Supporting Information

Effect of Graphene Electrode Functionalization on Machine Learning-Aided Single DNA Nucleotide Classification

Mohd Rashid, † Milan Kumar Jena, † Sneha Mittal, † Biswarup Pathak†,*

†Department of Chemistry, Indian Institute of Technology (IIT) Indore, Indore, Madhya Pradesh, 453552, India

*E-mail: biswarup@iiti.ac.in

Contents **Pages**

1. Machine Learning Details

Text S1:

For machine learning (ML) classification, we have considered four supervised classifiers such as K-nearest neighbors (KNN), support vector machine (SVM), random forest classifier (RFC), and decision tree classifier (DTC), as available in the machine learning package "scikit-learn library" using python version 3.10.[1] All the details about the four models have been discussed below.

(a) K-Nearest Neighbour (KNN)

The k-Nearest Neighbors (KNN) algorithm is a non-parametric and instance-based learning method widely used for classification and regression tasks.[2] It operates on the principle that similar data points are likely to have similar outcomes. For a given instance, KNN identifies the k closest data points (neighbors) in the training dataset using a distance metric, typically Euclidean/Manhattan/Minkowski distance. In classification tasks, the query instance is assigned the majority class label among its k neighbors. For regression tasks, the prediction is made by averaging the values of the k nearest neighbors. Despite its simplicity and effectiveness, KNN can be computationally intensive, especially with large datasets, as it requires exhaustive distance computations and storage of the entire dataset.

(b) Support Vector Machine (SVM)

Support vector machine (SVM) is a supervised learning algorithm used for both classification and regression. It finds the optimal hyperplane to separate data points of different classes, maximizing the margin. SVM uses kernel functions to handle non-linear data. It optimizes a convex objective function with a regularization parameter. Once trained, SVM classifies new data points based on their position relative to the hyperplane. It's effective in various applications like image classification and text categorization.[3]

(c) Decision Tree Classifier (DTC)

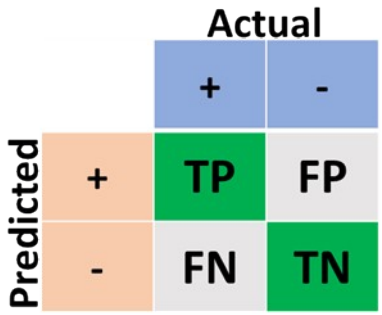
A decision tree classifier is a supervised learning algorithm that uses a tree-based model to classify a set of data points.[4] It works by constructing a tree-like representation of the relationships between different features in the data. Each internal node in the tree represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label. The algorithm starts at the root node and iterates down the tree by evaluating the features at each node and selecting the appropriate branch until a leaf node is reached. The class label assigned to the leaf node is the prediction made by the decision tree classifier for the input data. The goal of the decision tree is to find the splits that result in the highest information gain or reduction in entropy. The algorithm uses a criterion such as the Gini impurity or the information gain to determine the best split at each node. The tree is then grown until it reaches stopping criteria such as maximum depth, minimum samples per leaf, or minimum gain required for a split. Decision trees are simple and easy to interpret but can be prone to overfitting if the tree is allowed to grow too deep. To mitigate this, the tree can be pruned or ensembled with other models. Additionally, decision trees are sensitive to the scale of the features, so it may be necessary to normalize or standardize the data before using it to train the model.

(d) Random Forest Classifier (RFC)

Random Forest Classifier (RFC) is an extension of the decision tree algorithm that operates by constructing a multiple of decision trees at training time and outputting the class that is the mode of the classes.[5] In training, trees are grown using bootstrapped samples of the data and a random subset of the features. This results in a low correlation between the trees, reducing overfitting. During the prediction, the algorithm takes the average prediction across all trees, providing more stability and robustness to outliers. RFC can handle high dimensional and non-

linear data and is considered a robust algorithm for classification. However, it is computationally expensive and may have a longer training time compared to other algorithms.

In ML classification, the confusion matrix is the best evaluation matrix and basic of all other matrices. it is a table with combinations of predicted and actual values.



True positive (TP) = The number of correct positive predictions made by a model.

True negative (TN) = The number of correct negative predictions made by a model.

False positive (FP) = The number of incorrect positive predictions made by a model.

False negative (FN) = The number of incorrect negative predictions made by a model.

Accuracy = Accuracy measures the number of correct predictions done by the model among the total number of predictions.

Table S1. Details of Machine Learning Evaluation Metrics

Accuracy	Accuracy measures the number of correct predictions done by the model among the total number of predictions.	$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$
Precision	Precision explains how many of the correctly predicted instances turned out to be positive. Precision is helpful in cases where False Positive is a greater concern than False Negatives. It is also known as the true positive rate.	$Precision = \frac{TP}{TP + FP}$
Recall	Recall explains how many of the actual positive instances we were able to predict correctly with our model. It is a useful	$Recall = \frac{TP}{TP + FN}$

	metric in cases where a False Negative is of greater concern than a False Positive. It is also known as the sensitivity of the model.	
F1 score	It is the harmonic mean of Precision and Recall metrics.	$F1\ score = \frac{Precision\ \times Recall}{Precision + Recall}$

Validation Curve:

A validation curve is conventionally an essential diagnostic tool that depicts the performance of model accuracy on the vertical axis with a change in some parameters of the model on the horizontal axis. Two curves are present in a validation curve – one for the training set score and one for the cross-validation score. A validation curve is used to evaluate an existing model based on hyper-parameters and is not used to tune a model. This is because, if we tune the model according to the validation score, the model may be biased towards the specific data against which the model is tuned, thereby not being a good estimate of the generalization of the model. The validation curve is used to determine how effective an estimator is on data that it has been trained on, as well as how generalizable it is to new input.

ROC- AUC Curve:

The receiver operating characteristics (ROC) curve visually represents the performance of the ML classifier across various classification thresholds. It shows the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) as threshold values change.

$$TPR(Sensitivity) = \frac{TP}{TP + FN} \quad \text{and} \quad FPR = (1 - Specificity) = \frac{TP}{TP + FN}$$

The AUC-ROC metric quantifies the entire area under the ROC curve, extending from the origin point (0,0) to (1,1). This metric helps in assessing the ML model's ability to distinguish between different classes. An ideal ML classifier achieves an AUC value of 1, while a random classifier scores 0.5. It enables straightforward comparisons among various classification models. Typically, models with higher AUC-ROC values demonstrate better classification capabilities among different classes.

