

Supporting Information

Inverse Design of Hybrid Organic-Inorganic Perovskites with Suitable Bandgaps via Proactive Searching Progress

Tian Lu^a, Hongyu Li^a, Minjie Li^{b,c*}, Shenghao Wang^{a,*}, Wencong Lu^{a,b,c*}

^a Materials Genome Institute, Shanghai University, Shanghai 200444, China

^b Department of Chemistry, College of Sciences, Shanghai University, Shanghai 200444, China

^c Zhejiang Laboratory, Hangzhou 311100, China

*Corresponding Authors.

E-mail addresses: minjieli@shu.edu.cn (M.J. Li); shenghaowang@shu.edu.cn (S.H. Wang); wclu@shu.edu.cn (W.C. Lu)

Contents

Section S1 Codes for Training Process.....	3
Code S1 Dealing with multiple target values of singular sample.....	3
Code S2 Generating Descriptors.....	5
Code S3 Searching Optimal Sample Distribution	17
Code S4 Feature Selection	19
Code S5 SHAP Analysis.....	21
Code S6 Hyper-parameter Optimization	23
Code S7 Weighted Voting Regressor	25
Code S8 SHAP Analysis for Weighted Voting Regressor	27
Code S9 Proactive Searching Progress (PSP).....	42
Section S2 Other Figures and Tables.....	44

Section S1 Codes for Training Process

This section provides the detailed instructions to use our inhouse package to realize the steps and functions in the training progress. Before reproduce the steps, make sure to install the package “fast-machine-learning” by running “pip install fast-machine-learning” in cmd. The PyPI website is: <https://pypi.org/manage/project/fast-machine-learning/release/0.0.2.0/>. The training process is based on the version of 0.0.2.0, and the API functions might be changed in later developments.

The code in each step will be illustrated in general, and the code files are supplemented as the supporting files at <https://github.com/luktian/InverseDesignViaPSP>. The data set used in this manuscript and the relevant output files are also attached at the GitHub.

Code S1 Dealing with multiple target values of singular sample

As shown in Figure S1, the raw sample set (loaded from “code1/extracted_samples_1159.xlsx” in supporting files) was processed using the module “fml.targets.DealMultipleTarget”, in which the formulas were identified using the module “fml.formulars.SplitFormular”. The sample number reduced to 437 from 1159, in which the details could be seen in the variable “dealed_data” or the output excel file “./code1/pruned_samples_437.xlsx”.

From the variable “counters”, more information could be accessible to the readers. Typically, the bandgap of the identical HOIP material MAPbI₃ had 340 diverse values ranging from 1.45 ~ 2.0 eV, which may be caused by the experimental error or the diverse conditions. As shown in Table S1, there are 176 samples owned with bandgap 1.50 eV while the second top is 54 samples with 1.51 eV. After being processed by using the dealing method “prone_mean” in the module “fml.targets.DealMultipleTarget”, the bandgap of MAPbI₃ was adapted to 1.504 eV. Other identical HOIPs also had the multiple bandgap values. For example, MAPbBr₃ had 51 diverse bandgap values with the adapted value of 2.298 eV, while FAPbI₃ had 33 diverse values and the adapted value of 1.519 eV. For more details, it might be possible for readers to inspect the variables in code or the provided supporting data files in directory “./code1”.

```

from fml.data import read_data
from fml.formulars import SplitFormular
from fml.targets import DealMultipleTarget
import numpy as np

# Load the 1159 raw samples pre-extracted from 9509 publications
data = read_data("extracted_samples_1159.xlsx") # data is a dataframe
raw_formulars = data.PL
data.index = raw_formulars
raw_bg = data.PLBandGap

# Some unregular formular ratios are also considered
unregular_formular_ratios = [
    [1.05, 1, 3],
    [1, 1.1, 3.3],
    [0.97, 1, 2.77],
    [0.966, 1, 2.90],
    [0.9575, 1, 2.9575],
    [1, 1.05, 3],
    [1.02, 1, 3],
    [0.95, 1, 3],
    [1.05, 1, 3.05],
    [0.99, 1, 3],
]

# Use the module fml.formulars.SplitFormular to split the formualrs into string format
splitted_formular, unsplitted_formular = SplitFormular(unregular_formular_ratios).\
    split_formulars(raw_formulars.values, output=str)

# Use the module fml.targets.DealMultipleTarget to deal with
# the multiple target values of one singular sample
#
# The deal method is selected as "prone_mean" that considers
# the standard distribution of the targets. The deal alternatives
# could also be "mean", "max", "min"
#
# the first column of dealed_data is the formulars
# the second column of dealed_data is the processed band gaps
# the counters contain the statistical information of the multiple target values
dealed_data, counters = DealMultipleTarget(deal="prone_mean").deal_with_multiple_data(
    np.concatenate([np.array(splitted_formular).reshape(-1, 1), raw_bg.values.reshape(-1, 1)], axis=1)
)

# The dealed_data could be outputed in an excel by using dataframe
import pandas as pd
pd.DataFrame(dealed_data, columns=["PL", "Band Gap"]).to_excel("pruned_samples_437.xlsx", index=None)

```

Figure S1 The code to prune the multiple target values of singular sample

Table S1 The diverse bandgap values for the identical HOIPs

MAPbI ₃		MAPbBr ₃		FAPbI ₃	
Target values (eV)	Counts	Target values (eV)	Counts	Target values (eV)	Counts
1.5	176	2.3	29	1.48	5
1.51	54	2.17	3	1.47	3
1.55	24	2.25	3	1.51	3
1.53	18	2.02	2	1.53	3
1.58	14	2.18	2	1.55	3
1.57	12	2.2	2	1.41	2
1.56	9	2.42	2	1.43	2
1.6	8	1.72	1	1.5	2
1.52	6	2.16	1	1.52	2
1.49	4	2.24	1	1.45	1
1.59	4	2.27	1	1.46	1
1.54	2	2.28	1	1.49	1
2.0	2	2.35	1	1.52125	1
1.565	1	2.38	1	1.522054236	1
1.45	1	2.43	1	1.528	1
1.47	1			1.56	1

1.7	1			2.41	1
1.504	1				
1.566	1				
1.633	1				
Processed Value (eV)		Processed Value (eV)		Processed Value (eV)	
1.504		2.298		1.519	

Code S2 Generating Descriptors

As shown in Figure S2, the 437 pruned HOIP formulas along with their bandgap values were loaded from the file “code1/pruned_samples_437.xlsx” resulted from **Code S1**. The “fml.formulars.SplitFormular” module was used again to transform the string formulas into the dictionary format that would be passed into the class function “describe_formular” of “HOIP” instance, which could generate the pre-defined descriptors. The descriptors were divided into three main parts, namely 34 base descriptors, 8 string descriptors, and 66 other descriptors. The descriptors were listed in Table S2, which were marked with the original sources. The atomic descriptors were collected from Villars Database¹ and Mendeleev Python Package², while the part of organic properties was calculated by using Gaussian³ and Multiwfn⁴ software. The details of base and string descriptors were illustrated in Table S3, and more information could be assessed in the documents of Villars Database¹ and Mendeleev Python Package². We collected 80 organic cations that are suitable for A site in ABX₃ HOIPs, whose structures and origins could be seen in Table S4.

By switching the Boolean values in the module “HOIP”, three parts of descriptors could be generated optionally. The options of “base”, “ionization”, “ionic_radii” refer to the base descriptors, while the options “str_bool” and “other” refer to string and other descriptors. The option “onehot” could be used to control whether the string descriptors are encoded into numeric data or not.

At last, the generated data set was represented in a table format with the row referring to sample indexes and the columns referring to bandgap and descriptors. The data file was outputted as the “./code2/HOIP_dataset_437.xlsx” in the supporting files.

```

from fml.data import read_data
from fml.formulars import SplitFormular
from fml.descriptors import HOIP
import pandas as pd

# Load the 437 pruned samples in dataframe format
# Index refers to the formular strings, and the first column is the pruned band gaps
bandgap_data = read_data("../code1/pruned_samples_437.xlsx")
formulars = bandgap_data.index.values
bandgaps = bandgap_data.iloc[:, 0].values

# Some unregular formular ratios are also considered
unregular_formular_ratios = [
    [1.05, 1, 3],
    [1, 1.1, 3.3],
    [0.97, 1, 2.77],
    [0.966, 1, 2.90],
    [0.9575, 1, 2.9575],
    [1, 1.05, 3],
    [1.02, 1, 3],
    [0.95, 1, 3],
    [1.05, 1, 3.05],
    [0.99, 1, 3],
]

# Split the pruned formular strings into dict format that will be passed into
# the class module HOIP to generate descriptors
splitted_formulars, _ = SplitFormular(unregular_formular_ratios).split_formulars(formulars)

# The class HOIP is used to generate descriptors
# Firstly instantiate an HOIP object with the bool options as followed
# The base refers to use the base descriptors
# The str_bool refers to use the string descriptors
# The other refers to 66 other descriptors
# Ionization and ionic radii are calculated separately
hoip = HOIP(base=True, str_bool=True, other=False, ionization=True, ionic_radii=True)

# Use the class function "describe_formular" of hoip object to calculate descriptors
# Pass a singular formular, which is formatted as:
# formular = [{atom:ratio, atom:ratio, ...},
#             {atom:ratio, atom:ratio, ...},
#             {atom:ratio, atom:ratio, ...}]
# The first dict refers to A site, the second and third refer to B and C sites
# Switch the onehot to True. The string descriptors will be generated in numeric data
descriptors = []
for formular, bg in zip(splitted_formulars, bandgaps):
    descriptors.append(hoip.describe_formular(formular, onehot=True))
# Collecting descriptors into variable "descriptors" and concatenate into a dataframe
descriptors = pd.concat(descriptors, axis=1).T

# Concatenate the band gap and descriptors into one sheet
dataset = pd.concat([bandgap_data, descriptors], axis=1)
dataset.to_excel("HOIP_dataset_437.xlsx")

```

Figure S2 The code to generate descriptors

Table S2 The details of the assembled descriptors from Villars Databases, Mendeleev Package, and the quantum-based results.

Descriptor Types	Descriptor names	Properties in Villars Database	Properties in Mendeleev Package	Properties for Organic fragments
base descriptors	atomic_number	Atomic number start counting left top, left-right sequence	atomic_number	
	quantum_number	quantum number	period	
	atomic_weight	atomic weight	atomic_weight	molecular_weight
	group_number	group number		

valence_electron_number	valence electron number		
melting_point	temperature melting	melting_point	
boiling_point	temperature boiling	boiling_point	Boiling temperature
enthalpy_vaporization	enthalpy vaporization		Evaporation enthalpy
enthalpy_melting	enthalpy melting		
enthalpy_atomization	enthalpy atomization		
volume_Villars	volume atom		
volume_Mendeleev		atomic_volume	atomic volume
en_martynov	electronegativity (Martynov&Batsanov)		
en_pauling	electronegativity (Pauling)	en_pauling	
en_allred	electronegativity (Allred-Rochow)		
en_mulliken	electronegativity absolute		Mulliken electronegativity
en_allen		en_allen	
en_ghosh		en_ghosh	
first_ionization	energy ionization first		
second_ionization	energy ionization second		
third_ionization	energy ionization third		
chemical_potential	chemical potential Miedema		chemical potential
radii_pseudo_zunger	radii pseudo-potential (Zunger)		
radii_ionic_yagoda	radii ionic (Yagoda)		
radii_metal_waber	radii metal (Waber)	metallic_radius	
atomic_radii		atomic_radii	atomic radii

	density		density	density
	dipole_polarizability		dipole_polarizability	molecular polarity index
	electron_affinity		electron_affinity	vertical electron affinity
	evaporation_heat		evaporation_heat	evaporation enthalpy
	lattice_constant		lattice_constant	
	proton_affinity		proton_affinity	
	ionization		ionization	ionization
	_ionic_radii_		ionic_radii	ionic_radii
string descriptors	block		block	
	lattice_structure		lattice_structure	
	cas		cas	
	goldschmidt_class		goldschmidt_class	
	geochemical_class		geochemical_class	
	is_monoisotopic		is_monoisotopic	
	is_radioactive		is_radioactive	
	name		name	
other descriptors	atomic_number_2	Periodic number start counting left bottom, left-right sequence		
	atomic_number_3	Periodic number start counting top right, right-left sequence		
	atomic_number_4	Periodic number start counting bottom right, right-left sequence		
	enthalpy_surface_miedema	enthalpy surface Miedema		
	enthalpy_vacancies_miedema	enthalpy vacancies Miedema		
	mass_attenuation_coef_mokalpha	mass attenuation coefficient for MoKalpha		
	mass_attenuation_coef_cr	mass attenuation		

kalpha	coefficient CrKalpha		
mass_attenuation_coef_cr kalpha_2	mass attenuation coefficient for CuKalpha		
mass_attenuation_coef_fe kalpha	mass attenuation coefficient FeKalpha		
atomic_electron_scatterin g_factor	atomic electron scattering factor at 0.5 (/)		
work_function	work function		
nws1_3_miedema	nWS1/3 Miedema		
nuclear_charge_effective_ slater	nuclear charge effective Slater		
charge_nuclear_effective_ clementi	charge nuclear effective (Clementi)		
energy_cohesive_brewer	energy cohesive Brewer		
modulus_compression	modulus compression		
modulus_bulk	modulus bulk		
modulus_rigidity	modulus rigidity		
modulus_Young	modulus Young		
mendeleviev_number	Mendeleviev Number	mendeleviev_number	
mendeleviev_number_2	Mendeleviev Number		
mendeleviev_number_3	Mendeleviev Number		
mendeleviev_number_4	Mendeleviev Number		
pettifor_number	Mendeleviev Pettifor		
pettifor_number_regular	Mendeleviev Pettifor regular		
glawe_number		glawe_number	
mendeleviev_chemisits_seq uence	Mendeleviev chemists sequence		
mendeleviev_t_d_left	Mendeleviev t-d start left		

mendelev_t_d_right	Mendelev t-d start right		
mendelev_d_t_left	Mendelev d-t start left		
mendelev_d_t_right	Mendelev d-t start right		
mendelev_h_li_na_be_mg_block_t_d_left	Mendelev H,Li,Na,Be,Mg as block t-d start left		
mendelev_h_be_mg_t_d_left	Mendelev H,Be,Mg t-d start left		
mendelev_h_li_na_be_mg_t_d_left	Mendelev H,Li,Na,Be,Mg t- d start left		
distance_valence_electron	distance valence electron (Schubert)		
distance_core_electron	distance core electron (Schubert)		
v2_3_miedema	V2/3 Miedema		
atomic_env_number	atomic environment number (Villars, Daams)		
covalent_radius_cordero		covalent_radius_cordero	
covalent_radius_pyykko		covalent_radius_pyykko	
covalent_radius_bragg		covalent_radius_bragg	
covalent_radius_slater		covalent_radius_slater	
covalent_radius_pyykko_double		covalent_radius_pyykko_double	
covalent_radius_pyykko_triple		covalent_radius_pyykko_triple	
vdw_radius		vdw_radius	
vdw_radius_bondi		vdw_radius_bondi	
vdw_radius_truhlar		vdw_radius_truhlar	
vdw_radius_rt		vdw_radius_rt	
vdw_radius_batsanov		vdw_radius_batsano	




		v	
vdw_radius_dreiding		vdw_radius_dreiding	
vdw_radius_uff		vdw_radius_uff	
vdw_radius_mm3		vdw_radius_mm3	
vdw_radius_alvarez		vdw_radius_alvarez	
atomic_radius_rahm		atomic_radius_rahm	
atomic_weight_uncertainty		atomic_weight_uncertainty	
gas_basicity		gas_basicity	
heat_of_formation		heat_of_formation	
c6		c6	
c6_gb		c6_gb	
metallic_radius_c12		metallic_radius_c12	
dipole_polarizability_unc		dipole_polarizability_unc	
specific_heat		specific_heat	
fusion_heat		fusion_heat	
thermal_conductivity		thermal_conductivity	
abundance_crust		abundance_crust	
abundance_sea		abundance_sea	

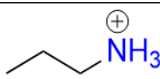
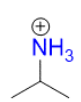
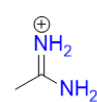



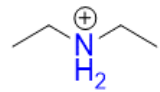

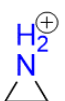


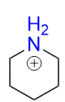
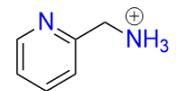
Table S3 The details of base and string descriptors

Descriptor names	Descriptions	Unit	Examples
atomic_number	Number in periodic table	/	Cl (17)
quantum_number	Quantum Number	/	Cl (3)
atomic_weight	Atomic weight	/	Cl (35.45)
group_number	Group number	/	Cl (7)
valence_electron_number	Valence electron number	/	Cl (7)
melting_point	Melting point	K	Cl (172 K)
boiling_point	Boiling point	K	Cl (238.6 K)
enthalpy_vaporization	Vaporization enthalpy	kJ/mol	Cl (20.42 kJ/mol)
enthalpy_melting	Melting enthalpy	kJ/mol	Cl (6.41 kJ/mol)
enthalpy_atomization	Atomization enthalpy	kJ/mol	Cl (121.7 kJ/mol)
volume_Villars	Volume from Villars database	10 ⁻² nm ³	Cl (2.886 10 ⁻² nm ³)
volume_Mendeleev	Volume from Mendeleev package	cm ³ /mol	Cl (18.7 cm ³ /mol)
en_martynov	Electronegativity (Martynov&Batsanov)	/	Cl (2.98)
en_pauling	Electronegativity (Pauling)	/	Cl (3)
en_allred (EA)	Electronegativity (Allred-Rochow)	/	Cl (2.83)
en_mulliken	Electronegativity absolute	//	Cl (8.3)
en_allen (EN)	Electronegativity	/	Cl (16.97)

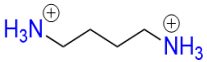
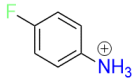

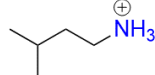
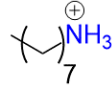
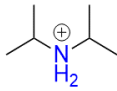
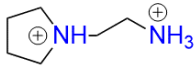

