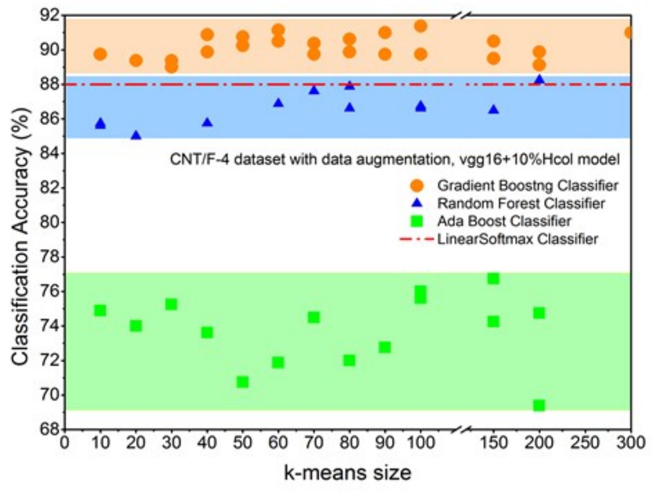

Fig. S1 Comparison of model classification accuracy (%) under different boosting classifiers.

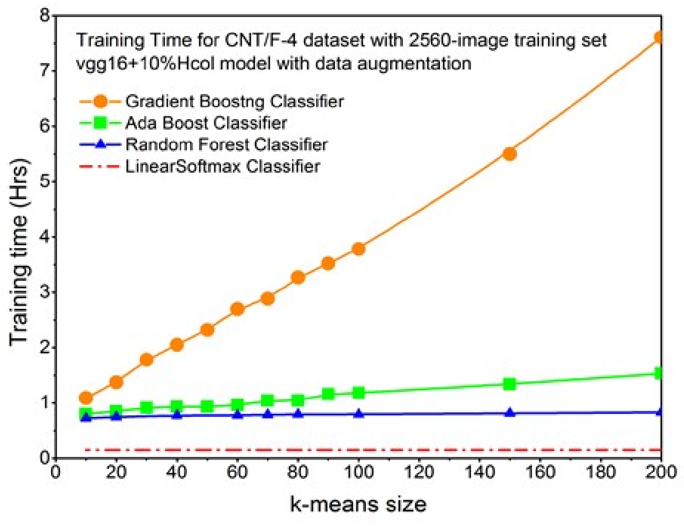

Fig. S2 Comparison of model training time (Hrs) under different boosting classifiers, test workstation configuration: GeForce RTX 2070 8GB + Intel Core i7-8700k.

Different K-means dictionary designs are often used to describe features or vectors in the image. K-means size is referred to the number of intersections of a grid or mesh to describe local image features as a vector of semantics, where then these semantics can be encoded into analyzable residuals. The larger the K size is, the more precise feature residuals can be summarized from the hypercolumn by VLAD encoding. However, the size of K-means is limited by training-testing mode of CNN model, when the overfitting of certain classes may lead to a lack of generalization ability to new images and results in bottleneck or limit value on performance. It was also noticed that storage memory of computational sources can be limited since these trained feature residuals need to be stored on RAM as a semantic dictionary. Therefore, the selection of K-means is mostly based on practical experiments by tuning on hyperparameters to reach global optimization. Three types of boosting algorithms including Ada Boost, Random Forest, and Gradient Boosting were tested. The effects of the K-means size and different boosting classifier conditions were evaluated by the classification performance (see Fig. S1) and computation expense (see Fig. S2). As seen in Fig. S1, Gradient Boosting achieved better accuracy in the range of 89–92% than the performance values of Random Forest and Ada Boost, both of which were below the baseline value (88%) obtained by applying LinearSoftmax classifier. When applying different K-means sizes to the same boosting classifier, it showed little impact on the overall classification accuracy. When it comes to the comparison of the computational cost of using different boosting classifiers, Fig. S2 showed that the training time for the Gradient Boosting classifier increased drastically as the size of the K-means dictionary increased whereas other two classifiers were less dependent upon the K-means size. Considering the overall classification accuracy and computational cost associated with the selected classifiers and K-means sizes, we chose Gradient Boosting to build our classifier under the hypercolumn model in the normal K-means size condition (5-100) to achieve the desired accuracy within reasonable computational time.