SHAP Analysis:

To understand the algorithm working principle of RFC model classification ability toward all three types of data sets, we have plotted a bar plot of SHAP values for each feature to identify how much impact the features have on the model prediction for an individual class. SHapley Additive exPlanations (SHAP) is a method based on the concepts of cooperative game theory for interpreting the prediction of the machine learning "black box" models.[6] Mathematically, the Shapley value for a feature 'x' in the context of a prediction 'p'is given by the following equation.

$$\phi_x(p) = \sum_{s \subseteq N/x} \frac{|S|!(n - |S| - 1)!}{n!}\{p(S \cup x) - p(S)\}$$

where $\phi_x(p)$ is the Shapley value of the x^{th} feature for the prediction of 'p', and 'S' is the subset of all the features 'N'. $p(S)$ represents the model's prediction when only the features in subset 'S' are considered, and $p(S \cup x)$ is the prediction when feature 'x' is included along with the features in subset 'S'.

2. Optimized Geometries of Four Functionalized Nanogap Devices

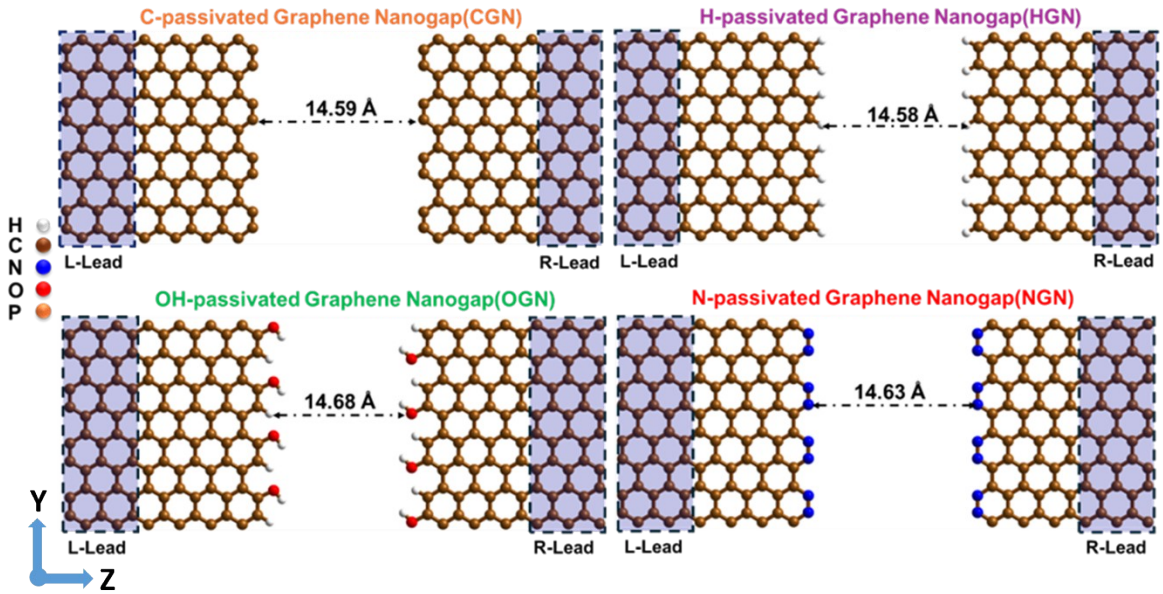


Figure S1. Optimized geometries of proposed CGN, HGN, OGN, and NGN devices. Each nanogap comprises left (L) and right (R) leads and a central scattering region with precise gap sizes customized for individual devices. Here, the z-axis is considered as the transport direction.

3. Top and Side Views of Optimized Nanogap+Nucleotide Geometries across Four Functionalized Nanogap Devices

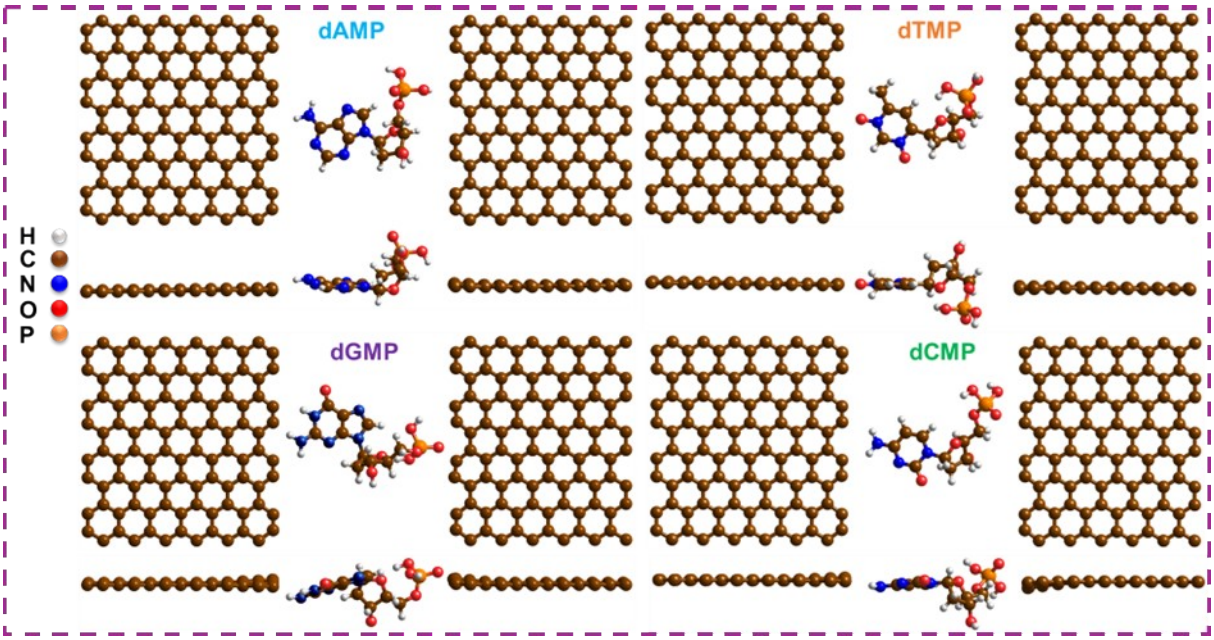


Figure S2. The optimized structures (top and side views) of the CGN+nucleotides (dAMP, dTMP, dGMP, and dCMP) systems are shown.