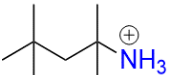
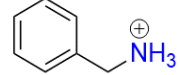
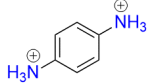
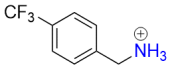
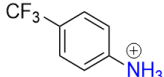
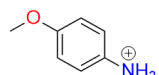
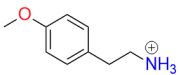
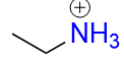
en_ghosh	Electronegativity	/	Cl (0.264)
first_ionization	First ionization energy	kJ/mol	Cl (1251.1 kJ/mol)
second_ionization	Second ionization energy	kJ/mol	Cl (2297 kJ/mol)
third_ionization (TI)	Third ionization energy	kJ/mol	Cl (3822 kJ/mol)
chemical_potential	Chemical potential	/	Cl (0)
radii_pseudo_zunger (RPZ)	Radii pseudo-potential (Zunger)	a.u.	Cl (0.92 a.u.)
radii_ionic_yagoda	Radii ionic (Yagoda)	Å	Cl (0.2 Å)
radii_metal_waber	Radii metal (Waber)	Å	Cl (30 Å)
atomic_radii	Radii from Mendeleev package	pm	H (79 pm)
density	Density	g/cm ³	Cl (1.4 g/cm ³)
dipole_polarizability (DP)	Dipole polarizability	a.u.	Cl (11.083 a.u.)
electron_affinity	Electron affinity	eV	Cl (-11.5 eV)
evaporation_heat	Heat evaporation	kJ/mol	Cl (6.52 kJ/mol)
lattice_constant (LS)	Lattice constant of element crystal	/	Cl (5.26)
proton_affinity	Proton affinity	kJ/mol	Cl (369.2 kJ/mol)
ionization	Ionization depending on degree	eV	Cl ⁻ (12.97 eV) Pb ²⁺ (15.03 eV)
_ionic_radii_	Ionic radii depending on charge	Å	Cl ⁻ (1.81 Å) Pb ²⁺ (1.19 Å)
block	Block position in periodic table	/	Cl (p)
lattice_structure	Lattice structure of element crystal	/	Cl (ORC) ORC refers to Primitive orthorhombic
cas	Chemical Abstracts Serice identifier	/	Cl (7782-50-5)
goldschmidt_class	Goldschmidt classification	/	Cl (lithophile)
geochemical_class	Geochemical classification	/	Cl (semi-volatile)
is_monoisotopic (IM)	Is the element monoisotopic	/	I (True)
is_radioactive	Is the element radioactive	/	Cl (True)
name (N)	Element name	/	Cl (Chlorine)

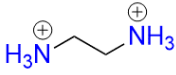
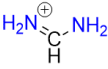
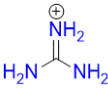
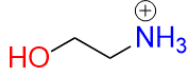

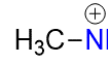
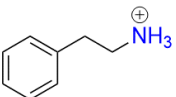
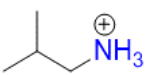

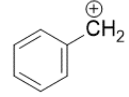






Table S4 Collected structures of organic cations that are suitable for the A site in ABX₃ HOIPs.

Cation Codes	Cation Names	Structures	Reference
A	ammonium		5
AA	dimethylammonium		6
AB	N,N-dimethylmethylammonium		7

AC	propan-1-aminium		6
AD	propan-2-aminium		6
AE	hydroxylammonium	$\text{HO}-\text{NH}_3^+$	7
AF	1-aminoethan-1-iminium		6
AG	hydrazinium	$\text{H}_2\text{N}-\text{NH}_3^+$	7
AH	azetidin-1-ium		7
AI	thiazol-3-ium		7
AJ	1H-pyrrol-1-ium		7
AK	diethylammonium		6
AL	benzenaminium		6
AM	aziridin-1-ium		7
AN	pyrrolidin-1-ium		6
AO	azepan-1-ium		7
AP	piperidin-1-ium		6
AQ	pyridin-2-ylmethanaminium		8
AR	trifluoromethanaminium	$\text{CF}_3-\text{NH}_3^+$	9

AS	difluoromethanaminium		9
AT	fluoromethanaminium		9
AU	cyclohexylmethanaminium		6
AV	hexan-1-aminium		6
AW	dodecan-1-aminium		6
AX	2-methylpropan-2-aminium		6
AY	N1,N1-diethylpropane-1,3-diaminium		6
AZ	N1,N1-dimethylpropane-1,3-diaminium		6
BA	butan-1-aminium		6
BB	morpholin-4-ium		6
BC	cyclohexanaminium		6
BD	2-(4-fluorophenyl)ethan-1-aminium		6
BE	(4-fluorophenyl)methanaminium		6
BF	propane-1,3-diaminium		6
BG	2,2-dimethylpropan-1-aminium		6
BH	N1,N1-dimethylethane-1,2-diaminium		6

BI	butane-1,4-diaminium		6
BJ	4-fluorobenzenaminium		6
BK	pyridin-1-ium		6
BL	3-methylbutan-1-aminium		6
BM	hexan-1-aminium		6
BN	diisopropylammonium		6
BO	1-(2-ammonioethyl)pyrrolidin-1-ium		6
BP	piperazine-1,4-dium		6
BQ	2,4,4-trimethylpentan-2-aminium		6
BR	phenylmethanaminium		6
BS	benzene-1,4-diaminium		6
BT	(4-(trifluoromethyl)phenyl)methanaminium		6
BU	4-(trifluoromethyl)benzenaminium		6
BV	4-methoxybenzenaminium		6
BW	2-(4-methoxyphenyl)ethan-1-aminium		6
EA	ethanaminium		6

ED	ethane-1,2-diaminium		6
FA	aminomethaniminium		6
GA	diaminomethaniminium		6
HE	2-hydroxyethan-1-aminium		10
IA	1H-imidazol-3-ium		6
MA	methanaminium		6
PE	2-phenylethan-1-aminium		6
PY	2-methylpropan-1-aminium		6
SA	tetrafluoroborate		
XA	phenylmethylium		7
XB	oxiran-1-ium		11
XC	oxetan-1-ium		11
XD	tetrahydro-1H-furan-1-ium		11
XE	hexahydropyrylium		11
XF	oxepan-1-ium		11
XG	thiiran-1-ium		12

XH	thietan-1-ium		12
XI	tetrahydro-1H-thiophen-1-ium		12
XJ	hexahydrothiopyrylium		12
XK	thiepan-1-ium		12
XL	phosphiran-1-ium		12
XM	phosphetan-1-ium		12
XN	phospholan-1-ium		12
XO	phosphinan-1-ium		12
XP	phosphepan-1-ium		12
XQ	sulfonium		13
XR	methylsulfonium		13
XS	ethyloxonium		13
XT	methylphosphonium		13

Code S3 Searching Optimal Sample Distribution

As shown in Figure S3, the data set (“./code2/HOIP_dataset_437.xlsx”) was preprocessed by using the module “fml.preprocessing.Preprocessing” under the criteria of trimming the variables that have missing values or standard values nearly to 0, bringing about 102 features left.

As for data set division, the most regular way is to divide the data set randomly into training and test sets, which however may cause the extreme unreasonable sample distribution for training set and result in an unilateral chemical space for fitting models. In practice, if a pair of training and test sets are generated by random splitting based on the packages such as scikit-learn¹⁴ or NumPy¹⁵, the sample distribution is controlled by pseudo-random number generators, *e.g.*, Mersenne Twister in Python standard module “random” and NumPy module “numpy.random.MT19937” and PCG-64 in NumPy module “numpy.random.PCG64”, in which a fixed integer parameter named as random state or seed could always produce the same random sample sequence. By controlling the values of random state and test set size, the distributions of training and test sets could be optimized numerically, in which the sample distribution optimization is converted into a two-parameter optimization task. In this regard, HyperOpt¹⁶ is employed as the backend to perform the optimization task for searching the optimal random state and test size efficiently, in which the package is built based on Bayes theory. In our inhouse code, such process could be realized by using the module “fml.sampling.HpSplit”. The parameter “rounds” and “cv” could control the searching iterations and the fitness type, in which “cv=True” means that the root mean squared error (RMSE) of leaving-one-out cross-validation (LOOCV) is used as the fitness value. The optimal random state and test size could be accessed in the variable “best_params”. It should be noted that the searching procedure via HyperOpt is not a global searching, which will lead to different optimizing results.

By passing the optimal random state and test size, we could easily obtain the divided training and test sets. We also prepared our data files that were generated under the random state of 1959 and test size of 18.05% which were named as “./code3/train_1805_1959.joblib” and “./code3/test_1805_1959.joblib” in the supporting files.

```

from fml.data import read_data
from fml.preprocessing import Preprocessing
from fml.sampling import HpSplit, random_split

# Load the generated data set composed of 437 samples × 129 features
dataset = read_data("../code2//H0IP_dataset_437.xlsx", df=False)
print(f"Feature numbers: {dataset.X.shape[1]}")
# Prune the variables that have missing values or low standard deviations
# nearly to zero, bringing about 102 remnant features left
preprocessed_dataset = Preprocessing(sd=True, corr=False).fit_transform(dataset)
print(f"Feature numbers: {preprocessed_dataset.X.shape[1]}")

# Use module HpSplit to optimize random state and test size in pursuit a suitable
# sample distribution of training and test sets
# "rounds" is set as 200 to perform 200 iterations
# verbose is set as True to print the log information
# Pass dataset variable as the first argument of "fit" function
# cv is set as True which means leaving one out cross validation (LOOCV)
# cv could also be set as an int number which refers to number-fold CV
# The root mean squared error (RMSE) of LOO is used as the optimizing fitness
hs = HpSplit(rounds=200, verbose=True).fit(dataset, cv=True)
# the parameters are saved in "best_params", and summary stores the log information
best_params, summary = hs.best_params, hs.summary
print(f"Random state: {best_params['random_state']}")
print(f"test size: {best_params['test_size']}")

# Use the best parameters from the HpSplit result to split the data
train, test = random_split(dataset, best_params['test_size'],
                           best_params['random_state'])

# Output the train and test
import joblib
joblib.dump(train, "train.joblib")
joblib.dump(test, "test.joblib")

# If you want see the data, use "to_df()" to transform to dataframe
train = train.to_df()
test = test.to_df()

```

Figure S3 The code to split data set into training and test sets

Code S4 Feature Selection

As shown in Figure S4, the training and test sets were loaded from the output files in **Code S3**. The recursive feature addition (RFA) method was performed by importing the inhouse module "fml.pipelines.RFA", which takes training and test sets as main parameters. The optional parameters "min_f" and "max_f" control the minimum and maximum feature numbers in RFA procedure. The RFA approach will fit the models by recursively iterating both of the selected features and the employed 7 algorithms, in which selected features is ranked according to the feature importance extracted from SHAP package¹⁷ and maximum relevance minimum redundancy (mRMR) method¹⁸, and the algorithms cover CatBoost (CAT)¹⁹, XGBoost (XGB)²⁰, LightGBM (LGB)²¹, gradient boosting machine (GBM)²², support vector machine (SVM)²³, decision tree regressor (DTR)²⁴, and multiple linear regressor (MLR)¹⁴. The processed result could be accessed in the variable "summary" in the pandas.DataFrame format, which is outputted as an excel file "./code4/RFA_result.xlsx".

The various model performance could be seen in Figure S5 ~ Figure S8. The model performance was mainly determined based on the LOOCV RMSEs of the 7 models. By comparing the integral LOOCV RMSEs trend between in the Figure S5 (a) and (b), the over performance of the tree-based models was largely greater than the other models. The LOOCV RMSEs of tree-based models were fixed in 0.08 ~ 0.12 after 13 features while the range of the latter were almost over 0.14. The same result could be drawn from the LOOCV determination coefficient (R^2) trend in Figure S6, in which the LOOCV R^2 values of the tree-based models were almost over 0.90. Figure

S7 (a) and Figure S8 (b) signaled the powerful predictivities of the tree-based models with the test RMSEs ranging 0.10 ~ 0.14 and the test R^2 values higher 0.85, indicating that the models were fitted well and avoided the over-fitting problems. Finally, the CAT, XGB, LGB, and GBT models were built up based on the top 13, 13, 12, and 10 features respectively, whose LOOCV R^2 were 0.94, 0.92, 0.90, and 0.93 respectively.

Figure S5 ~ Figure S8 could be generated by the file “./code4/code4_plot.py”.

```

from fml.pipelines import RFA
import joblib

# Load the training and test sets from the step Code3
train = joblib.load("../code3//train_1805_1959.joblib")
test = joblib.load("../code3//test_1805_1959.joblib")

# Use our inhouse module RFA to perform feature selection
# Pass the training and test sets
# min_f and max_f refer to minimum and maximum feature numbers
rfa = RFA().fit_all(train, test, min_f=3, max_f=21)
# The results could be accessed in a dataframe "summary"
summary = rfa.summary_all
# Output the dataframe to an excel file if needed
summary.to_excel("RFA_result.xlsx", index=None)

```

Figure S4 The code to perform RFA feature selection combining with 7 algorithms.

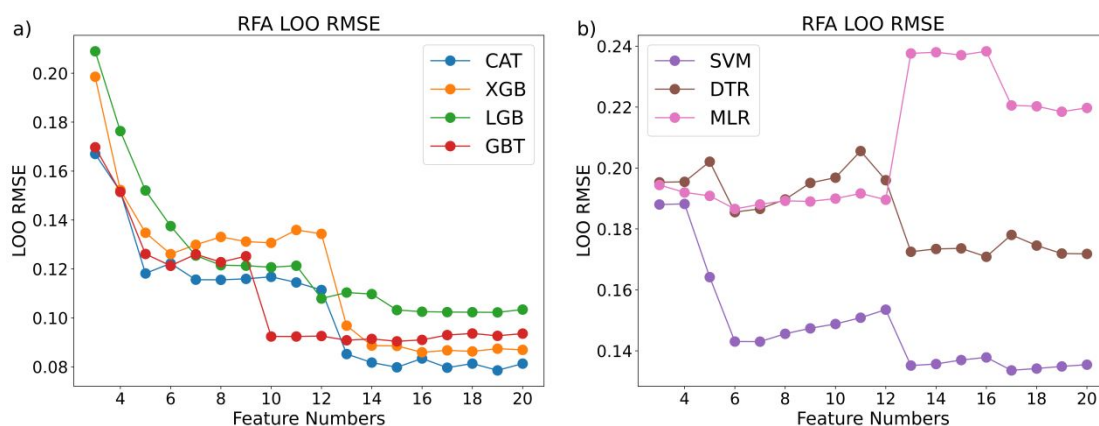


Figure S5 The changing trend of LOO RMSEs of the tree-based (a) and other (b) models along with the increasing number.

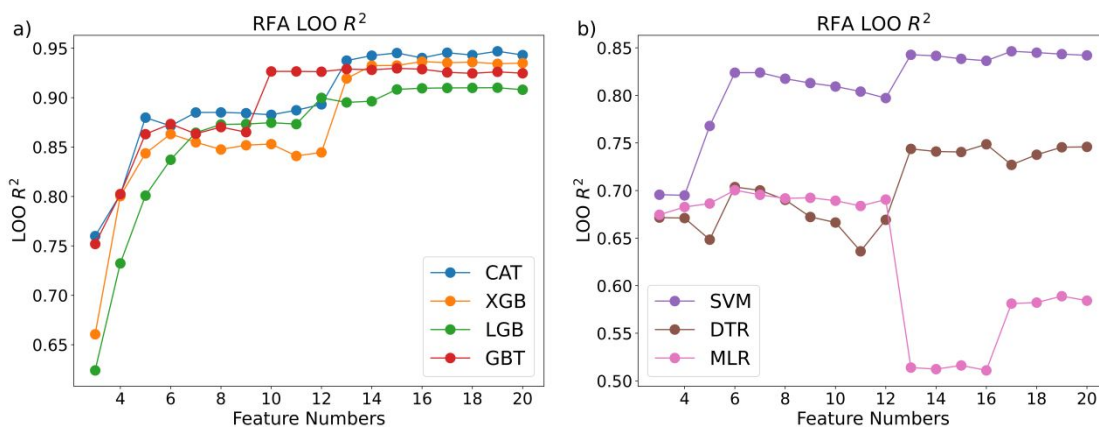


Figure S6 The changing trend of LOO R^2 of the tree-based (a) and other (b) models along with the increasing number.

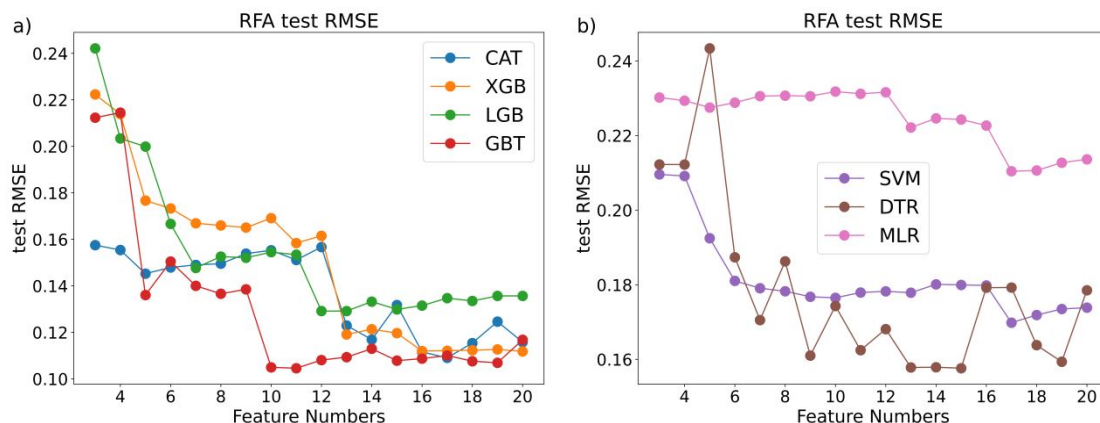


Figure S7 The changing trend of test set RMSEs of the tree-based (a) and other (b) models along with the increasing number.