S2. Particle classification performance

Table S1. Average 5-fold cross-validation classification accuracy (%) for the 4-class and 8-class datasets. The effects of the models (linear-softmax vs. hypercolumn) and the datasets (original vs. augmented) were evaluated.

Model	**Dataset**	**Processing**	**Cl**	**Fi**	**Ma**	**MS**	**GS**	**SP**	**HDP**	**PR**	**Total (std.)**
Linear-SoftMax	4-class	Original	67.7	69.2	86.5	83.2	–	–	–	–	80.3 (4.79)
Linear-SoftMax	4-class	Augmented	88.4	93.7	84.9	84.9	–	–	–	–	88.0 (2.08)
10% Hcol	4-class	Original	65.2	72.6	91.1	88.2	–	–	–	–	82.9 (6.21)
10% Hcol	4-class	Augmented	90.5	91.5	92.4	89.1	–	–	–	–	90.9 (0.71)
Linear-SoftMax	8-class	Original	64.0	71.5	86.6	74.8	60.0	70.0	52.7	52.6	74.3 (5.80)
Linear-SoftMax	8-class	Augmented	80.4	84.7	77.4	79.1	72.5	89.2	81.7	76.7	80.2 (2.56)
10% Hcol	8-class	Original	67.0	58.0	88.8	87.6	70.0	56.5	51.3	70.7	78.5 (6.91)
10% Hcol	8-class	Augmented	77.9	87.8	87.0	88.3	83.8	90.3	89.7	71.0	84.5 (3.37)

S3. Confusion Matrix Analysis

A confusion matrix was introduced to evaluate the classification performance, with its vertical axis representing the ground truth labels and horizontal axis as the predicted labels. It illustrates the classification accuracy for each individually labeled class on the diagonal direction and false negative or false positive rates on other lattices.

As shown in Fig. 11 in the main content, the classification error for a class (for example Ma) is primarily attributed to a particular class (MS) rather than other evenly distributed misclassified labels. The reason for this biased error is rooted within the sample image dataset, where some labeled images in certain classes pertain highly similar features (See Figure S3), leading to higher classification errors than other classes (e.g., 9% of MS being predicted as Ma and 6% of Ma vice versa as seen in Fig. 11(a)).

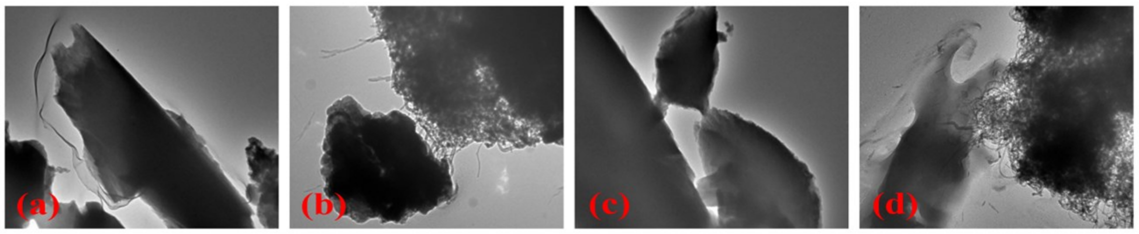

Fig. S3. Comparison of similar sample images from different classes: (a-b) Matrices (Ma), (c-d) Matrix-surfaces (MS).

To improve the misclassification of particular classes with high similarity, two prospective approaches are suggested: (1) optimize the existing classifier, and (2) reconstruct a new classification workflow with multi-level architecture. Instead of applying linear distribution on final output for processed features, one approach is to implement a nonlinear or bias-weighted output layer as a classifier, with enhanced capabilities to identify classes with high similarities. Also, combing multiple classifiers can be applied as an alternative approach to achieve improved accuracy on classification tasks. Applying a multi-level classification workflow is another workable approach to improve the classification specifically for similar classes. Taken 4-class dataset as an instance, instead of using Clusters, Fibers, Matrices, and Matrix-Surfaces as four initial labels for output, similar classes can be grouped into the large classes (such as Fi grouped with Cl, and Ma coupled with MS) followed by the classification of the individual classes themselves, which separates the overall hard classification tasks into multiple easy tasks.