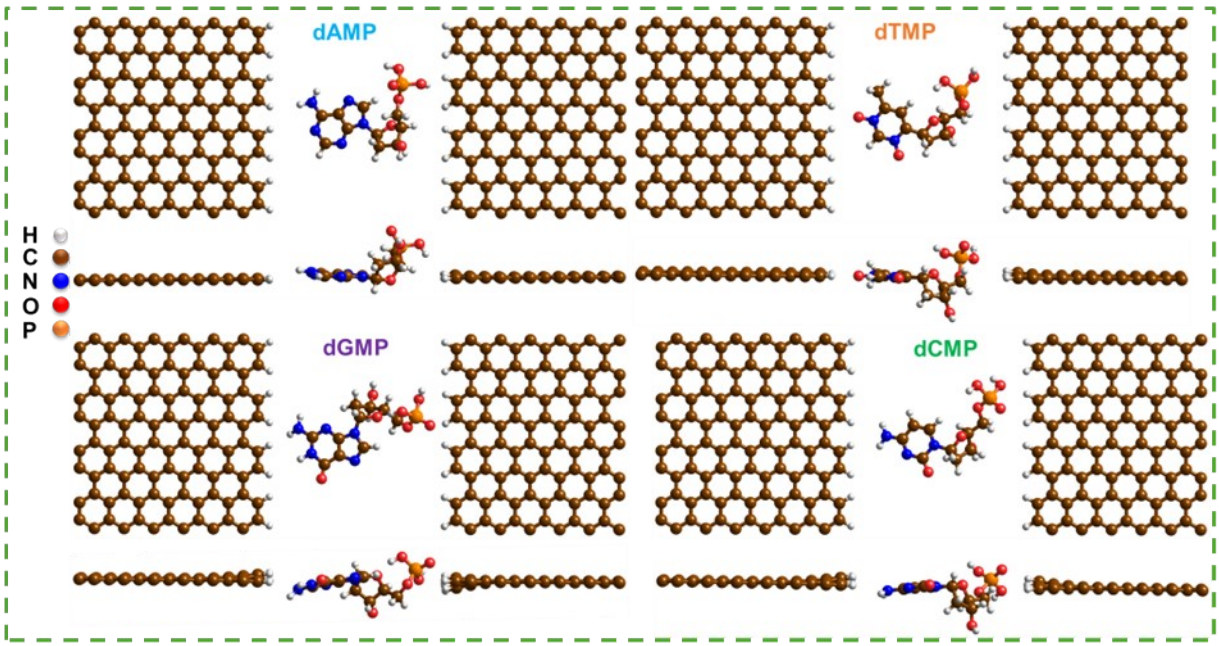


Figure S3. The optimized structures (top and side views) of the HGN+nucleotides (dAMP, dTMP, dGMP, and dCMP) systems are shown.

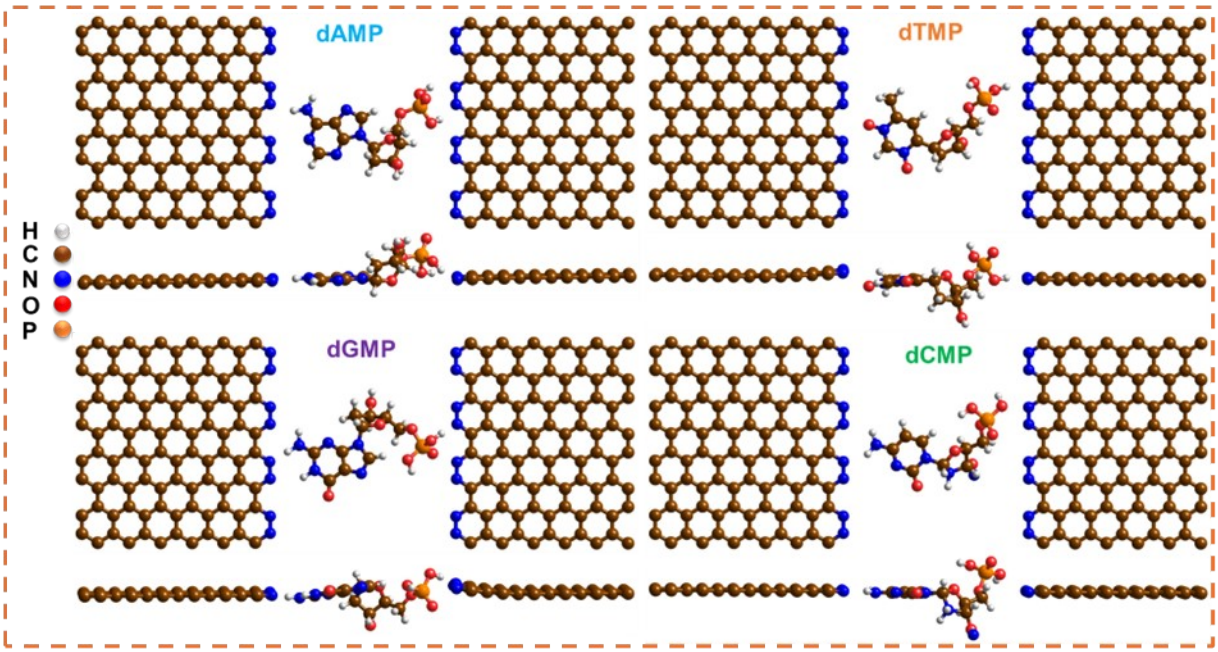


Figure S4. The optimized structures (top and side views) of the NGN+nucleotides (dAMP, dTMP, dGMP, and dCMP) systems are shown.