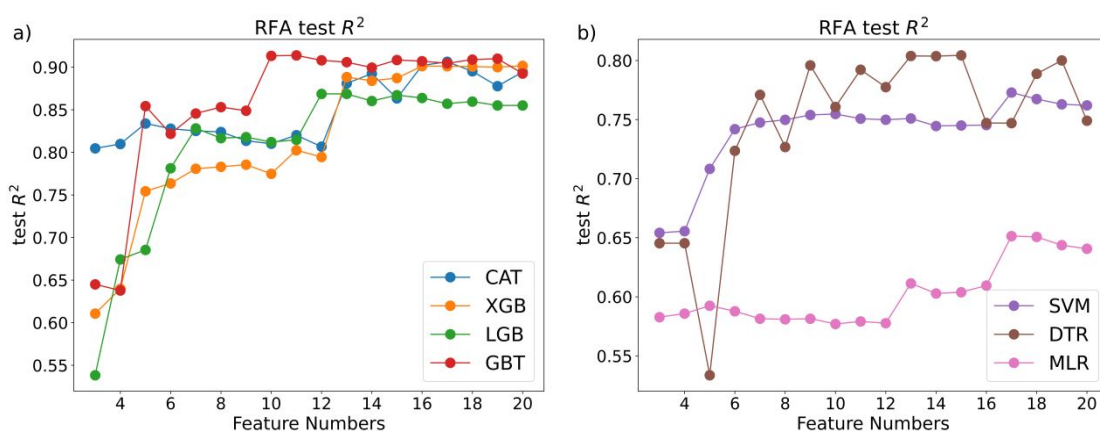


Figure S8 The changing trend of test set R^2 of the tree-based (a) and other (b) models along with the increasing number.

Code S5 SHAP Analysis

The feature contributions of each established model were calculated using SHAP package.¹⁷ As shown in Figure S9, the training and test set loaded from **Code S3** were passed to module “fml.pipelines.SHAPModelling” to obtain the filtered training and test set for each algorithm along with the selected feature number determined in **Code S4**, namely 13, 13, 12, and 10 for CAT, XGB, LGB, and GBT models respectively. Then the SHAP values would be calculated based on the filtered respective training and test set for each model. The extracted feature contributions could be seen in Figure S10.

```

from fml.pipelines import SHAPModelling
from fml.feature_selection import Shap
import joblib
import matplotlib.pyplot as plt
plt.rcParams["figure.dpi"] = 600

from xgboost import XGBRegressor as XGB
from catboost import CatBoostRegressor as CAT
from lightgbm import LGBMRegressor as LGB
from sklearn.ensemble import GradientBoostingRegressor as GBT

# Load the training and test sets from the step Code3
train = joblib.load("../code3//train_1805_1959.joblib")
test = joblib.load("../code3//test_1805_1959.joblib")

# Define the feature numbers resulted from Code4
feature_numbers = {
    CAT: 13,
    XGB: 13,
    LGB: 12,
    GBT: 10,
}

# Define the model abbreviations
abbrs = {
    CAT: "CAT",
    XGB: "XGB",
    LGB: "LGB",
    GBT: "GBT",
}

# Iterate 4 models
for algo in [CAT, XGB, LGB, GBT]:
    # Transform the original training and test sets with the pre-defined optimal feature numbers
    train_, test_ = SHAPModelling().fit(algo, train.copy(), test.copy()).transform(feature_numbers[algo])
    # Use fml.Shap module to perform SHAP calculations automatically
    explainer = Shap().fit(algo, train_)
    # Plot the SHAP importance
    explainer.bar(max_display=20, xlabel=f"{abbrs[algo]} SHAP Values")

```

Figure S9 The code to calculate SHAP values for the fitted tree-based models.

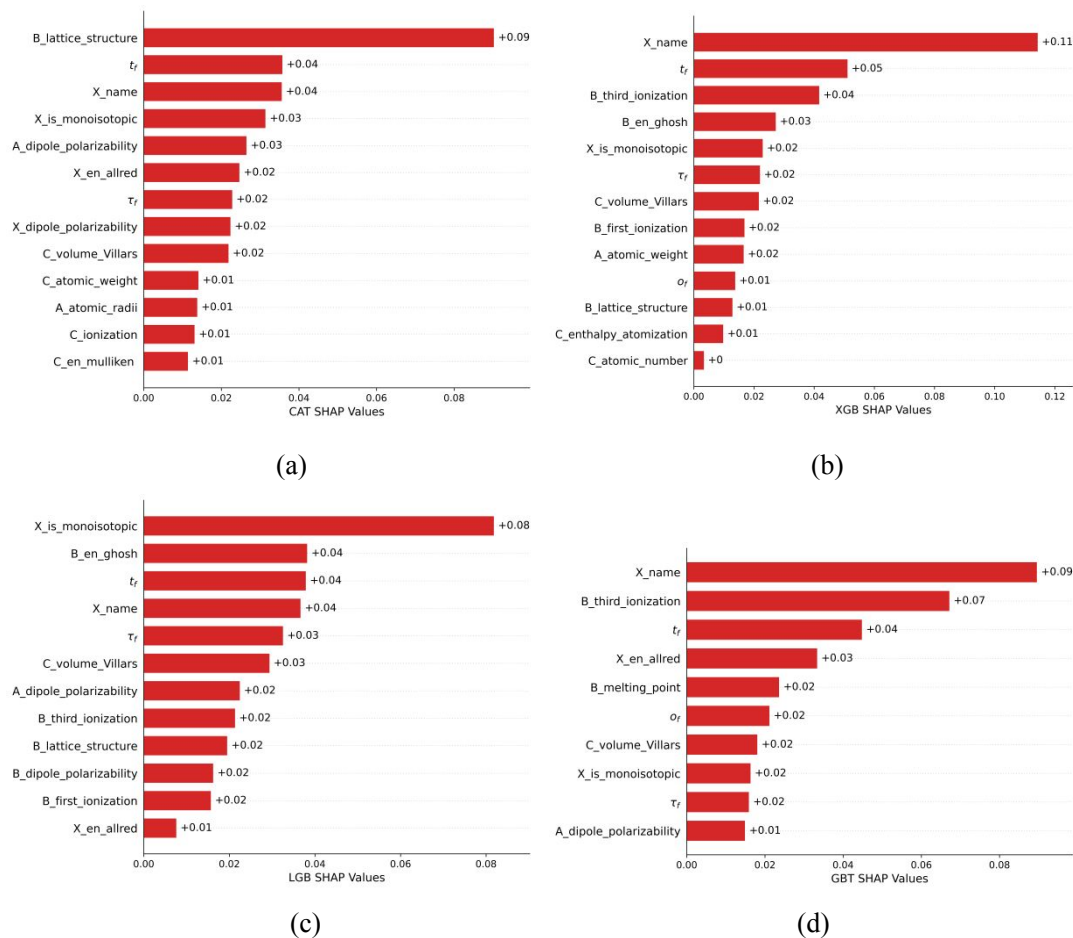


Figure S10 SHAP values of CAT (a), XGB (b), LGB (c), GBT (d) models.

Code S6 Hyper-parameter Optimization

The hyper-parameters were optimized for CAT, XGB, LGB, GBT models. The grid search (GS) module from scikit-learn¹⁴ may lack of more detailed processing information and not perform so efficiently in parallel calculations. Thence we used the self-developed module “fml.parameter_opt.GridSearch” to perform GS approach.

Herein, we consider the tree number, learning rate and tree depth for the optimization. As shown in Figure S11, the training and test set were loaded from **Code 3**, and the feature number were extracted from **Code 4**. The filtered training and test set were obtained from the module “fml.pipelines.SHAPModelling”, which was the same as **Code S5**. Then the grid search were performed using the module “fml.parameter_opt.GridSearch” along with the pre-defined algorithm (XGB in Figure S11), training set, test set, and the parameter ranges loaded from module “fml.configs.auto_config.gridsearch_parameters_reg”. We set the range 0.05~1 with step 0.05 for learning rate in XGB, LGB, GBT algorithms, while the learning rate will be automatically calculated in CAT algorithm according to the tree number value. The tree number (n_estimators) was set as 50~350 with step 10 for XGB, LGB, GBT algorithms, while 50~600 with step 50 for CAT algorithm. The tree depth (max_depth) was set as 3~8 with step 1 for XGB, LGB, GBT algorithms, while 4~10 with step 1 for CAT algorithm. Then the GS procedure would be running for hours depending on the computers. The full GS results could be accessed in “results”, while the best parameters could be seen in “best_p”. The results were outputted into table file and “./code6/*.joblib” file.

The model performance along with the best hyper-parameters were listed in Table S5. LOOCV R^2 of the CAT, XGB, LGB, GBT models were 0.928~0.950, while LOOCV RMSEs were 0.076~0.091, indicating the robust model performance. The corresponding test R^2 and test RMSEs were 0.883~0.918 and 0.102~0.122 respectively, showing the good predictivities. Also, we performed 5-folds cross-validation (CV5) and 10-folds cross-validation (CV10) for the models, whose R^2 were all over 0.90, which proved the models were trained well.

```

from fml.pipelines import SHAPModelling
from fml.parameter_opt import GridSearch
from fml.configs.auto_config import gridsearch_parameters_reg
import joblib

from xgboost import XGBRegressor as XGB
from catboost import CatBoostRegressor as CAT
from lightgbm import LGBMRegressor as LGB
from sklearn.ensemble import GradientBoostingRegressor as GBT

algorithm = XGB

# Load the training and test sets from the step Code3
train = joblib.load("../code3//train_1805_1959.joblib")
test = joblib.load("../code3//test_1805_1959.joblib")

# Define the feature numbers resulted from Code4
feature_numbers = {
    CAT: 13,
    XGB: 13,
    LGB: 12,
    GBT: 10,
}

# Define the model abbreviations
abbrs = {
    CAT: "CAT",
    XGB: "XGB",
    LGB: "LGB",
    GBT: "GBT",
}

# Transform the original training and test sets
# with the pre-defined optimal feature numbers
train_, test_ = SHAPModelling().fit(
    algorithm, train, test
).transform(feature_numbers[algorithm])

# Perform grid search (GS) approach
grid_p = gridsearch_parameters_reg[algorithm]
gs = GridSearch(n_jobs=10).fit(
    algorithm, train_, test_, cv=True, **grid_p
)

# GS results are exhibited as a table format
results = gs.results
# Obtain the best parameters and the best result
best_p = gs.best_p
best_result = gs.best_result
# Output the GS results into a table
results.to_excel(f"{algorithm}_gs_results.xlsx")

import joblib
joblib.dump(gs, f"gs_{algorithm}.joblib")

```

Figure S11 The code to optimize the hyper-parameters for the CAT, XGB, LGB, GBT models.

```

gridsearch_parameters_reg = {
    XGBRegressor: {
        "learning_rate": linspace(0.05, 1, 0.05),
        "n_estimators": linspace(50, 350, 10, dtype=int),
        "max_depth": linspace(3, 8, 1, dtype=int),
        "n_jobs": [1]
    },
    GradientBoostingRegressor: {
        "learning_rate": linspace(0.05, 1, 0.05),
        "n_estimators": linspace(50, 350, 10, dtype=int),
        "max_depth": linspace(3, 8, 1, dtype=int)
    },
    LGBMRegressor: {
        "learning_rate": linspace(0.05, 1, 0.05),
        "n_estimators": linspace(50, 350, 10, dtype=int),
        "max_depth": linspace(3, 8, 1, dtype=int),
    },
    CatBoostRegressor: {
        "max_depth": linspace(4, 10, 1, dtype=int),
        "n_estimators": linspace(50, 600, 50, dtype=int),
        "verbose": [False],
        "thread_count": [1]
    }
}

```

Figure S12 Ranges of the hyper-parameters for CAT, XGB, LGB, and CAT algorithms.

Table S5 Model performance of CAT, XGB, GBT, LGB models. The validation methods involve training validation, leaving-one-out cross-validation (LOOCV), 5-folds cross-validation (CV5), 10-folds cross-validation (CV10), and test validation. The metrics involves determination coefficient (R^2), Pearson correlation coefficient (ρ), root mean squared error (RMSE), mean average error (MAE), and

mean squared error (MSE).

		CAT	XGB	LGB	GBT
Tree number		600	50	350	80
Learning rate		0.048	0.3	0.15	0.25
Tree depth		7	5	5	4
Feature number		13	13	12	10
Training	R^2	0.994	0.998	0.992	0.999
	ρ	0.997	0.999	0.996	0.999
	RMSE	0.025	0.017	0.031	0.013
	MAE	0.018	0.012	0.020	0.010
	MSE	0.001	0.000	0.001	0.000
LOOCV	R^2	0.950	0.934	0.932	0.928
	ρ	0.976	0.966	0.966	0.963
	RMSE	0.076	0.088	0.089	0.091
	MAE	0.048	0.058	0.057	0.060
	MSE	0.006	0.008	0.008	0.008
CV5	R^2	0.946	0.905	0.904	0.904
	ρ	0.974	0.952	0.951	0.951
	RMSE	0.079	0.105	0.106	0.106
	MAE	0.051	0.066	0.067	0.066
	MSE	0.006	0.011	0.011	0.011
CV10	R^2	0.936	0.920	0.920	0.903
	ρ	0.968	0.959	0.960	0.951
	RMSE	0.086	0.096	0.096	0.106
	MAE	0.053	0.063	0.061	0.068
	MSE	0.007	0.009	0.009	0.011
Test	R^2	0.906	0.894	0.883	0.918
	ρ	0.954	0.946	0.940	0.959
	RMSE	0.109	0.116	0.122	0.102
	MAE	0.054	0.068	0.069	0.059
	MSE	0.012	0.013	0.015	0.010
External Set	R^2	0.790	0.800	0.740	0.800
	ρ	0.894	0.902	0.870	0.903
	RMSE	0.068	0.066	0.076	0.066
	MAE	0.040	0.045	0.044	0.047
	MSE	0.005	0.004	0.006	0.004

Code S7 Weighted Voting Regressor

Inspired by the voting regressor (VR) in scikit-learn¹⁴, we developed a weighted voting regressor (WVR) that combines various machine learning regression models and return the weighted predicted values. There are two main differences between the self-developed voting

regressor and the one of scikit-learn: i) the former combines the predictions of the sub-models by weights, while VR directly returns the average value; ii) the sub-models in WVR could use the individual feature set while the sub-models in VR use the same feature set. The utilization could be seen in Figure S13. The first part was to load the data set and the grid search results from the previous code part. Before fitting the WVR model, we needed to prepare 4 augments. The first was the variable containing algorithm modules that would be trained as sub-models. The second and third variables involved the training sets and test sets. The last one contained the model hyper-parameters that extracted from **Code S6**. An example form for the 4 augments was given in Figure S14. After fitting a WVR model, the best weights among the sub-models would be searched by using HyperOpt tool¹⁶. The search result might be different since HyperOpt does not perform a global search. In our case, a set of weights (Table S6) were outputted with 0.38 for CAT, 0.05 for XGB, 0.05 for LGB, and 0.51 for GBT model. The LOOCV R^2 and test R^2 reached 0.95 and 0.91 respectively, while the corresponding RMSEs were 0.079 and 0.106 respectively. The WVR model comprehended the outputs from the 4 sub-models, which has a more compliment model performance. The WVR model was outputted in “./code7/vr.joblib”.

The WVR model was also validated by predicting the bandgaps of external samples set of 42 HOIPs. The validating result could be seen in Table S7 and the code could be found in “./code7/code7_validating_48samples_2021.py”.

```

from fml.pipelines import SHAPModelling
from fml.ensemble import VotingRegressor
import joblib, glob

from xgboost import XGBRegressor as XGB
from catboost import CatBoostRegressor as CAT
from lightgbm import LGBMRegressor as LGB
from sklearn.ensemble import GradientBoostingRegressor as GBT

# Load the training and test sets from the step Code3
train = joblib.load("../code3/train_1805_1959.joblib")
test = joblib.load("../code3/test_1805_1959.joblib")

# Define the feature numbers resulted from Code4
feature_numbers = {
    CAT: 13,
    XGB: 13,
    LGB: 12,
    GBT: 10,
}

# Define the model abbreviations
abbrs = {
    CAT: "CAT",
    XGB: "XGB",
    LGB: "LGB",
    GBT: "GBT",
}

# Obtain the best parameters from code 6
best_parameters = { i.split("_")[1].split(".")[0]:joblib.load(i).best_p \
                    for i in glob.glob("../code6/*.joblib")}
best_parameters = { i:best_parameters[j] for i, j in abbrs.items() }

# Organize the algorithms, trains, tests, model parameters
trains, tests, model_ps, algos = [], [], [], []
for algorithm in [CAT, XGB, LGB, GBT]:
    train_, test_ = SHAPModelling().fit(
        algorithm, train.copy(), test.copy()
    ).transform(feature_numbers[algorithm])
    trains.append(train_)
    tests.append(test_)
    model_ps.append(best_parameters[algorithm])
    algos.append(algorithm)

# Fit the VotingRegressor
vr = VotingRegressor(verbose=True, rounds=100)
vr.fit(algos, trains, tests, model_ps)
# Get the results from vr object
results = vr.results
# Best weights
best_w = vr.best_weights
# Save the vr object
joblib.dump(vr, "vr.joblib")

```

Figure S13 The code to fit a voting regressor

```

algos = [CAT, XGB, LGB, GBT]
trains = [train_cat, train_xgb, train_lgb, train_gbt]
tests = [test_cat, test_xgb, test_lgb, test_gbt]
model_ps = [
    {'learning_rate': 0.03,
     'n_estimators': 800,
     'max_depth': 8,
     'verbose': False,
     'thread_count': 1},
    {'n_estimators': 50, 'learning_rate': 0.3, 'max_depth': 5, 'n_jobs': 1},
    {'learning_rate': 0.15000000000000002, 'n_estimators': 350, 'max_depth': 5},
    {'learning_rate': 0.25, 'n_estimators': 80, 'max_depth': 4}]

```

Figure S14 Example augments for fitting a weighted voting regressor

Table S6 Weights in weighted voting regressor.