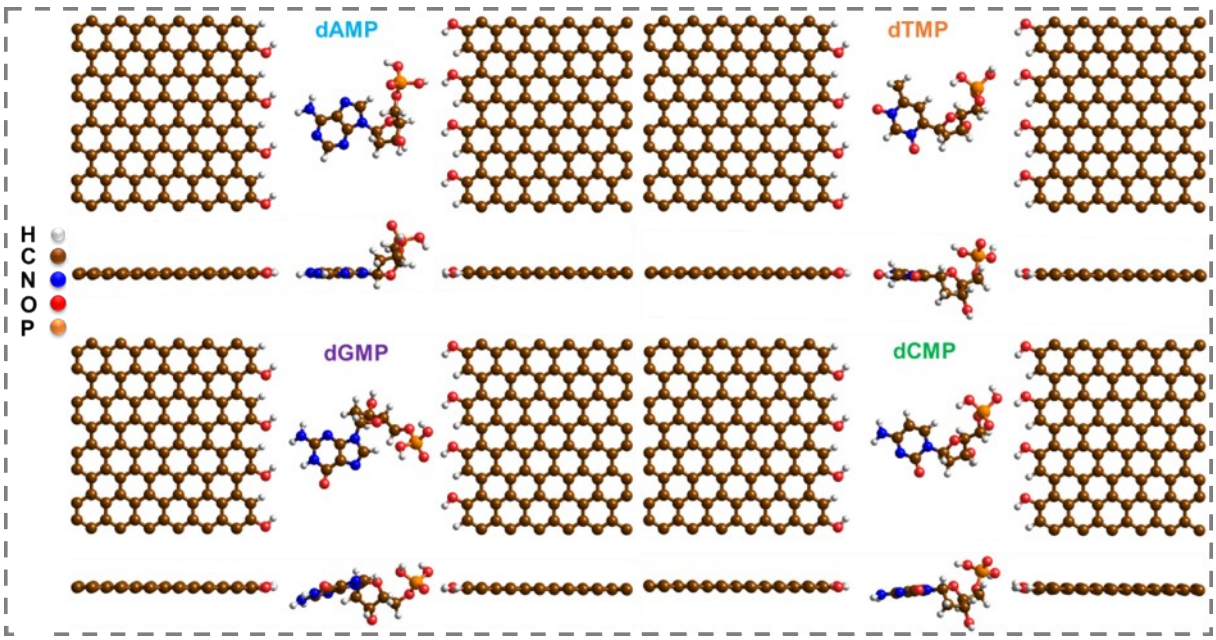


Figure S5. The optimized structures (top and side views) of the OGN+nucleotides (dAMP, dTMP, dGMP, and dCMP) systems are shown.

4. Effect of In-plane Rotation Dynamics of Nucleotides (dAMP, dTMP, dGMP, and dCMP) on their Transmission Function

To counter the effect of the dynamic behavior of translocating nucleotides inside the nanogap device, we have considered seven different orientations (0º to 180º in steps of 30º) along the *x*-axis in the *yz*-plane for each nucleotide, and their transmission functions are calculated as shown below:

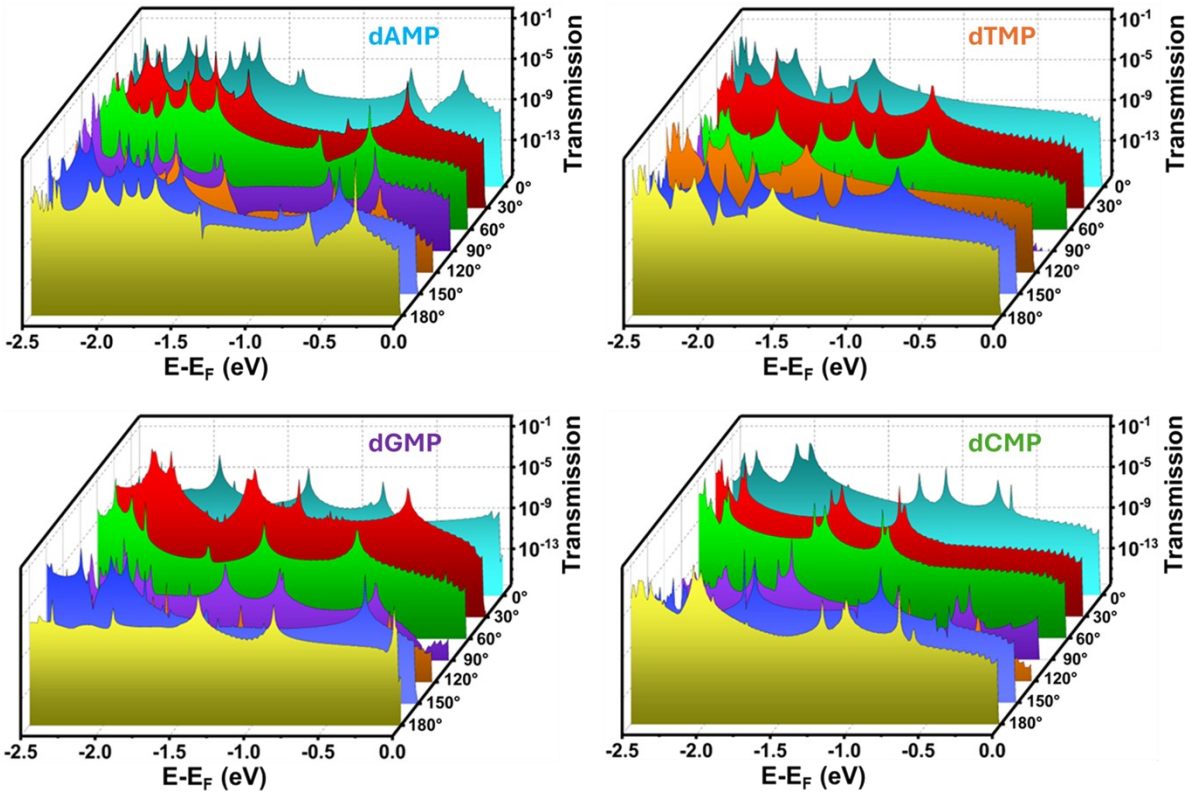


Figure S6. Transmission function plots of all four nucleotides (dAMP, dTMP, dGMP, and dCMP) at seven different orientations inside the CGN device.

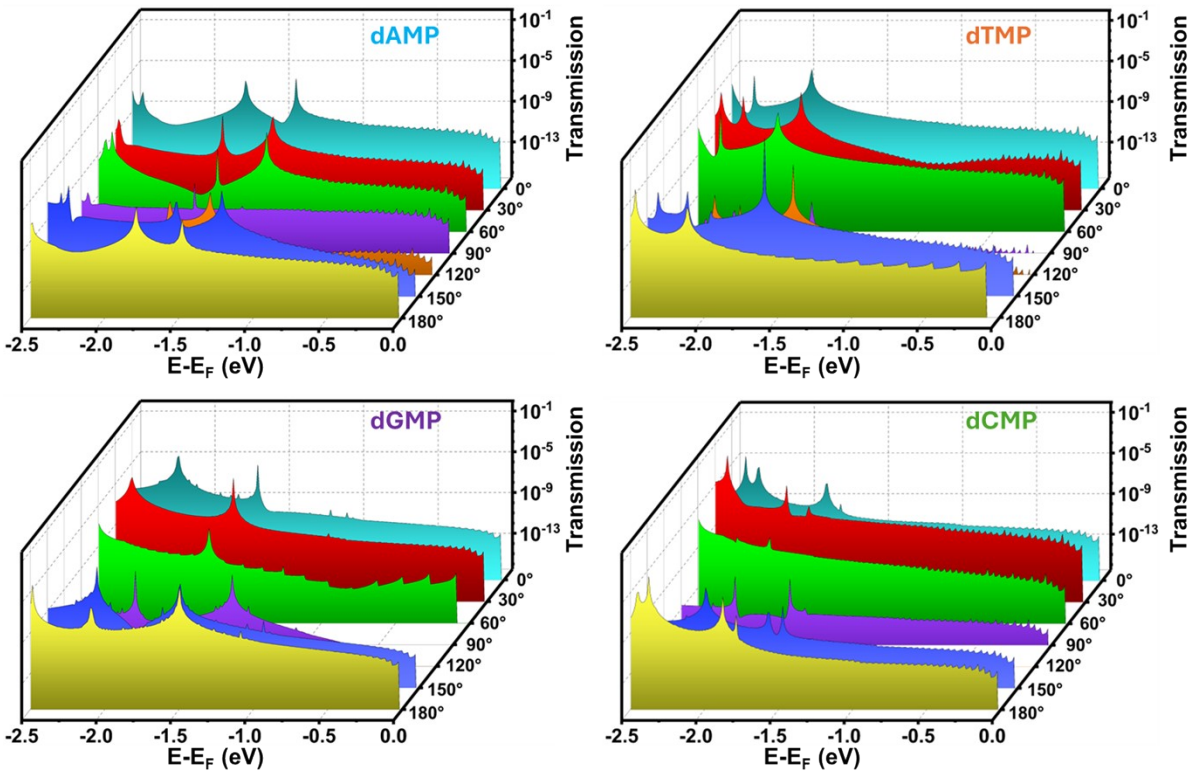


Figure S7. Transmission function plots of all four nucleotides (dAMP, dTMP, dGMP, and dCMP) at seven different orientations inside the HGN device.

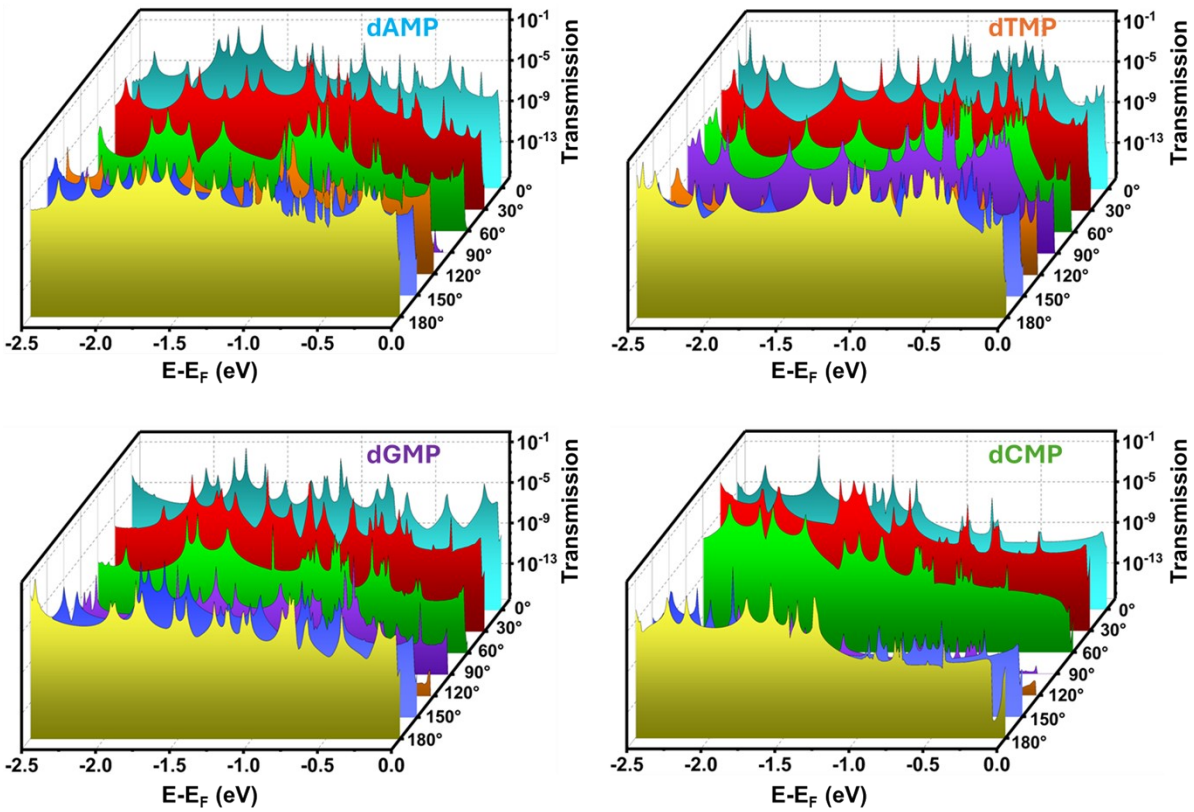


Figure S8. Transmission function plots of all four nucleotides (dAMP, dTMP, dGMP, and dCMP) at different orientations inside the NGN device.