	CAT	XGB	LGB	GBT
权重	0.381301	0.0520176	0.0520183	0.514664

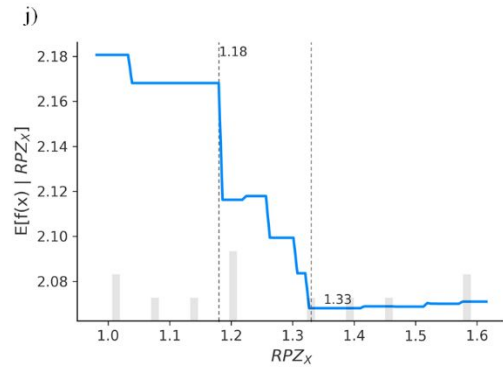
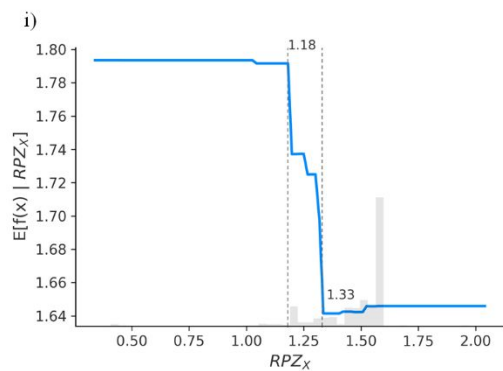
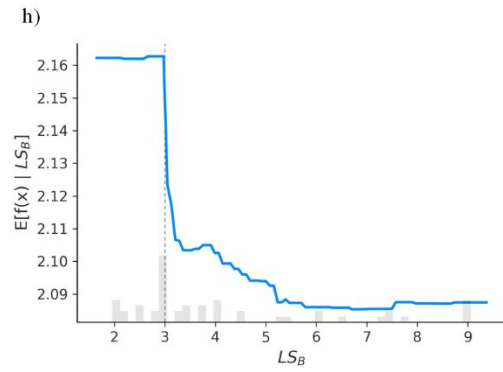
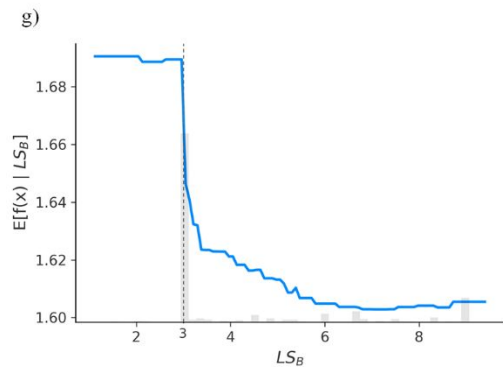
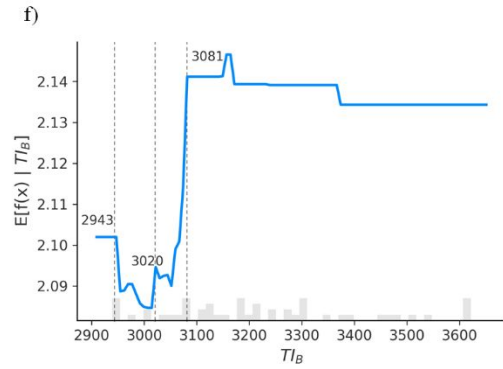
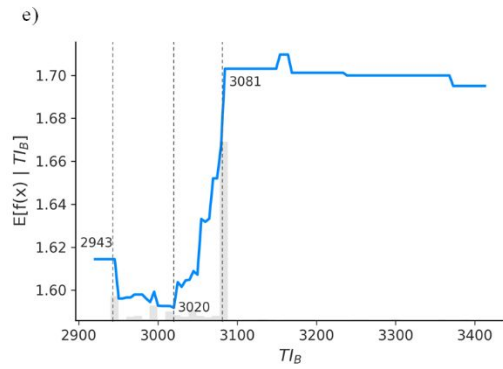
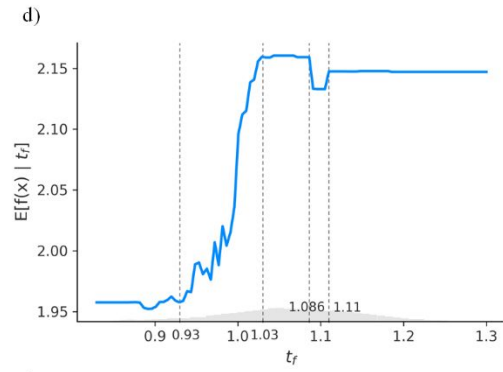
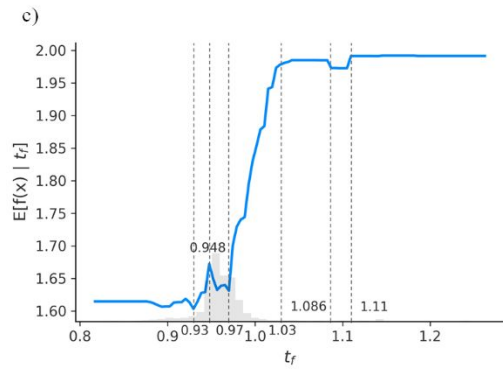
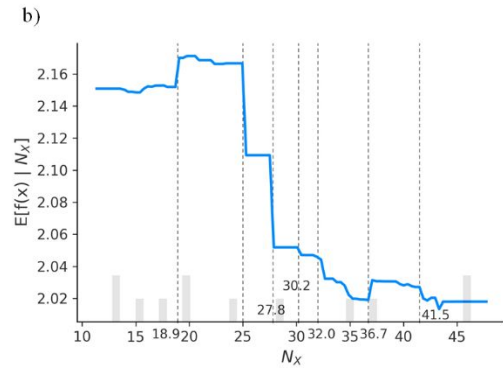
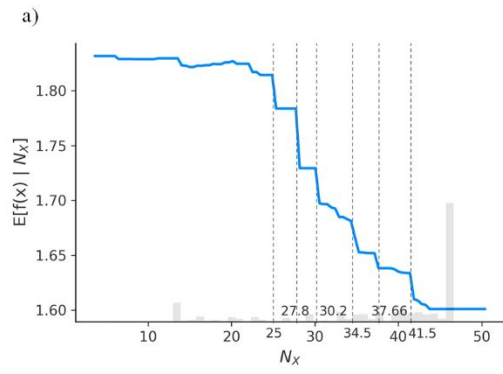
Table S7 Validating result for the external data set of 42 HOIPs.

VWR	Training	LOO	Test	CV5	CV10	External validating set of 42 HOIPs
R^2	0.997	0.946	0.912	0.939	0.937	0.837
ρ	0.999	0.973	0.956	0.970	0.969	0.919
RMSE	0.017	0.079	0.106	0.084	0.086	0.060
MAE	0.012	0.052	0.056	0.056	0.056	0.041
MSE	0.000	0.006	0.011	0.007	0.007	0.004

Code S8 SHAP Analysis for Weighted Voting Regressor

To perform the SHAP analysis based on WVR model, we built up a module “fml.ensemble.VotingShap” to overcome the incompatibility between WVR model and SHAP package. The SHAP values for experimental and virtual data set were calculated for various SHAP analysis, as seen in Figure S15 and Figure S16. The virtual data set was generated via the module “fml.searching.HOIPWithVotingRegressor” in Figure S17, whose chemical formular was defined as $A'_x A''_{1-x} B'_y B''_{1-y} C'_z C''_{3-z}$. As shown in Table S8, the elements in red color were chosen to supplement the virtual data set.

As for the calculating code, the procedures for both data set were almost the same except for the data set. Firstly, the WVR model was read as the “voting_model”. The “VotingShap” was fitted by passing the WVR model and the data set. The user could use the fitted “VotingShap” objects and the generated virtual data set. However, because of the large file size, these files were only supplied on our Github page (<https://github.com/luktian/InverseDesignViaPSP>). After fitting the “VotingShap” model, the SHAP values could read from the property “shap_values”. Then various SHAP plots could be drawn directly, which could be seen in Figure S19~



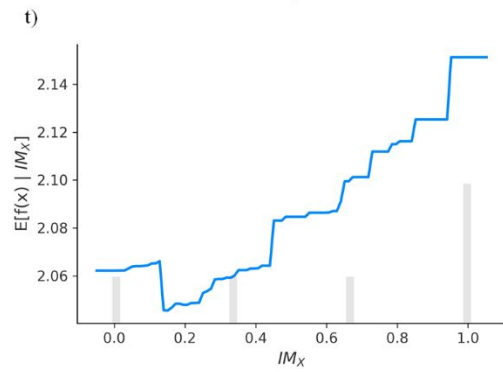
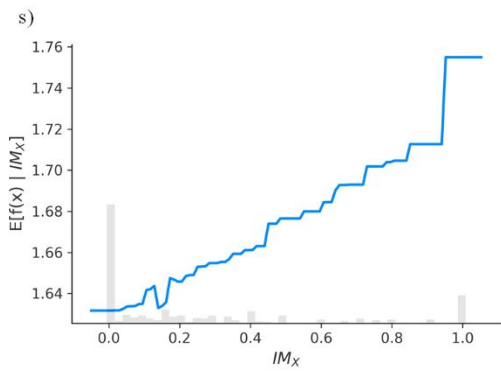
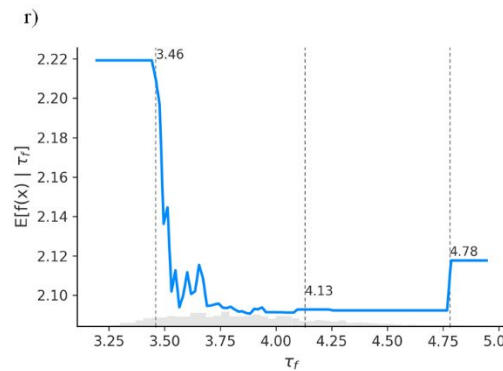
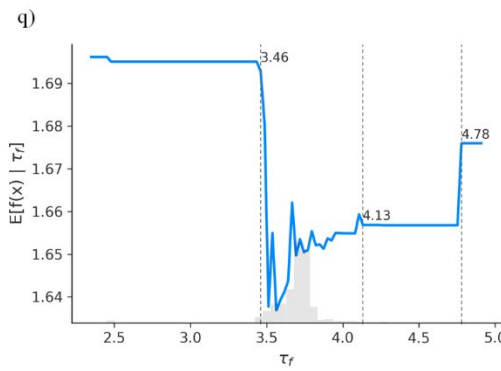
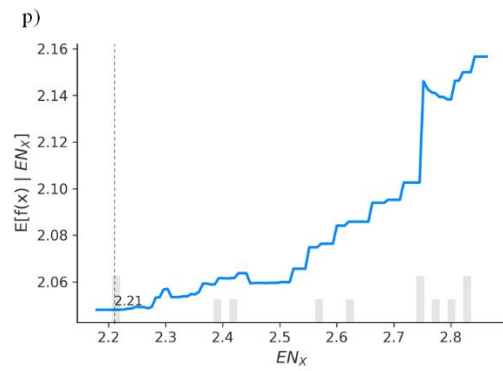
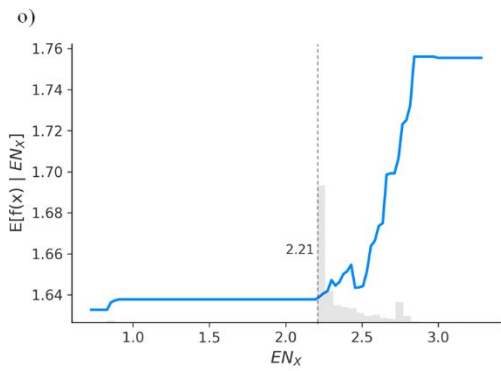
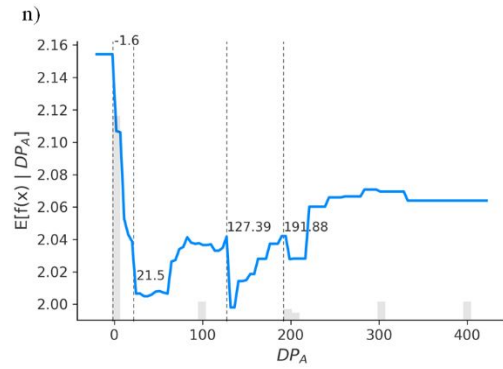
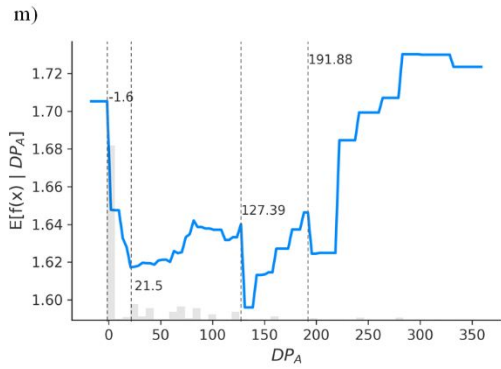
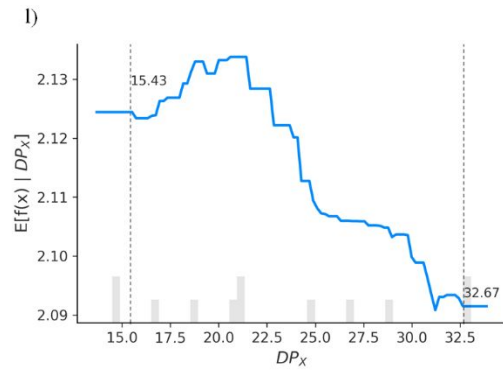
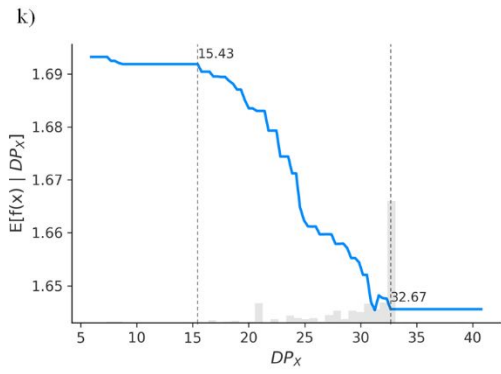


Figure S21.

Feature Analysis in Detail

Figure S18 (a) and (b) exhibited the top 10 features ranked by their contributions extracted from SHAP values in experimental and virtual data sets, whose vertical axis comprised feature ranking, and horizontal axis was the SHAP values for the features. The top 10 features in both data sets were identically the same, regardless of their ordering sequences. Specifically, the descriptors N_X (representing element name in X site) and t_f were the top 2 features in both of data set. The descriptors related to B site also had the important contributions to bandgap in experimental data set, which took the 3rd and 4th positions, followed by the descriptors related to A and X sites. Meanwhile in virtual data set, the B site descriptors were ranked at 9th and 10th, whose contributions were slightly lower than the descriptors in A and X sites.

Figure S18 (c) and (d) displayed the distributions of SHAP values of features for experimental and virtual data sets, where the horizontal axis comprised the sorted sample indexes according to model predictions that were presented in the top panel with their mean values (1.68 eV for experimental data set and 2.12 eV for the virtual one) signaled by the dotted line in gray color. The red/blue color expressed the positive/negative SHAP values for each sample and feature, which furtherly indicated the positive/negative contributions to predictions. The positive SHAP values for each feature (in red color) were mainly localized in the left part that corresponds to the higher bandgaps in both of the data sets, while the negative values (in blue color) were located at the right part related to the lower predictions, thus resulting a separately isolated distributions of positive/negative SHAP values for both data sets.

The scatter plots between features and SHAP values were drawn in Figure S19 with the color indicating prediction value to acquire a further understanding of each feature. Individual conditional expectation (ICE)²⁵ plots for each feature were drawn in Figure S20. ICE plots could signify the marginal effect that the feature had on the prediction, and revealed how the prediction of one sample (one line per sample) changed as the feature value increased. The partial dependence (PD)²⁶ plot was the average case that focused on the overall margin effect of the feature, which was drawn in Figure S21.

Starting from N_X , as shown in Figure S19 (a) and (b), the SHAP values for N_X had the decreasing trend in both data sets as the N_X values were increasing. The samples with lower N_X values and thence corresponding higher SHAP values tended to express the larger bandgaps (in darker red color) and the vice versa. By marking an inflection point (determining the SHAP values positive or negative) 34.72 of N_X in experimental data set, the N_X values over 34.72 would result in the negative SHAP values, and the ones lower than 34.72 led to the positive, while the inflection point in virtual data set was adapted to 26.33. The original values of N_X have been converted to numbers in the procedure of generating descriptors, in which chlorine, bromine and iodine were signaled by 20, 13, and 46 respectively. Hence, in pursuit of a higher/lower bandgap, the ratio of iodine in X site should be decreased/increased for a low/high N_X value that would trigger a high/low SHAP value, which was consistent to the current domain knowledges.²⁷

As indicated by the ICE and PD plots for N_X in Figure S21 (a ~ b) and Figure S22 (a ~ b), the same conclusion was extracted that the bandgap decreased as the N_X value increased. Besides, the steep points that triggered sharp decreases in predictions were labelled out for both of two data sets, whose locations were 25.00, 27.80, 30.20, 34.50, 37.66, 41.50 for experimental data set and 18.90, 25.00, 27.80, 30.20, 32.00, 36.70, 41.50 for virtual data set respectively. Four points of 25.00, 27.80,

30.20 and 41.50 were the same for both data sets, in which the steep point 25.00 for N_X might refer to the doped couple $\text{Br}_{1.915}\text{I}_{1.091}$ or $\text{Cl}_{2.423}\text{I}_{0.577}$, and 27.80 to $\text{Br}_{1.655}\text{I}_{1.345}$ or $\text{Cl}_{2.100}\text{I}_{0.900}$, 30.20 to $\text{Br}_{1.455}\text{I}_{1.545}$ or $\text{Cl}_{1.846}\text{I}_{1.154}$, 41.50 to $\text{Br}_{0.409}\text{I}_{2.591}$ or $\text{Cl}_{0.519}\text{I}_{2.481}$. As the ratios of iodine in these couples increased, their bandgap values were expected to receive steep decreases.