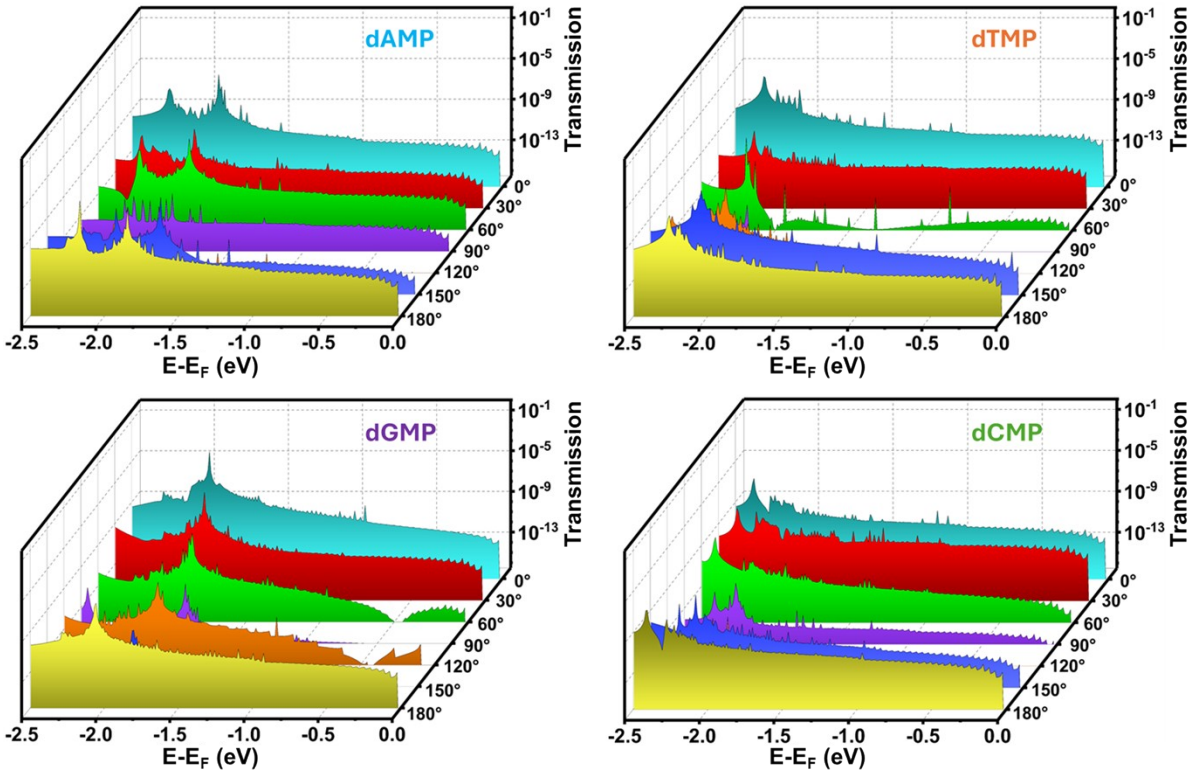


Figure S9. Transmission function plots of all four nucleotides (dAMP, dTMP, dGMP, and dCMP) at different orientations inside the OGN device.

5. List of Optimized Hyperparameters and Test Accuracy Scores (%)

Table S2. Optimized Hyperparameters of the Considered ML Classification Algorithms with their Test Accuracy Score (%)

Sl. No	ML Model	Optimized Hyperparameters	Test Accuracy Score (%)			
			CGN	HGN	NGN	OGN
1	KNN	algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, 'n_neighbors': 30, 'p': 2, 'weights': 'uniform'	48	33	46	33
2	SVM	'C': 1.0, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': False	22	21	21	20
3	DTC	'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 22, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': 35, 'splitter': 'best'	96	90	95	91
4	RFC	'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'entropy', 'max_depth': 25, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': 100, 'verbose': 0, 'warm_start': False	98	91	94	96

6. K-Fold Cross-validation Results

Table S3. 10-Fold Cross-validation Scores (Fold 1-10) of the RFC Model with Four Functionalized Nanogaps Datasets along with Mean Accuracy, Standard Deviation (Std. Dev.), and Test Accuracy Scores for Comprehensive Assessment.

Number of Fold	**Accuracy (%)**			
	CGN	HGN	NGN	OGN
Fold 1	95	86	97	91
Fold 2	98	93	100	87
Fold 3	95	97	96	97
Fold 4	100	96	96	97
Fold 5	100	95	96	99
Fold 6	93	95	91	97
Fold 7	95	87	92	92
Fold 8	93	95	96	84
Fold 9	93	82	96	87
Fold 10	97	95	93	97
Mean Accuracy ± std. Dev.	95.9 ± 2.73	92.1 ± 5.15	95.3 ± 2.63	93.8 ± 4.37
Test Accuracy	98	91	94	96

7. Interaction Energy (E_i) and Translocation Time (τ)

To understand the electronic interaction between the graphene electrode and the nucleotides located inside, we have calculated the interaction energy (E_i) using the following equation S1.[7]

$$E_i = E_{gn+nu} - (E_{gn} + E_{nu}) \quad \text{(S1)}$$

where E_{gn+nu} is the total energy of the graphene nanogap + nucleotide, E_{gn} and E_{nu} are the single-point energy values of isolated graphene nanogap and isolated targeted nucleotide, respectively.

Table S4. The Interaction Energy (E_i) Values and Translocation Times (τ) of all Four DNA Nucleotides while Located inside Four Functionalized Nanogap (CGN, HGN, NGN, and OGN) Devices

CGN

Nucleotides	E_i (eV)	τ
dAMP	-0.75	1.88×10^{-13}
dTMP	-1.04	2.11×10^{-18}
dGMP	-1.70	1.39×10^{-29}
dCMP	-0.87	1.71×10^{-15}

HGN

Nucleotides	E_i (eV)	τ
dAMP	-0.29	1.13×10^{-05}
dTMP	-0.64	1.38×10^{-11}
dGMP	-0.71	7.54×10^{-13}
dCMP	-0.61	3.84×10^{-11}

NGN

Nucleotides	E_i (eV)	τ
dAMP	-1.99	1.19×10^{-34}
dTMP	-1.28	2.18×10^{-22}
dGMP	-2.36	9.04×10^{-41}
dCMP	-1.77	8.45×10^{-31}

OGN

Nucleotides	E_i (eV)	τ
dAMP	-0.18	7.64×10^{-04}
dTMP	-0.46	1.85×10^{-08}
dGMP	-0.91	3.01×10^{-16}
dCMP	-0.44	3.63×10^{-08}