There were 4 other X-site-related descriptors that largely contributed to model predictions. RPZ_X was the element (in X site) radii defined by Zunger *et al.*²⁸, in which the radius of chlorine, bromine, and iodine were 1.01, 1.2, and 1.585 a.u. respectively. IM_X determined whether an element in X site had only a single stable isotope, whose values were defined 0 for iodine and 1 for chlorine/bromine. EN_X was the electronegativity defined by Allred *et al.*²⁹, in which electronegativities of chlorine, bromine, and iodine were 2.83, 2.74, and 2.21 respectively. DP_X represented the dipole polarizability of the element in X site, whose values were 14.60, 21.00, and 32.90 a.u. respectively. Presented by the scatter plots in Figure S19 (i, j, k, l, o, p, s, t), the higher/lower values of IM_X , and EN_X indicated the higher/lower proportions of chlorine/bromine and the higher/lower bandgap predictions, while the cases of RPZ_X and DP_X was the vice versa. The ICE and PD plots for the features in Figure S20 (i, j, k, l, o, p, s, t) and Figure S21 (i, j, k, l, o, p, s, t) also unveiled that the feature values of IM_X , and EN_X were proportional to the bandgap prediction while the values of the other X-related features were the reverse. Particularly, the RPZ_X range in 1.18 ~ 1.33 a.u. resulted in a steep decrease while the range over/lower than 1.88/1.33 a.u. led to unshaking tendencies. The inflection point of 1.18 a.u. might refer to the doped couple $\text{Cl}_{0.330}\text{Br}_{2.670}$ or $\text{Cl}_{2.130}\text{I}_{0.870}$, and the point 1.33 a.u. might refer to $\text{Br}_{1.980}\text{I}_{1.020}$ or $\text{Cl}_{1.350}\text{I}_{1.650}$. As the EN_X value was less than 2.21 that represented the electronegativity of iodine, the corresponding bandgap prediction was nearly unchanged, while the prediction started to augment after 2.21. The DP_X values in 15.43 ~ 32.67 a.u. would cause a sharp decrease of the bandgap, while the range over/lower than 32.67/15.43 a.u. led to the steady change. Herein the points signaled to the couples of $\text{Cl}_{2.610}\text{Br}_{0.390}$, $\text{Cl}_{2.580}\text{I}_{0.150}$, $\text{Br}_{0.060}\text{I}_{2.940}$, or $\text{Cl}_{0.030}\text{I}_{2.970}$.

TI_B signaled the third ionization energy of the element in B site, in which the energies of tin, lead, germanium, cadmium, and palladium were 2943, 3081, 3302, 3616, and 3177 kJ/mol respectively. LS_B stood for the lattice crystal structure of the simple substance for the elements in B site, whose values were tetragonal (TET, encoded as 9) for tin, face-centered cubic (FCC, encoded as 3) for lead/cadmium/palladium, and diamond cubic (DIA, encoded as 2) for germanium. Indicated by the scatter plots in Figure S19 (e, f, g, h) for the features, the TI_B value was proportional to its SHAP value and bandgap prediction in both of data sets, while the LS_B was the reverse situation. Combining the relevant ICE and PD plots of TI_B in Figure S20 (e, f, g, h) and Figure S21 (e, f, g, h), it could be noted that the steep point 2943 and 3081 kJ/mol were the third ionizations of tin and lead, which revealed the overall increasing trending of the bandgap as the proportion of lead arose in the doped B site couple SnPb. When the TI_B value was over 3081, the prediction was nearly unchanged and even slightly declined, indicating that the higher ratios of germanium/cadmium/palladium might have little influence on bandgap. The point of 3 in the ICE and PD plots of LS_B signaled the existence of lead/cadmium/palladium. The rising LS_B value represented the increasing proportion of tin, which further rendered the lower model prediction.

DP_A was the dipole polarizability of the element in A site, whose values were 400.90 a.u. for caesium (Cs), 289.70 a.u. for potassium, 1.07 a.u. for FA, 0.51 a.u. for MA, 1.13 a.u. for GA, 0.52 a.u. for EA, and 0 a.u. for ED. Because of the much smaller dipole polarizability of organic fragments, lots of samples with pure organic components in A site were concentrated around the

feature value of 0, whose SHAP values mingled either positive or negative. The positive ones might refer to the samples containing, *e.g.*, GA/EA/ED, whose bandgaps were much larger than [MA]BX₃ or [FA]BX₃, such as EA_{0.25-1}MA_{0.75-0}PbBr₃ (2.56 ~ 2.94 eV versus 2.30 eV of MAPbBr₃) and ED_{0.1-0.5}MA_{0.9-0.5}PbI₃ (1.58 ~ 2.10 eV versus 1.50 eV of MAPbI₃), while the negative samples might refer to the ordinary MAPbX₃ or FAPbX₃ (X referred to chlorine, bromine and iodine). As the additions of inorganic elements into MAPbX₃ or FAPbX₃, the SHAP value had a sluggish growing trend toward 0, which revealed that the higher ratio of inorganic elements was contributed to the higher bandgap.

As shown in Figure S19 (c), the SHAP values of t_f had an overall increasing trend with the ascendent t_f values, whose inflection point was 0.971 in experimental data set. In the case of virtual data set in Figure S19 (d), the t_f could be divided into three ranges. The SHAP values for t_f in the range 0.850~0.861 were mostly negative contributing to the lower bandgap, while the majority of SHAP values for t_f in 0.861~1.014 were negative and the ones in 1.014~1.277 were almost positive. Thence, for the purpose of higher/lower bandgaps, the t_f values should be controlled higher/lower than 1.014. Revealed by the ICE plots for t_f in Figure S20 (c, d), the predictions of each sample were shrinking from the range 1.21 ~ 2.95 eV to the range 1.76 ~ 2.75 eV. The PD plots in Figure S21 (c, d) displayed sigmoid-like function trends in experimental and virtual data sets, in which the t_f range below/above than 0.930/1.086 indicated steady prediction changes, and the data in the range of 0.930 ~ 1.086 resulted in a steep increment for bandgap values. From the scatter plots of t_f in Figure S19 (q, r), the inflection points 3.50 and 4.78 could be noticed in both of data sets. The SHAP values were mostly positive as the τ_f value was lower than 3.50 or higher than 4.78, while the values become negative as the τ_f value was in 3.50 ~ 4.78. The ICE plots in Figure S20 (q, r) for τ_f expressed that the prediction ranges were shrinking from 1.22 ~ 3.15 to 1.24 ~ 2.83 as the τ_f value arose. The PD plots in Figure S21 (q, r) showed that the τ_f ranges lower than 3.46, 4.13 ~ 4.78 and higher than 4.78 would render steady prediction trends, while the range 3.46 ~ 4.13 signaled an overall decreasing tendency. Taking both factors into consideration, if we seek HOIPs with lower bandgaps, the value of t_f should not be over 1.014 (or even lower than 0.971) and the value of τ_f is recommended to be 3.50 ~ 4.18 (4.18 determines the structure formability as indicated in reference³⁰), while HOIPs with $t_f > 1.014$ and $\tau_f < 3.5$ tend to own the larger bandgaps.


```

import joblib, rumpy as np
from fml.data import read_data
from fml.ensemble import VotingShap
from shap import plots
from copy import deepcopy
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 300

# read voting model
voting_model = joblib.load("../code7//vr.joblib")
# read experimental data set
dataset = read_data("../code2//H0IP_dataset_437.xlsx", df=False)
# fit voting shap object or the user could choose the supplied fitted object "votingshap_experimental.joblib"
vs = VotingShap(voting_model, dataset).fit() # vs = joblib.load("votingshap_experimental.joblib")
# read the feature names
columns = vs.columns_set[1:]
# read X and Y
X = vs.dataobject.to_df().iloc[:, 1:]
predicted_Y = voting_model.predict(dataset)
# read shap values
shap_values = vs.shap_values

# feature importance bar plot
plots.bar(deepcopy(shap_values), max_display=11)

# instance distribution plot
instance_order = np.array(sorted(zip(np.arange(len(X)), predicted_Y), key=lambda x: x[1], reverse=True))[:, 0].astype(int)
plots.heatmap(deepcopy(shap_values), max_display=11, instance_order=instance_order)

# decision plot
plots.decision(shap_values.base_values[0], deepcopy(shap_values.values), feature_names=columns)

# beeswarm plot
plots.beeswarm(deepcopy(shap_values), order=vs.feature_order, max_display=11)

# violin plot
plots.violin(deepcopy(shap_values.values), X, feature_names=columns, max_display=11)

# scatter plots for each feature
for findex in vs.feature_order[:10]:
    fname = X.columns[findex]
    plots.scatter(deepcopy(shap_values[:, findex]), color=predicted_Y)

# ICE plots for each feature
for findex in vs.feature_order[:10]:
    fname = X.columns[findex]
    plots.partial_dependence(findex, vs.predict, X, feature_names=columns)

# PD plots for each feature
for findex in vs.feature_order[:10]:
    fname = X.columns[findex]
    plots.partial_dependence(findex, vs.predict, X, feature_names=columns, ice=False)

```

Figure S15 The code to perform SHAP analysis for experimental data set

```

import joblib, numpy as np
from fml.ensemble import VotingShap
from shap import plots
from copy import deepcopy
import matplotlib.pyplot as plt
plt.rcParams["figure.dpi"] = 300

# read voting model
voting_model = joblib.load("../code7//vr.joblib")
# read virtual data set
dataset = joblib.load("dataset_virtual.joblib")
dataset = dataset.astype(float)
# fit voting shap object or the user could choose the supplied fitted object "votingshap_virtual.joblib"
vs = VotingShap(voting_model, dataset).fit() # vs = joblib.load("votingshap_virtual.joblib")
# read the feature names
columns = vs.columns_set[1:]
# read X and Y
X = vs.dataobject.to_df().iloc[:, 1:]
predicted_Y = voting_model.predict(dataset)
# read shap values
shap_values = vs.shap_values

# feature importance bar plot
plots.bar(deepcopy(shap_values), max_display=11)

# instance distribution plot
instance_order = np.array(sorted(zip(np.arange(len(X)), predicted_Y), key=lambda x: x[1], reverse=True))[:, 0].astype(int)
plots.heatmap(deepcopy(shap_values), max_display=11, instance_order=instance_order)

# decision plot
plots.decision(shap_values.base_values[0], deepcopy(shap_values.values), feature_names=columns)

# beeswarm plot
plots.beeswarm(deepcopy(shap_values), order=vs.feature_order, max_display=11)

# violin plot
plots.violin(deepcopy(shap_values.values), X, feature_names=columns, max_display=11)

# scatter plots for each feature
for findex in vs.feature_order[:10]:
    fname = X.columns[findex]
    plots.scatter(deepcopy(shap_values[:, findex]), color=predicted_Y)

# ICE plots for each feature
for findex in vs.feature_order[:10]:
    fname = X.columns[findex]
    plots.partial_dependence(findex, vs.predict, X, feature_names=columns)

# PD plots for each feature
for findex in vs.feature_order[:10]:
    fname = X.columns[findex]
    plots.partial_dependence(findex, vs.predict, X, feature_names=columns, ice=False)

```

Figure S16 The code to perform SHAP analysis for virtual data set

```

import joblib, pandas as pd
from fml.data import DataObject
from fml.searching import HOIPHTSWithVotingRegressor

# load voting regressor model
voting_model = joblib.load("../code7//vr.joblib")
# define site elements
site_elements = {'A': ['MA', 'FA', 'CS', 'GA', 'EA', 'ED'],
                 'B': ['Pb', 'Sn', 'Ge', 'Cd', 'Pd'],
                 'C': ['I', 'Cl', 'Br']}

# high-throughput calculating
hoip_hts = HOIPHTSWithVotingRegressor(True). \
    fit_with_full_range(voting_model,
                       list(site_elements.values()), [2, 2, 2],
                       steps=[.25, .25, 1.0], starts=[0, 0, 0],
                       ends=[1, 1, 3])

# generating the virtual data set
descriptors = pd.concat(hoip_hts.descriptors, axis=1).T
HTS_shap_data = DataObject(X=descriptors.values, Y=hoip_hts.predictions.values,
                           Xnames=descriptors.columns, indexes=hoip_hts.predictions.index,
                           Yname=hoip_hts.predictions.name)

# save to local
joblib.dump(HTS_shap_data.to_df(), "dataset_virtual.joblib")

```

Figure S17 The code to generate virtual data set

Table S8 Elements in A, B, C sites along with their count numbers. The references and SMILE formats were also given after table.

Element in A site	Count	Element in B site	Count	Element in C site	Count
MA	291	Pb	386	I	386
FA	269	Sn	135	Br	235
CS	145	Bi	11	Cl	26
GA	8	Cd	6		

ED	8	Ge	5		
Rb	7	Sr	3		
BA	6	Ca	3		
K	6	Cr	2		
EA	5	Pd	1		
PE	5	La	1		
HE	5				
A	3				
AN	1				
AZ	1				
AA	1				

MA -- C[NH3+] -- ref³¹;

FA -- NC=[NH2+] -- ref³²;

GA -- NC(N)=[NH2+] -- ref³³;

ED -- [NH3+]CC[NH3+] -- ref³⁴;

BA -- CCCC[NH3+] -- ref³⁵;

EA -- CC[NH3+] -- ref³⁶;

PE -- [NH3+]CCC1=CC=CC=C1 -- ref³⁷;

HE -- OCC[NH3+] -- ref¹⁰;

A -- [NH4+] -- ref⁵;

AN -- C1CC[NH2+]C1 -- ref³⁸;

AZ -- C[NH+](C)CCC[NH3+] -- ref³⁹;

AA -- C[NH2+]C -- ref⁴⁰.

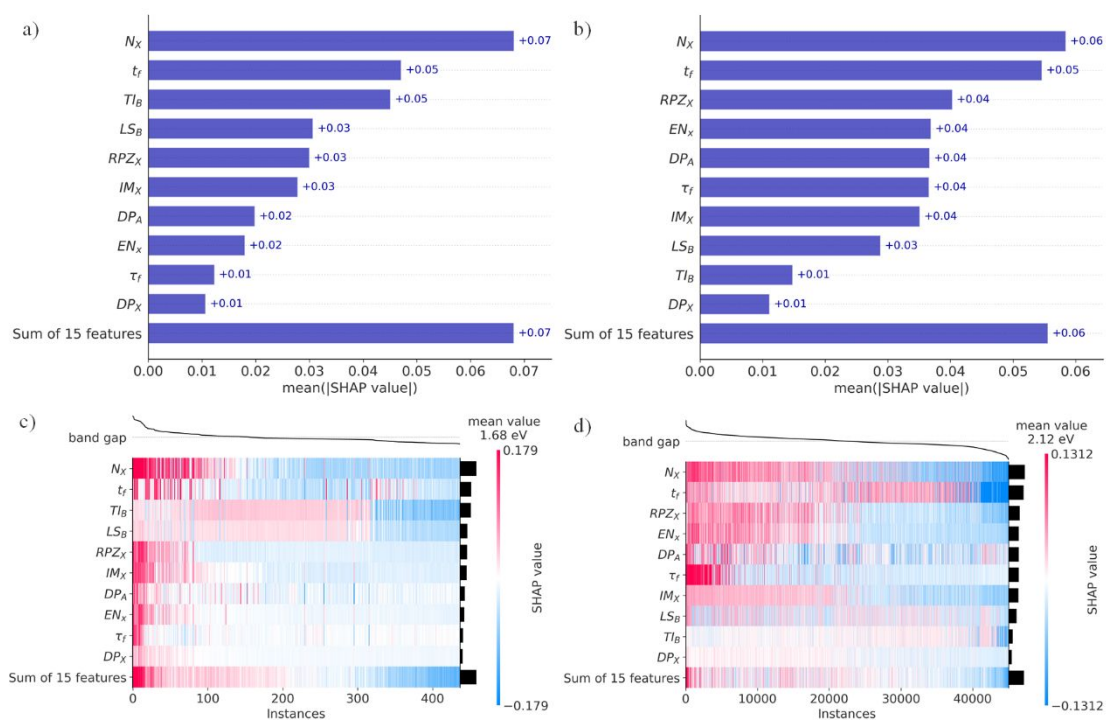
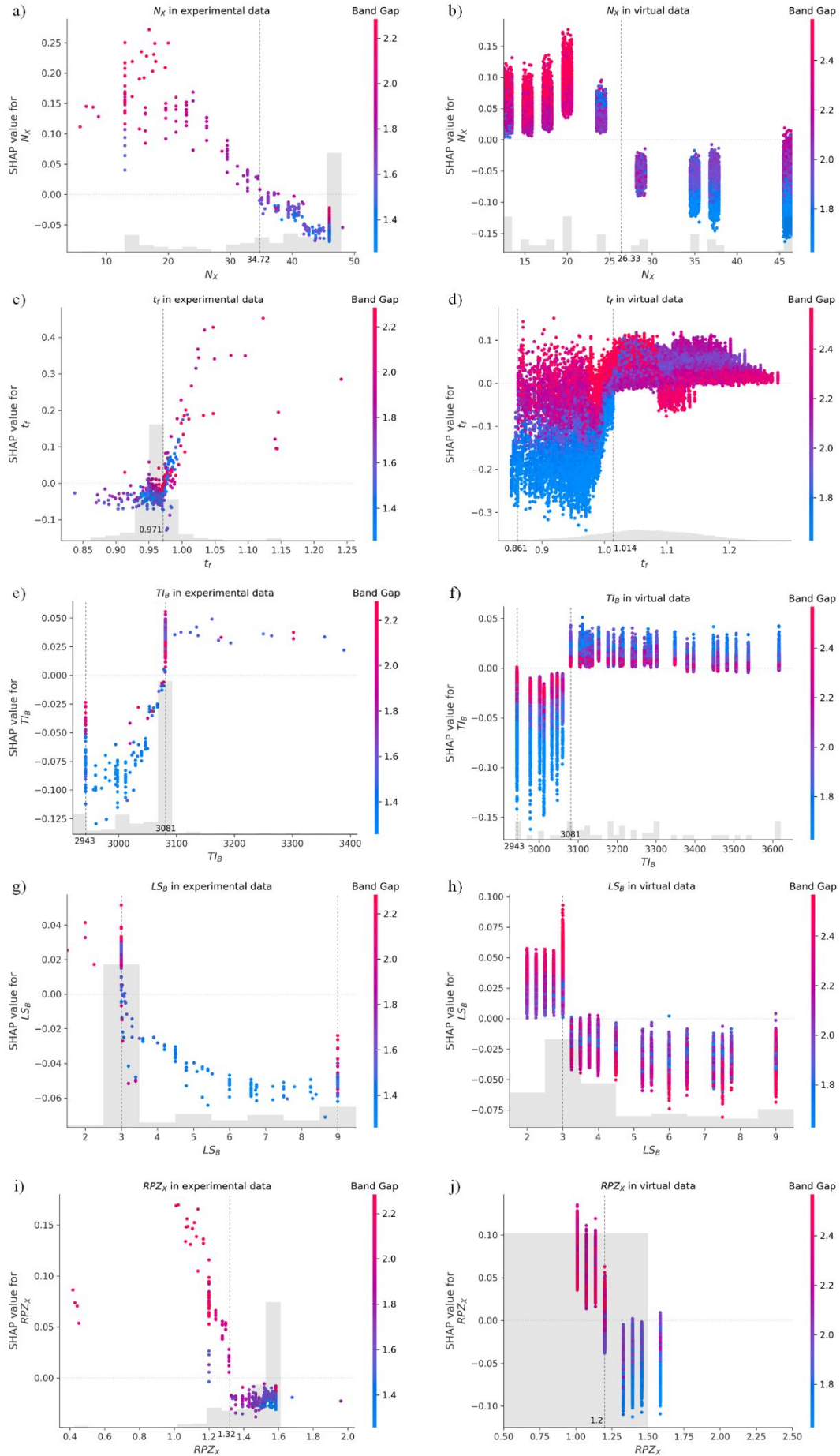


Figure S18 Feature importance for experimental data set (a) and virtual data set (b); Distributions of SHAP values with feature values for experimental data set (c) and virtual data set (d)



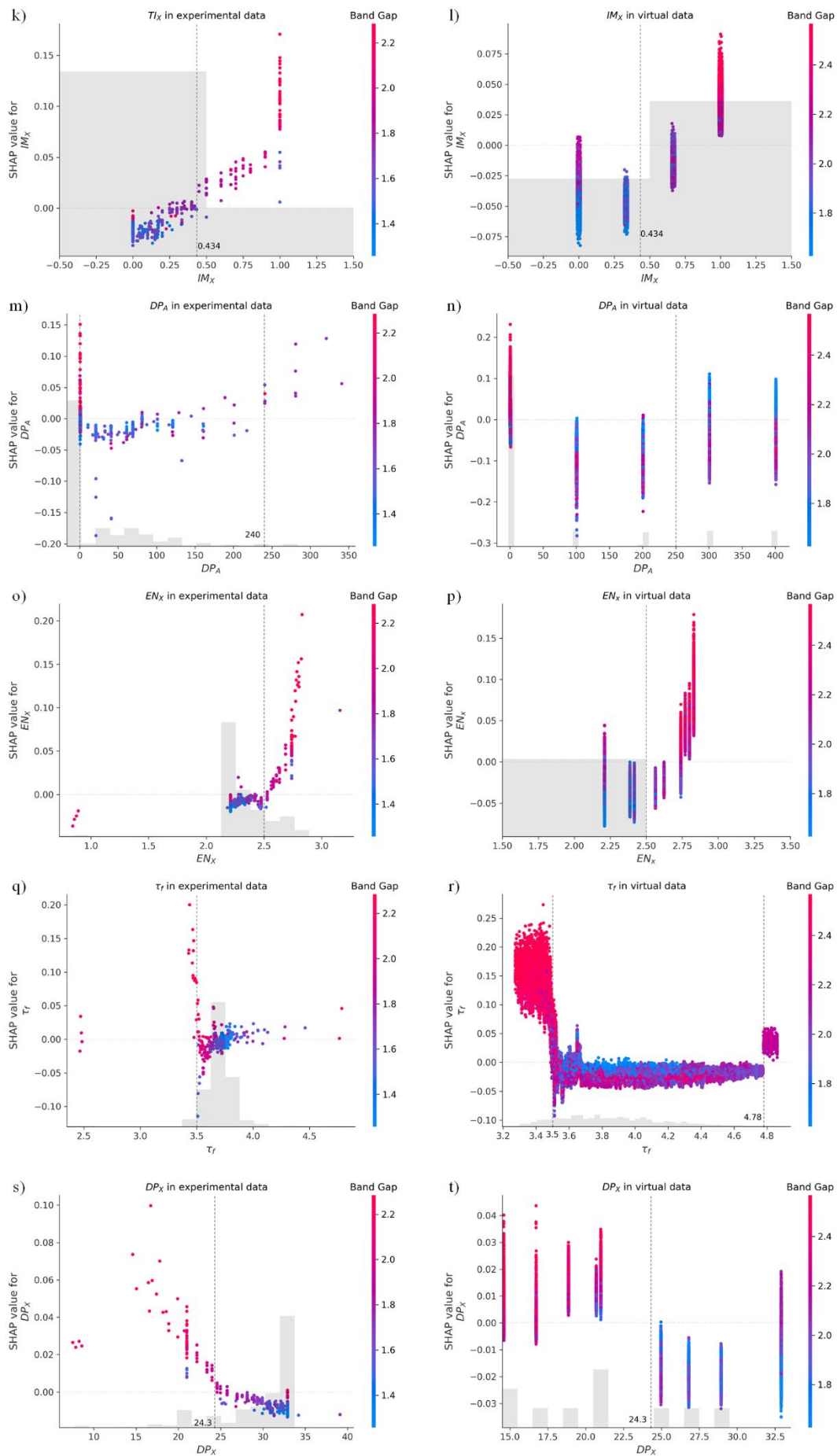
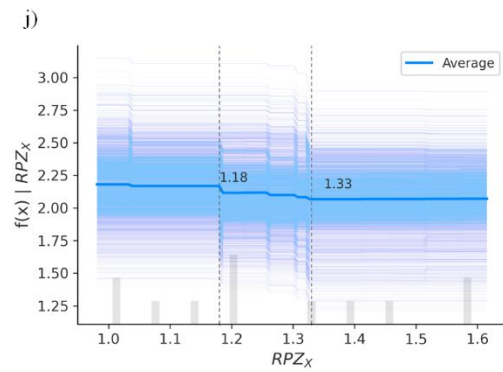
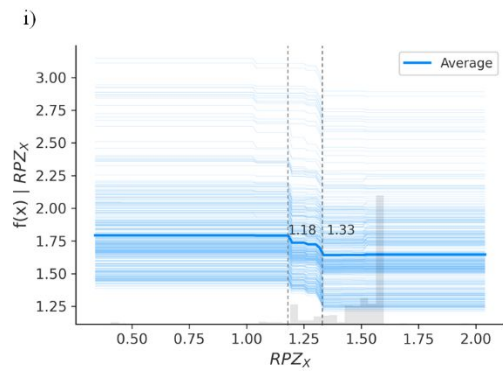
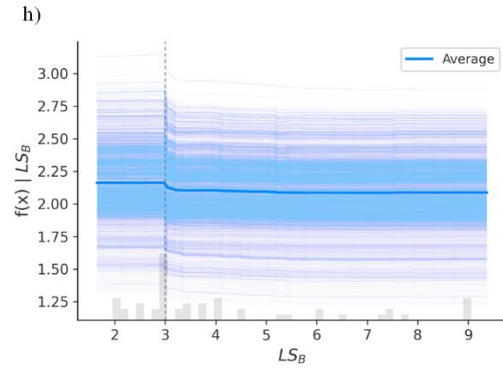
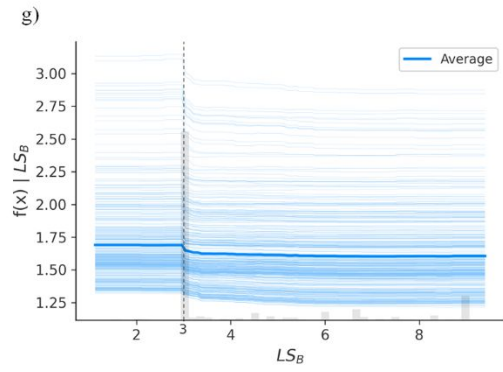
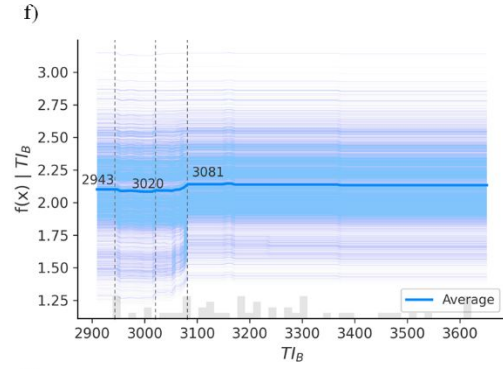
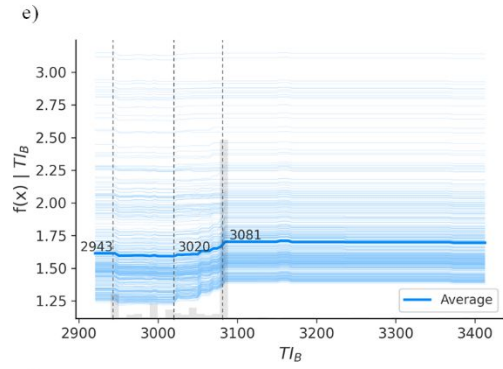
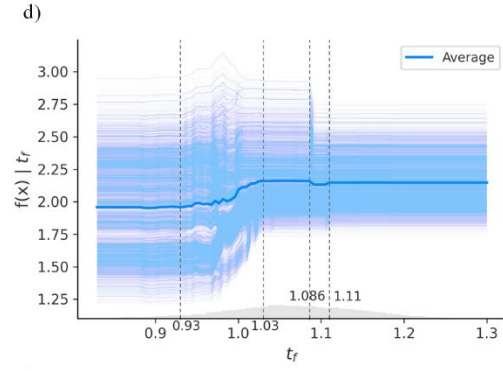
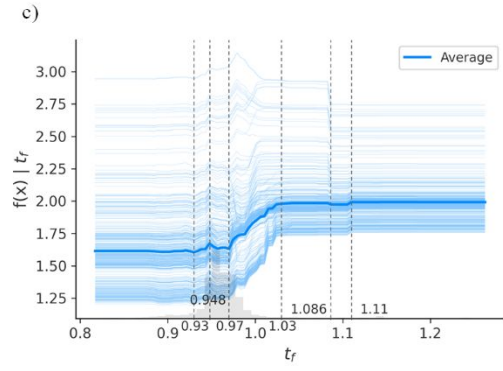
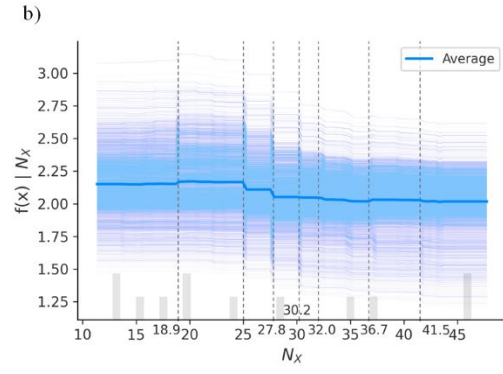
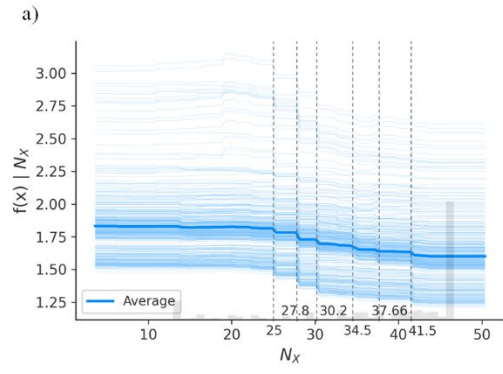


Figure S19 Scatter plots of the features N_x (a), t_f (c), TI_B (e), LS_B (g), RPZ_X (i), IM_X (k), DP_A (m), EN_X (o), τ_f (q), DP_X (s) in experimental data, and the features N_x (b), t_f (d), TI_B (f), LS_B (h), RPZ_X (j), IM_X (l), DP_A (n), EN_X (p), τ_f (r), DP_X (t) in virtual data.



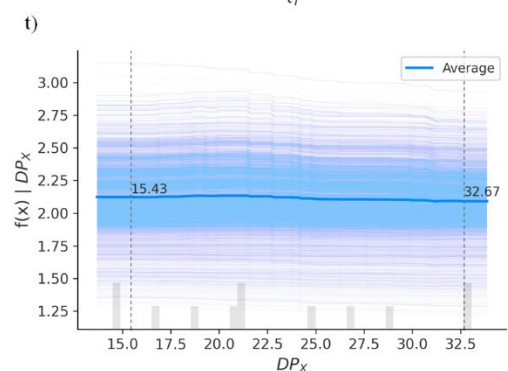
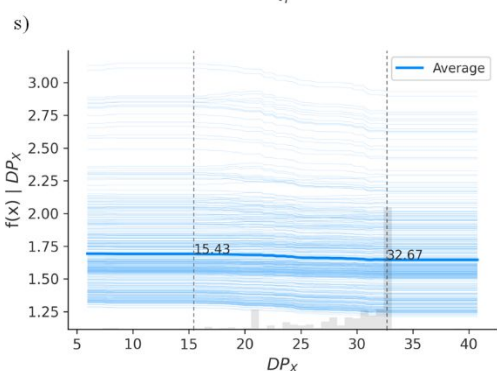
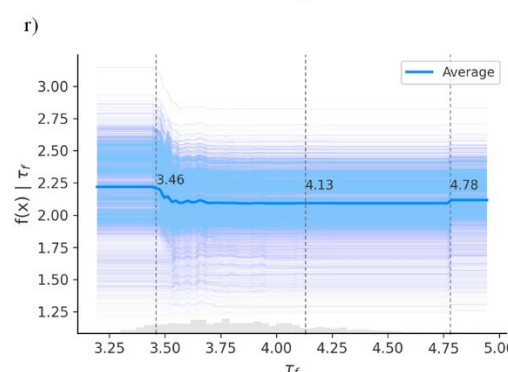
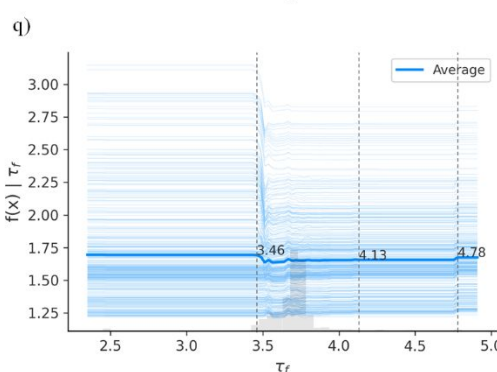
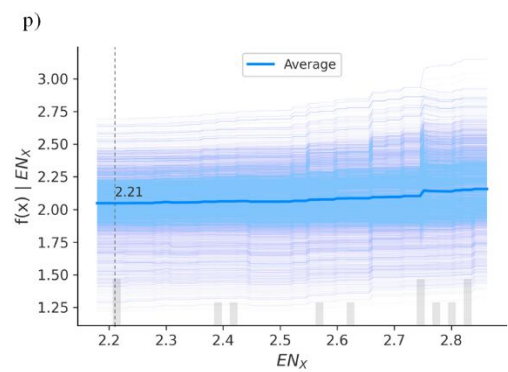
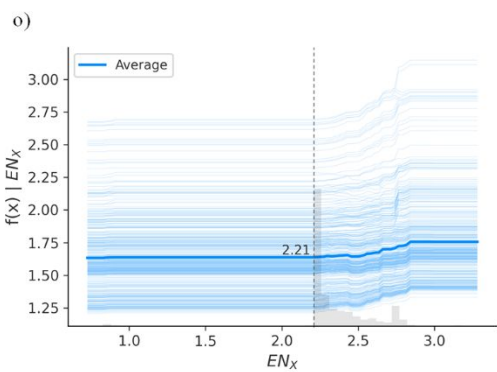
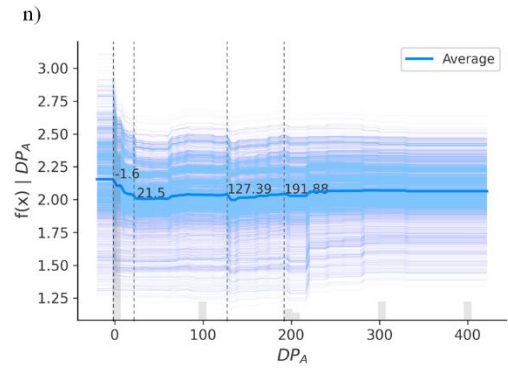
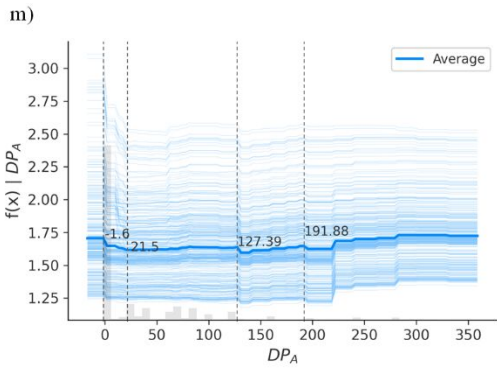
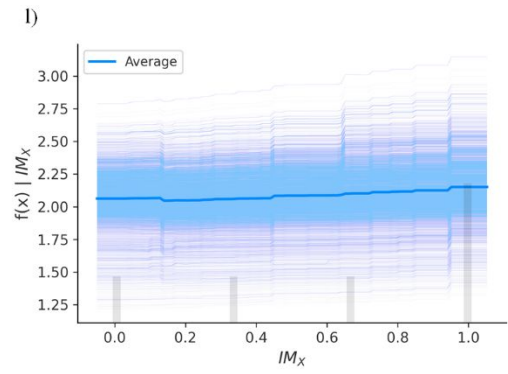
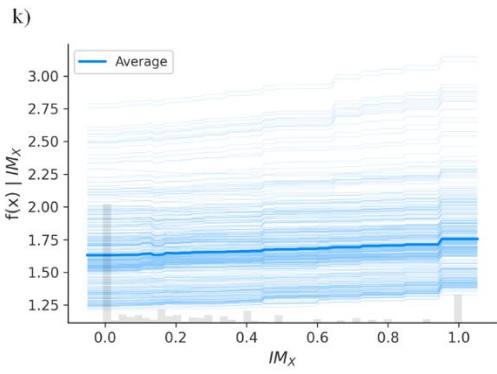
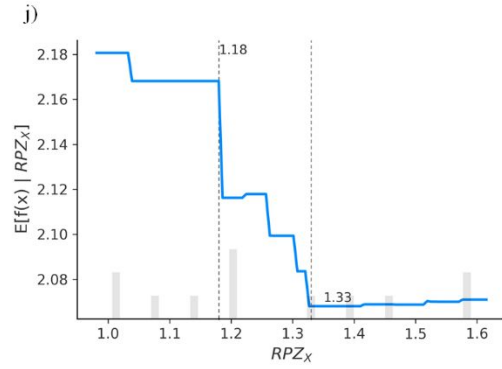
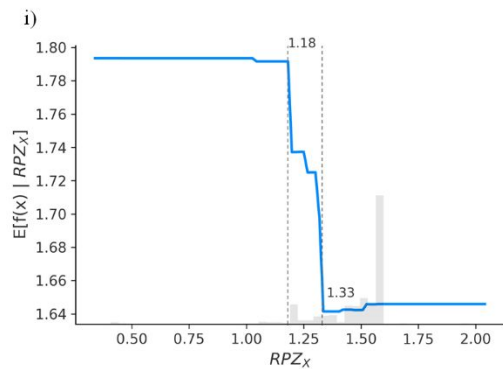
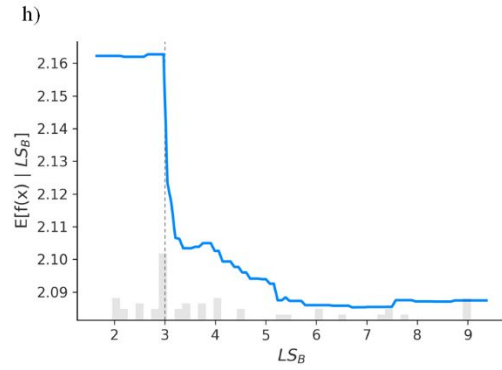
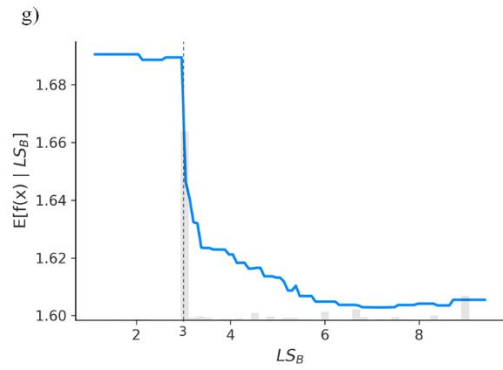
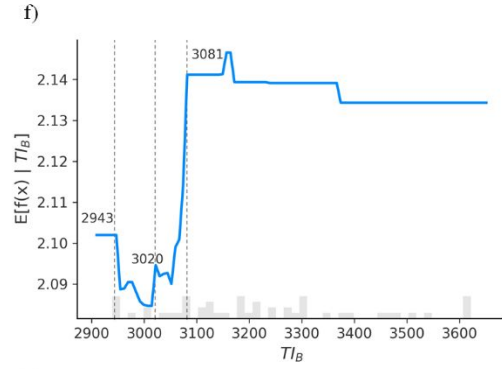
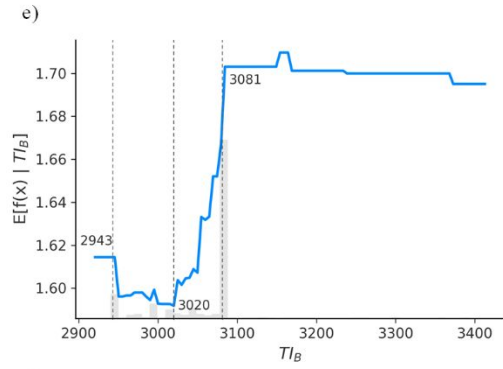
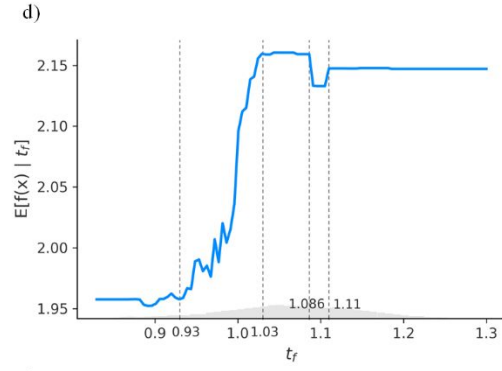
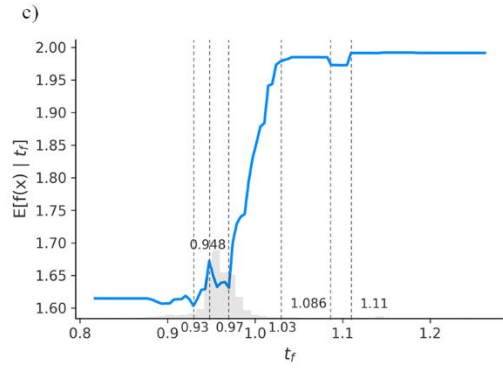
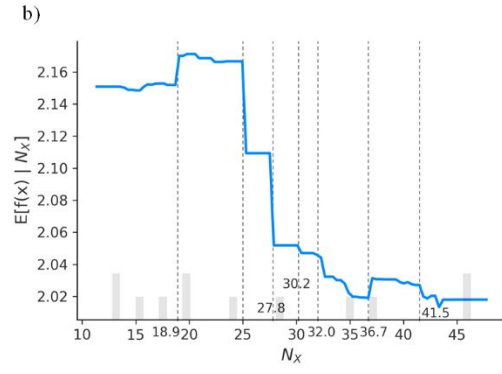
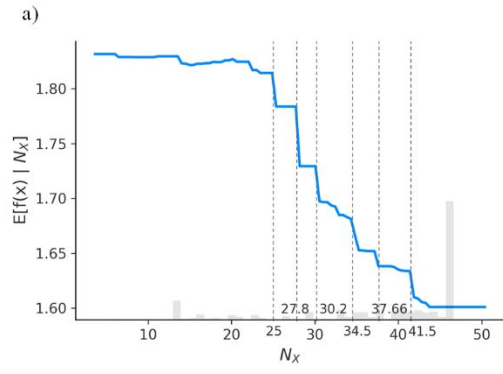