8. Density of State (DOS) Plots

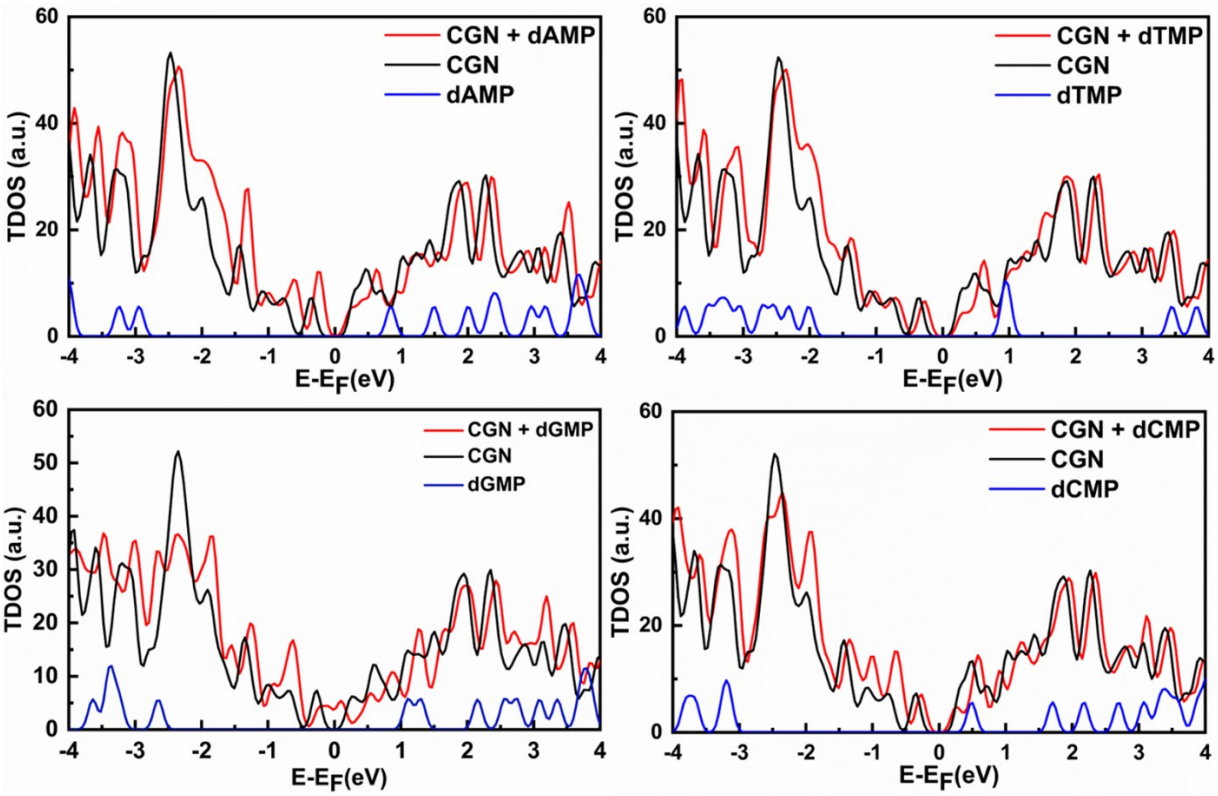


Figure S10. The electronic density of state (DOS) plots for CGN+nucleotide, bare CGN device, and isolated nucleotide molecule are represented in red, black, and blue colors, respectively.

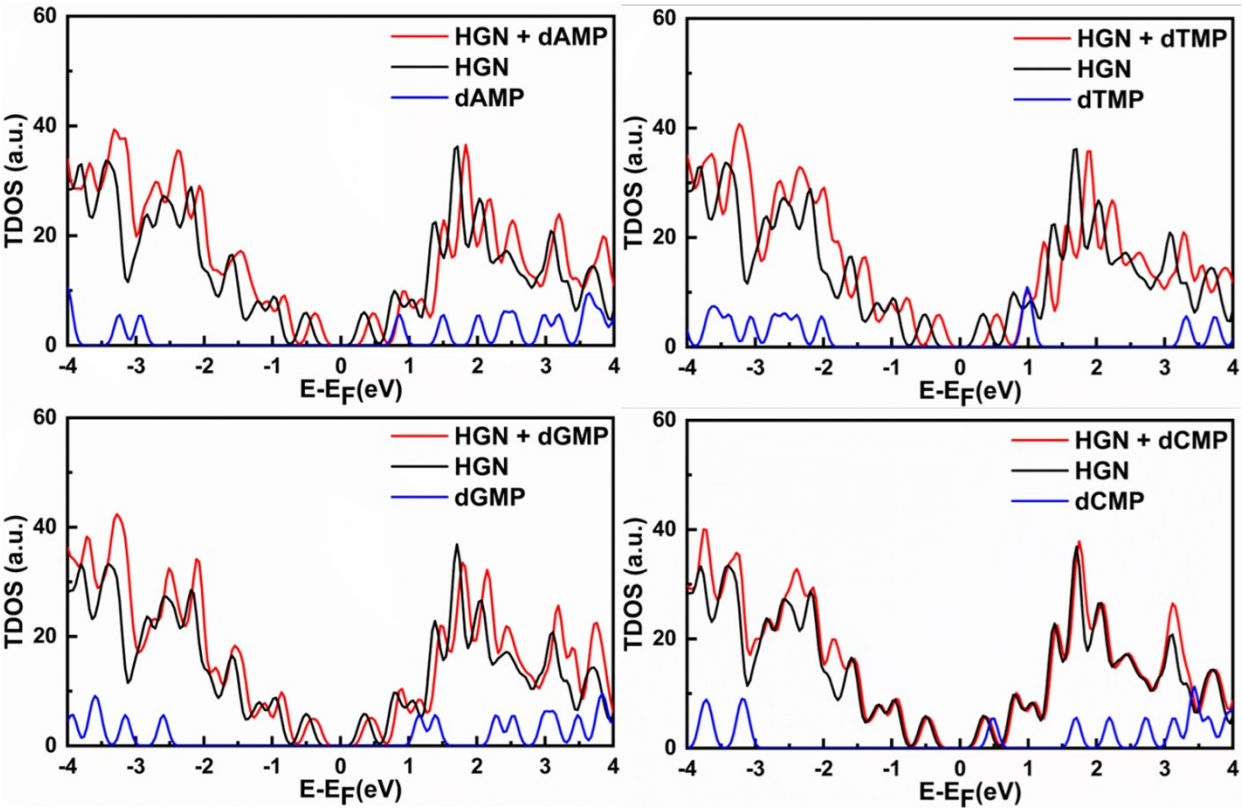


Figure S11. The electronic density of state (DOS) plots for HGN+nucleotide, bare HGN device, and isolated nucleotide molecule are represented in red, black, and blue colors respectively.