Figure S20 Individual conditional expectation (ICE) plots of the features N_x (a), t_f (c), TI_B (e), LS_B (g), RPZ_X (i), IM_X (k), DP_A (m), EN_X (o), τ_f (q), DP_X (s) in experimental data, and the features N_x (b), t_f (d), TI_B (f), LS_B (h), RPZ_X (j), IM_X (l), DP_A (n), EN_X (p), τ_f (r), DP_X (t) in virtual data.



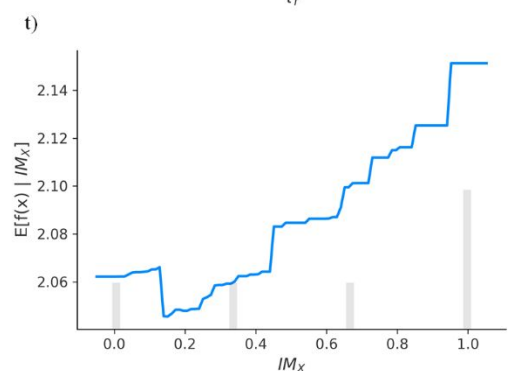
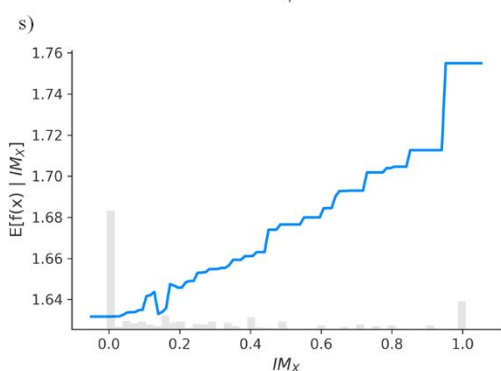
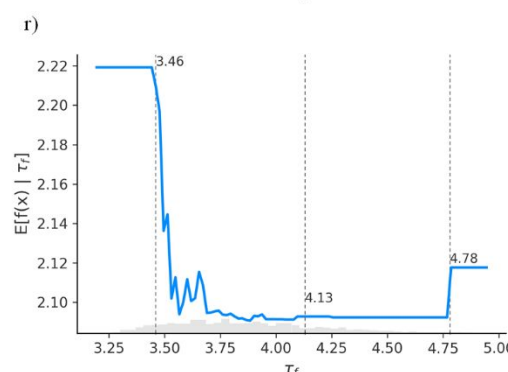
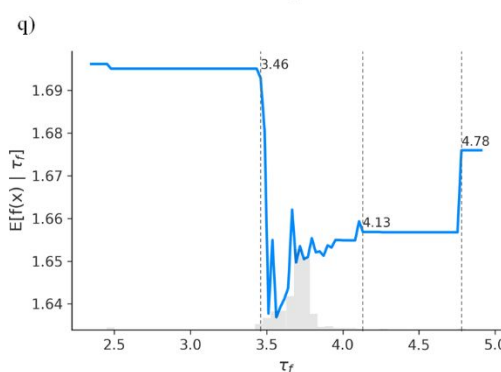
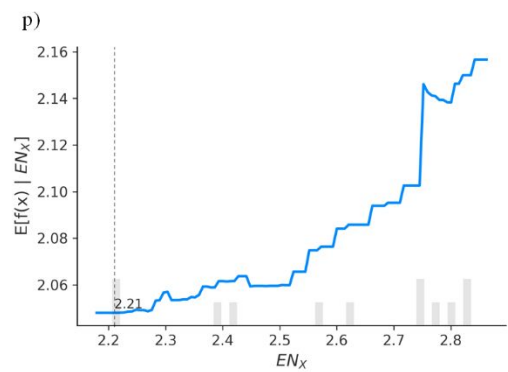
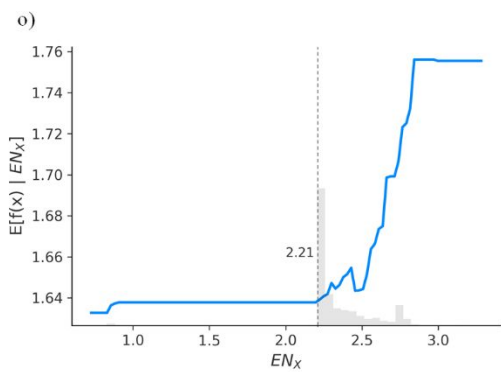
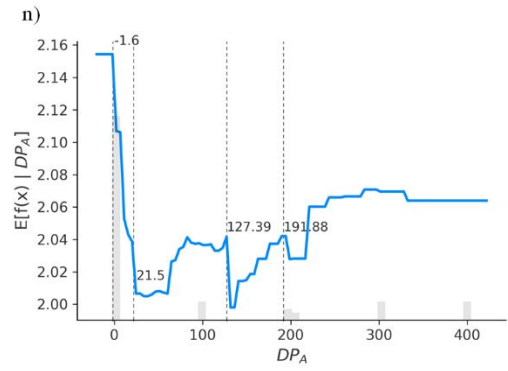
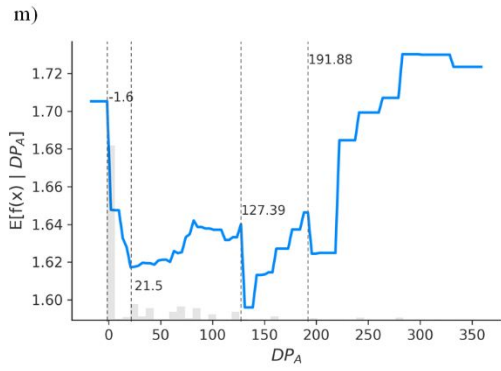
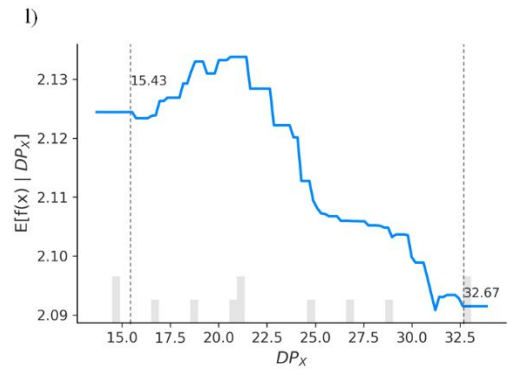
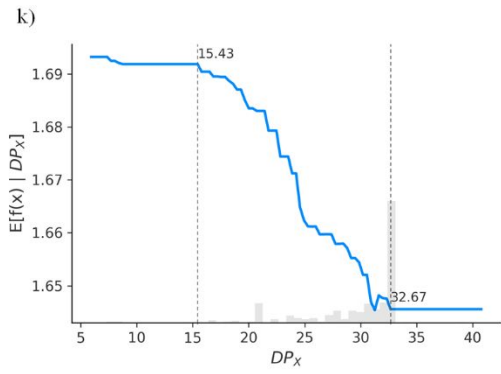


Figure S21 Partial dependency (PD) plots of the features N_x (a), t_f (c), TI_B (e), LS_B (g), RPZ_X (i), DP_X (k), DP_A (m), EN_X (o), τ_f (q), IM_X (s) in experimental data, and the features N_x (b), t_f (d), TI_B (f), LS_B (h), RPZ_X (j), DP_X (l), DP_A (n), EN_X (p), τ_f (r), IM_X (t) in virtual data.

Code S9 Proactive Searching Progress (PSP)

A so-called proactive searching progress (PSP) (Figure S22) was proposed to exert an efficient searching for the potential materials with expected values. The first step in this code is to load the established WVR “voting_model” from Code S7 and assemble the feature names into “columns_set”. The collected 88 organic fragments and 3 three metal elements were considered for A site, while 9 metal elements and 3 halogen elements were prepared for B/C site. The criterion was set as 0.05 eV determining whether the potential materials could be retained. The PSP could be run for multiple times for various initial point to start searching. The expected value was set as a fixed number or chosen from a list that contained multiple values. The doping form was needed to be defined such as 3, 3, 3 which meant for each site there were 3 elements/fragments. The iteration determined the searching steps in PSP. The results for each run of PSP would be recorded into local file in .csv format. Then the PSP would be exerted by using the module “HOIPHyperSearchingWithVotingRegressor”. The code is in “./code9/code9.py”, while the searching results were only accessible in our GitHub website (<https://github.com/luktian/InverseDesignViaPSP>) because of the large file size.

```

from fml.formulars import SplitFormular
from fml.descriptors import base_descriptor
from fml.searching import HOIPHyperSearchingWithVotingRegressor
import joblib, numpy as np, pandas as pd, os, time, random

# Load WVR model
voting_model = joblib.load("../code7/vr.joblib")

# Load the feature names from WVR model
columns_set = []
for trainobj in voting_model.trainobjects:
    columns_set += trainobj.Xnames.tolist()
columns_set = list(set(columns_set))

# Set the organic fragments and elements for each site
formular_info = [
    base_descriptor.index[118:].tolist() + ["K", "Cs", "Rb"], # atoms in A site
    ["Pb", "Sn", "Ge", "Pd", "Bi", "Sr", "Ca", "Cr", "La"], # atoms in B site
    ["Cl", "Br", "I"] # atoms in C site
]

# Set the criterion value
criterion = 0.05
# The object used to split formulae
splitformular = SplitFormular()

start_time = time.time()

# Run the proactive searching progress (PSP) for multiple times
for i in range(1000):

    # Set multiple expectation values to exert different searching
    targeted_value = random.choice([1.35, 1.40, 1.45])
    # Or set a single value
    targeted_value = 1.35

    # Choose one doping form
    site_counts = np.array(random.choice([
        [1, 1, 1], [2, 1, 1], [1, 2, 1], [1, 1, 2],
        [1, 2, 2], [2, 1, 2], [2, 2, 1], [2, 2, 2], [3, 3, 3],
    ]))
    # Or set a single form
    site_counts = np.array([3, 3, 3])

    # Choose an iteration step in PSP
    iteration = random.choice(np.linspace(100, 1000, 11).astype(int))

    # Set the output file
    output_file = f"outputs[''.join(site_counts.astype(str))].csv"

    # Write information into output file
    if not os.path.exists(output_file):
        with open(output_file, "w") as f:
            f.writelines(f"formular, Prediction, {''.join(columns_set)}\n")
        existed_formulars = []
    else:
        existed_formulars = pd.read_csv(output_file, index_col=0).index.tolist()
        if len(existed_formulars) > 0:
            existed_formulars = splitformular.split_formulars(existed_formulars)[0]

    # Perform the PSP
    hhwvr = HOIPHyperSearchingWithVotingRegressor(iteration, True, True). \
        fit_with_full_range(voting_model, formular_info, targeted_value,
                           criterion, site_counts=site_counts)

    # Write the PSP results into output file
    if len(hhwvr.trials) > 0:
        trials = np.array(hhwvr.trials)
        trials_descriptors = pd.DataFrame(trials[:, 3:],
                                         index=trials[:, 0],
                                         columns=hhwvr.descriptor_names.tolist()
                                         )[columns_set].astype(float)

        predictions = trials[:, 1].astype(float)

        with open(output_file, "a") as f:
            for prediction, (formular, descriptors) in \
                zip(predictions, trials_descriptors.iterrows()):
                if splitformular.split_formular(formular) not in existed_formulars:
                    f.writelines(f"{formular}, {prediction}, {''.join(descriptors.astype(str).tolist())}\n")

end_time = time.time()
print(f"{end_time - start_time} s")

```

Figure S22 Code for proactive searching progress (PSP).