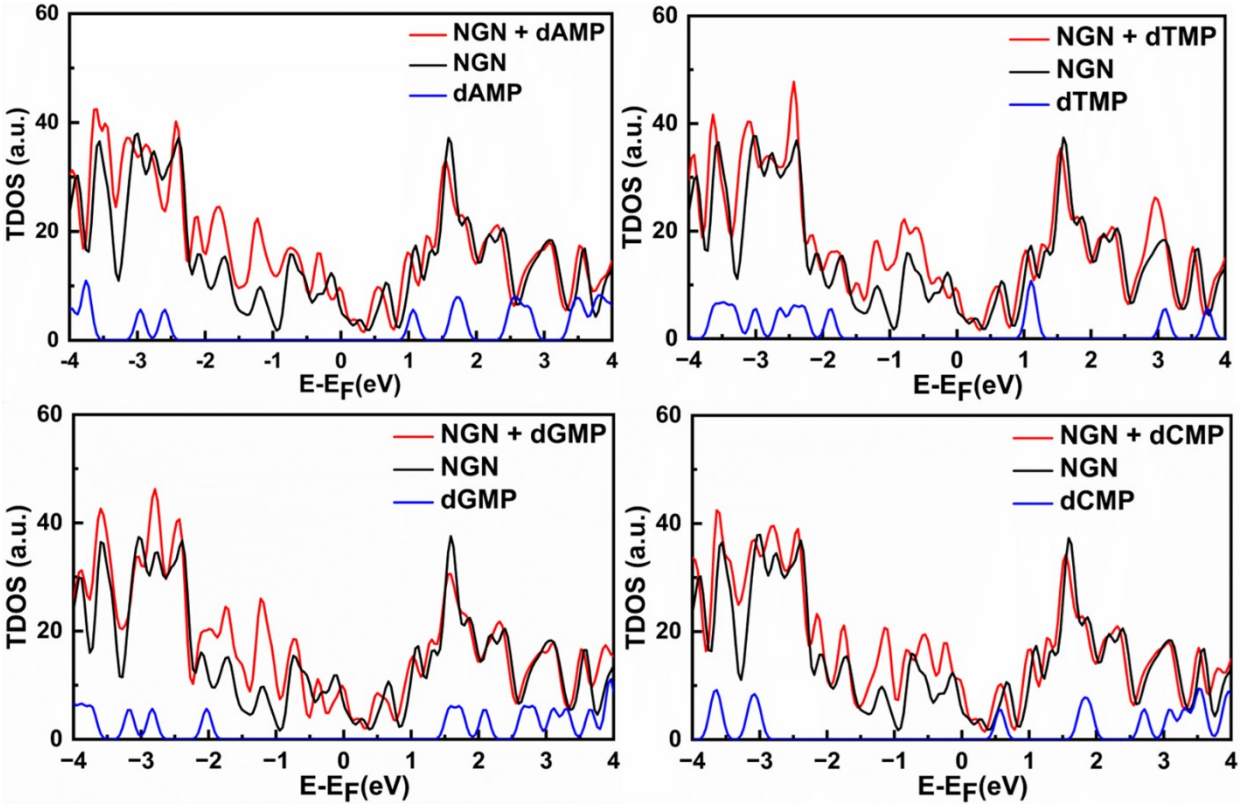


Figure S12. The electronic density of state (DOS) plots for NGN+nucleotide, bare NGN device, and isolated nucleotide molecule are represented in red, black, and blue colors respectively.

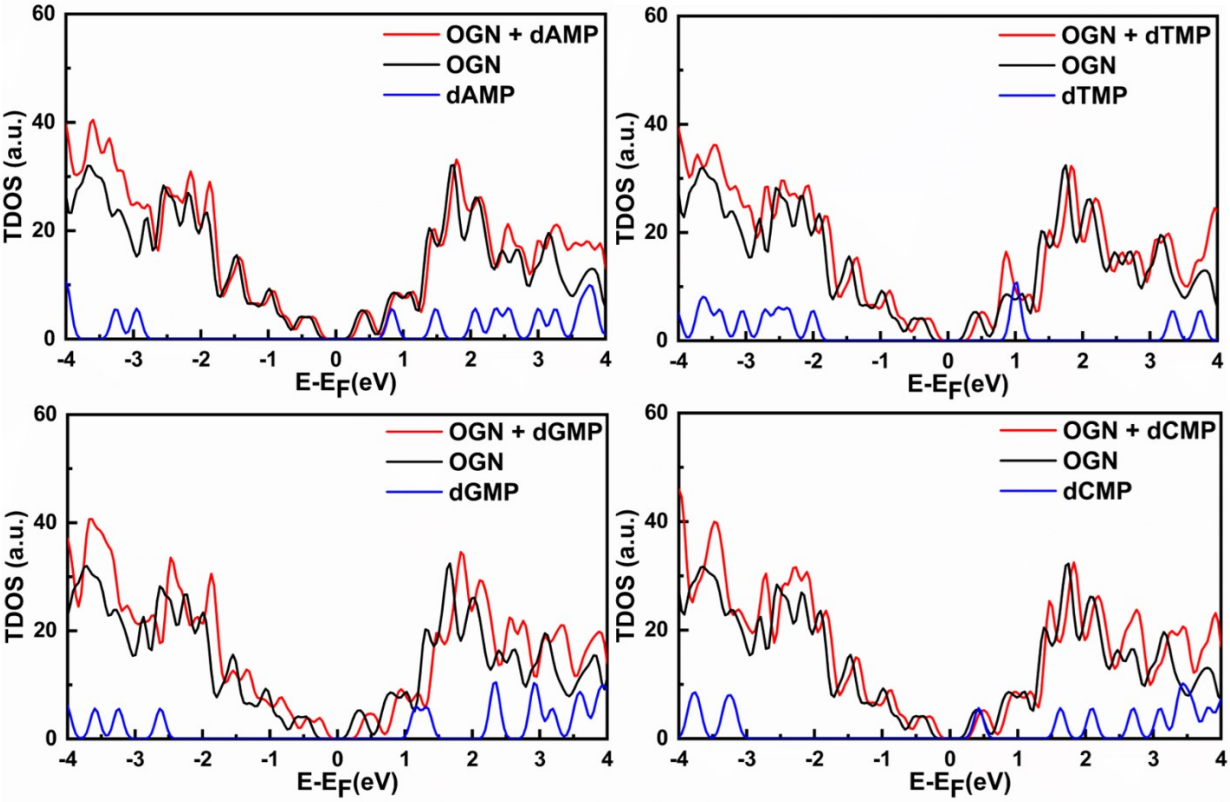


Figure S13. The electronic density of state (DOS) plots for OGN+nucleotide, bare OGN device, and isolated nucleotide molecule are represented in red, black, and blue colors, respectively.

References:

1. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos and D. Cournapeau, *Mach. Learn. PYTHON*.
2. F. Nigsch, A. Bender, B. van Buuren, J. Tissen, E. Nigsch, J.B.O. Mitchell, J. Chem. Inf. Model. 2006, 46 (6), 2412– 2422.
3. C. Cortes and V. Vapnik, *Mach. Learn.*, 1995, **20**, 273–297.
4. Breiman, L. Random Forests. Machine Learning 2001, 45, 5– 32.
5. Y. Freund and R. E. Schapire, *J. Comput. Syst. Sci.*, 1997, **55**, 119–139.
6. S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal and S.-I. Lee, *Nat. Mach. Intell.*, 2020, **2**, 56–67.
7. S. Mittal, S. Manna, M. Jena and B. Pathak, *ACS Mater. Lett.*, 2023, **5**, 1570–1580.