Section S2 Other Figures and Tables

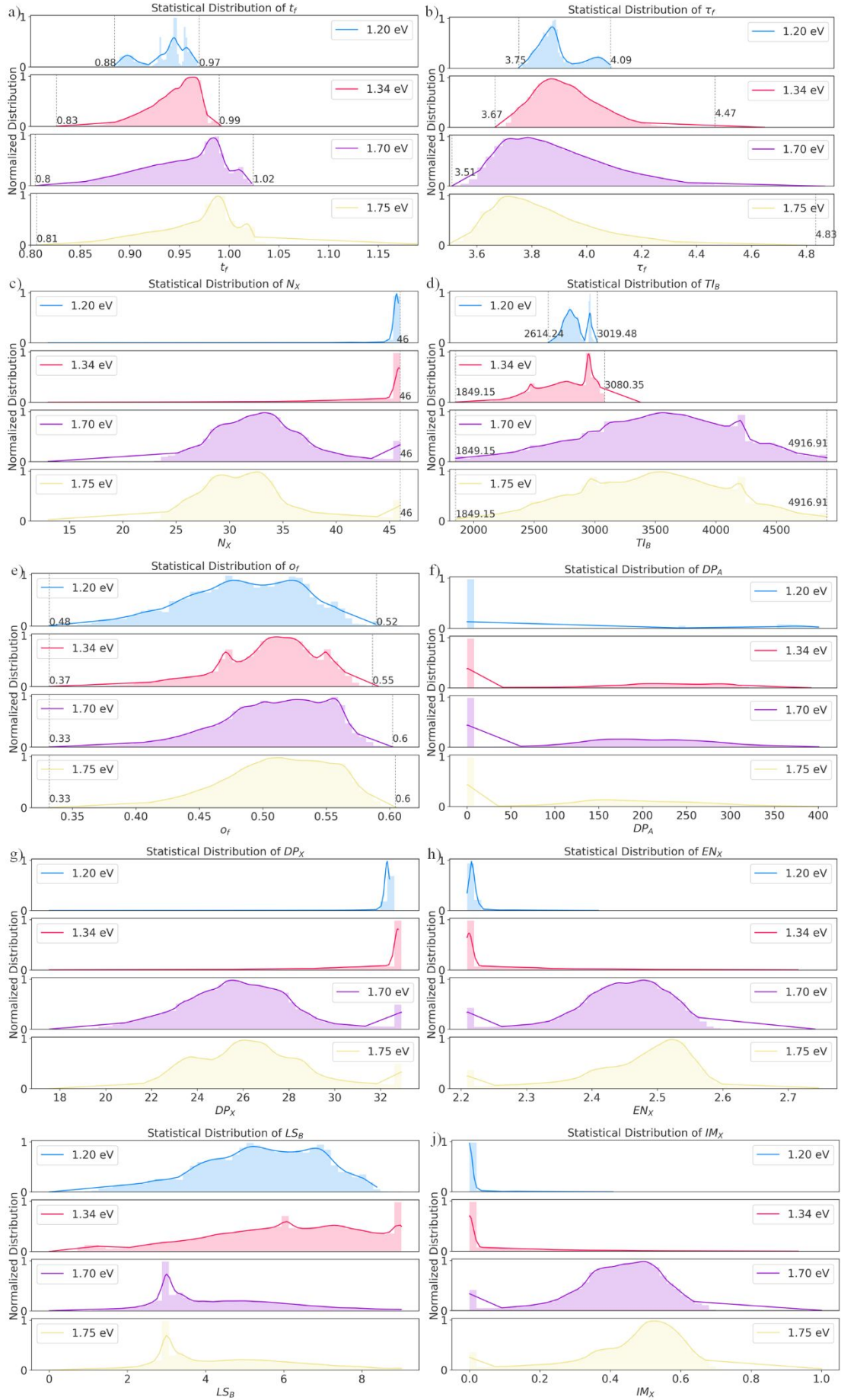


Figure S23 Feature distributions for 4 candidate sets.

Table S9 Searched candidates for 1.34 eV.

Triple divalent cations HOIP candidate	Predicted bandgap
MA0.215Cs0.432FA0.353Sn0.924Ge0.039Pd0.038I2.708Br0.29	1.3400
2	
Cs0.334FA0.266MA0.4Pd0.228Ge0.003Sn0.769I2.837Br0.164	1.3400
MA0.296FA0.338Cs0.366Cr0.104Sn0.8Pd0.096Br0.056I2.944	1.3401
FA0.412Cs0.509MA0.079Ge0.095Cr0.271Sn0.634Br0.109I2.89	1.3399
1	
MA0.289FA0.333Cs0.378Cr0.099Sn0.804Pd0.097Br0.005I2.995	1.3398
MA0.29FA0.339Cs0.372Cr0.001Sn0.8Pd0.199Br0.061I2.939	1.3398
MA0.345FA0.306Cs0.349Cr0.003Sn0.921Pd0.076Br0.005I2.995	1.3398
Cs0.307FA0.373MA0.32Ge0.019Sn0.939Ca0.041Br0.287I2.713	1.3403
MA0.282FA0.336Cs0.381Pd0.159Sn0.839Cr0.002Br0.003I2.998	1.3397
FA0.02MA0.693Cs0.287Cr0.168Pd0.037Sn0.795Br0.003I2.998	1.3396
MA0.314FA0.348Cs0.338Cr0.066Sn0.816Pd0.118Br0.176I2.824	1.3404
MA0.306FA0.338Cs0.356Cr0.097Sn0.81Pd0.093Br0.046I2.954	1.3404
Cs0.396MA0.035FA0.569Sn0.719Ge0.219Pd0.062Br0.013I2.98	1.3406
7	
Cs0.419FA0.253MA0.328Ge0.003Sn0.708Pd0.288Br0.019I2.98	1.3407
2	
FA0.462MA0.169Cs0.369Cr0.151Ge0.138Sn0.711I2.986Br0.01	1.3408
4	
MA0.305FA0.337Cs0.359Cr0.129Sn0.723Pd0.148Br0.05I2.95	1.3391
FA0.424MA0.185Cs0.39Cr0.158Ge0.162Sn0.68I2.989Br0.011	1.3390
FA0.403MA0.168Cs0.429Cr0.195Pd0.113Sn0.693I2.984Br0.016	1.3390
Cs0.423FA0.243MA0.334Ge0.059Sn0.713Pd0.228Br0.01I2.99	1.3411
MA0.289FA0.334Cs0.377Cr0.113Sn0.773Pd0.113Br0.057I2.943	1.3388
Cs0.429MA0.138FA0.433Sn0.765Ge0.182Pd0.053Br0.059I2.94	1.3412
1	
FA0.425Cs0.513MA0.062Ge0.142Cr0.122Sn0.735Br0.105I2.89	1.3387
5	
MA0.315FA0.335Cs0.35Cr0.088Sn0.777Pd0.135Br0.061I2.94	1.3414
MA0.304FA0.332Cs0.364Cr0.098Sn0.825Pd0.078Br0.054I2.946	1.3386
MA0.373Cs0.495FA0.133Cr0.243Pd0.081Sn0.676I2.902Br0.098	1.3386
MA0.278FA0.314Cs0.407Ge0.197Sn0.782Cr0.021Br0.047I2.95	1.3385
3	
FA0.466Cs0.499MA0.035Ge0.188Cr0.127Sn0.686Br0.196I2.80	1.3385
3	
MA0.288FA0.339Cs0.373Cr0.069Sn0.891Pd0.041Br0.061I2.939	1.3385
Cs0.436FA0.306MA0.258Ge0.004Sn0.732Pd0.264Br0.052I2.94	1.3384
8	
MA0.327FA0.316Cs0.357Cr0.022Sn0.869Pd0.11Br0.002I2.998	1.3384

MA0.296FA0.339Cs0.365Cr0.131Sn0.78Pd0.089Br0.044I2.956	1.3416
FA0.363Cs0.521MA0.116Ge0.174Cr0.224Sn0.602Br0.106I2.89	1.3383
3	
FA0.037MA0.746Cs0.217Cr0.073Pd0.047Sn0.88Br0.058I2.942	1.3418
MA0.302FA0.327Cs0.371Cr0.055Sn0.772Pd0.172Br0.003I2.997	1.3418
FA0.483Cs0.501MA0.016Ge0.138Cr0.157Sn0.704Br0.142I2.85	1.3418
8	
FA0.486Cs0.358MA0.155Cr0.123Pd0.163Sn0.714I2.998Br0.002	1.3381
MA0.324FA0.343Cs0.333Cr0.116Sn0.851Pd0.033Br0.162I2.838	1.3381
MA0.338FA0.347Cs0.315Cr0.062Sn0.865Pd0.073Br0.001I2.998	1.3419
FA0.432Cs0.369MA0.199Ge0.148Cr0.084Sn0.767Br0.12I2.881	1.3420
Double divalent cations HOIP candidate	Predicted bandgap
MA0.191FA0.403Cs0.406Sn0.76Cr0.24Br0.01I2.99	1.3400
FA0.177Cs0.353MA0.471Cr0.279Sn0.721I2.995Br0.004	1.3400
Cs0.494MA0.217FA0.289Sn0.866Ge0.134I2.623Br0.377	1.3400
FA0.235Cs0.214MA0.551Sn0.955Ge0.044I2.256Br0.744	1.3400
Cs0.299MA0.675FA0.025Sn0.816Pd0.183Br0.051I2.949	1.3400
MA0.211FA0.391Cs0.398Sn0.872Pd0.128Br0.031I2.968	1.3400
MA0.306FA0.325Cs0.369Cr0.262Sn0.738Br0.002I2.998	1.3400
Cs0.305FA0.273MA0.422Sn0.975Cr0.025I2.737Br0.264	1.3399
MA0.215FA0.406Cs0.379Sn0.87Pd0.131Br0.009I2.991	1.3399
MA0.295Cs0.378FA0.327Pd0.148Sn0.852I2.999Br0.001	1.3399
MA0.485FA0.07Cs0.445Sn0.764Cr0.236Br0.032I2.969	1.3401
Cs0.181MA0.409FA0.41Cr0.096Sn0.905I2.995Br0.005	1.3399
MA0.191FA0.415Cs0.394Sn0.892Pd0.108Br0.037I2.963	1.3399
FA0.592MA0.333Cs0.075Sn0.957Ge0.044I2.745Br0.255	1.3399
Cs0.248MA0.343FA0.409Pd0.246Sn0.754I2.962Br0.039	1.3399
Cs0.256MA0.524FA0.22Sn0.873Pd0.127I2.993Br0.007	1.3401
Cs0.23MA0.401FA0.369Pd0.2Sn0.799I2.942Br0.058	1.3401
FA0.285MA0.334Cs0.382Sn0.739Cr0.262I2.911Br0.089	1.3398
MA0.504FA0.077Cs0.419Sn0.704Cr0.295Br0.005I2.995	1.3402
FA0.278MA0.33Cs0.392Sn0.752Cr0.248I2.984Br0.016	1.3402
Cs0.227MA0.367FA0.406Pd0.178Sn0.823I2.923Br0.077	1.3398
MA0.364Cs0.225FA0.411Sn0.999Pd0.001I2.757Br0.243	1.3398
Cs0.136FA0.084MA0.78Sn0.949Ge0.051Br0.51I2.49	1.3402
Cs0.365FA0.461MA0.174Pd0.191Sn0.809I2.964Br0.036	1.3402
FA0.283MA0.322Cs0.395Sn0.77Cr0.23I2.991Br0.009	1.3397
FA0.371MA0.294Cs0.335Sn0.977Pd0.023Br0.473I2.528	1.3397
FA0.068MA0.6Cs0.333Pd0.323Sn0.677I2.775Br0.225	1.3403
MA0.266FA0.517Cs0.217Sn0.979Cr0.022Br0.317I2.683	1.3397
MA0.507FA0.105Cs0.389Sn0.707Cr0.293Br0.077I2.923	1.3403
FA0.398Cs0.397MA0.204Sn0.823Pd0.178Br0.003I2.997	1.3397
MA0.239FA0.492Cs0.269Sn0.992Pd0.007Br0.354I2.646	1.3403
FA0.407Cs0.295MA0.299Sn0.947Ge0.052Br0.43I2.57	1.3397

MA0.179Cs0.527FA0.294Sn0.84Pd0.16I2.909Br0.09	1.3403
Cs0.241MA0.375FA0.384Pd0.203Sn0.797I2.934Br0.066	1.3403
Cs0.239MA0.372FA0.389Pd0.211Sn0.789I2.957Br0.043	1.3404
MA0.203FA0.513Cs0.284Sn0.949Pd0.051Br0.352I2.647	1.3396
MA0.239FA0.384Cs0.377Sn0.693Pd0.308Br0.092I2.908	1.3396
FA0.283MA0.325Cs0.392Sn0.749Cr0.251I2.982Br0.018	1.3404
Cs0.267MA0.337FA0.396Pd0.21Sn0.791I2.905Br0.095	1.3396
MA0.23FA0.373Cs0.397Sn0.687Pd0.314Br0.099I2.901	1.3396
MA0.303Cs0.331FA0.367Sn0.995Sr0.005I2.668Br0.332	1.3404
FA0.357MA0.305Cs0.337Sn0.633Pd0.367Br0.101I2.899	1.3396
FA0.123Cs0.418MA0.459Sn0.775Cr0.226I2.96Br0.04	1.3405
FA0.328Cs0.52MA0.152Pd0.209Sn0.792Br0.085I2.915	1.3395
MA0.194FA0.402Cs0.405Sn0.82Pd0.181Br0.054I2.946	1.3405
Cs0.28FA0.536MA0.185Ge0.049Sn0.952Br0.847I2.154	1.3395
FA0.428Cs0.017MA0.555Sn0.978Pd0.022Br0.298I2.702	1.3395
Cs0.073MA0.63FA0.297Pd0.052Sn0.948Cl0.123I2.73Br0.147	1.3405
MA0.187Cs0.349FA0.463Pd0.225Sn0.774I2.938Br0.061	1.3405
MA0.213FA0.535Cs0.252Sn0.961Pd0.039Br0.475I2.524	1.3405
Cs0.263MA0.567FA0.17Sn0.927Cr0.074Br0.031I2.969	1.3406
FA0.453MA0.116Cs0.431Pd0.161Sn0.839Br0.072I2.928	1.3406
MA0.235FA0.465Cs0.3Sn0.93Pd0.07Br0.302I2.698	1.3406
Cs0.174MA0.703FA0.123Sn0.803Pd0.196I2.998Br0.002	1.3394
MA0.213FA0.394Cs0.393Sn0.733Pd0.267Br0.008I2.992	1.3394
MA0.216FA0.392Cs0.392Sn0.733Pd0.267Br0.008I2.992	1.3394
Cs0.28MA0.547FA0.173Sn0.844Pd0.156I2.973Br0.027	1.3406
Cs0.122MA0.757FA0.121Sn0.952Ge0.048I2.353Br0.646	1.3394
MA0.173FA0.402Cs0.425Sn0.774Pd0.225Br0.001I2.999	1.3406
MA0.465FA0.078Cs0.457Sn0.766Cr0.234Br0.001I2.999	1.3406
Cs0.24MA0.391FA0.369Pd0.14Sn0.86I2.924Br0.076	1.3394
Cs0.248MA0.381FA0.371Pd0.22Sn0.78I2.959Br0.041	1.3394
Cs0.086MA0.733FA0.181Sn0.929Ge0.071I2.418Br0.582	1.3406
FA0.276MA0.325Cs0.399Sn0.742Cr0.258I2.984Br0.016	1.3394
MA0.323FA0.313Cs0.364Cr0.301Sn0.699Br0.001I2.998	1.3407
Cs0.251MA0.379FA0.37Pd0.219Sn0.781I2.921Br0.079	1.3393
Cs0.073MA0.692FA0.235Sn0.951Ge0.048I2.414Br0.586	1.3407
FA0.259MA0.481Cs0.26Ge0.105Sn0.895I2.57Br0.43	1.3393
Cs0.264MA0.375FA0.361Pd0.118Sn0.882I2.999Br0.001	1.3393
Cs0.236MA0.394FA0.37Pd0.138Sn0.862I2.938Br0.062	1.3393
Cs0.233MA0.381FA0.386Pd0.135Sn0.866I2.956Br0.044	1.3393
Cs0.235MA0.407FA0.358Pd0.122Sn0.878I2.938Br0.062	1.3393
Cs0.481MA0.215FA0.304Sn0.687Pd0.313I2.999Br0.001	1.3408
MA0.196FA0.386Cs0.418Sn0.764Pd0.236Br0.041I2.959	1.3392
Cs0.257MA0.381FA0.362Pd0.259Sn0.741I2.918Br0.082	1.3408
Cs0.347FA0.216MA0.437Sn0.823Cr0.177I2.956Br0.044	1.3408

FA0.06Cs0.544MA0.396Sn0.954Ge0.046Br0.555I2.445	1.3408
Cs0.188MA0.66FA0.152Sn0.86Pd0.139I2.995Br0.005	1.3408
MA0.335FA0.212Cs0.453Cr0.199Sn0.802Br0.036I2.964	1.3408
MA0.223FA0.601Cs0.176Sn0.887Ge0.113Br0.307I2.693	1.3392
Cs0.055MA0.94FA0.004Sn0.956Ge0.044I2.35Br0.651	1.3392
MA0.323FA0.443Cs0.234Sn0.811Cr0.19Br0.006I2.994	1.3392
Cs0.168MA0.699FA0.133Sn0.892Pd0.108I2.997Br0.003	1.3391
Cs0.081FA0.025MA0.894Sn0.939Ge0.061Br0.644I2.356	1.3409
Cs0.032MA0.578FA0.389Cr0.002Sn0.999I2.804Br0.196	1.3409
Cs0.247MA0.385FA0.369Pd0.156Sn0.844I2.946Br0.054	1.3409
FA0.382Cs0.338MA0.28Pd0.061Sn0.939Br0.653I2.347	1.3409
FA0.156MA0.6Cs0.244Sn0.903Ge0.097Br0.42I2.579	1.3409
FA0.343MA0.225Cs0.432Sn0.772Ge0.228I2.892Br0.108	1.3409
Cs0.251FA0.425MA0.324Sn0.848Ge0.152I2.868Br0.132	1.3409
MA0.368FA0.418Cs0.214Sn0.944Pd0.056Br0.289I2.711	1.3390
Cs0.095MA0.744FA0.162Sn0.913Ge0.087I2.496Br0.504	1.3390
Cs0.177MA0.73FA0.094Sn0.873Pd0.127I2.965Br0.035	1.3410
MA0.195FA0.398Cs0.407Sn0.803Pd0.198Br0.055I2.945	1.3410
FA0.209MA0.5Cs0.292Sn0.749Cr0.252Br0.018I2.982	1.3410
FA0.413MA0.254Cs0.333Sn0.997Sr0.002I2.633Br0.367	1.3410
MA0.201FA0.408Cs0.391Sn0.832Pd0.168Br0.083I2.917	1.3410

Table S10 Searched candidates for 1.67 eV.

Triple divalent cations HOIP candidate	Predicted bandgap
FA0.016Cs0.392MA0.592Cr0.383Sr0.347Sn0.27Br1.171I1.829	1.6702
Cs0.445FA0.161MA0.394Pd0.508Cr0.228Sn0.263Br1.094I1.906	1.6693
FA0.057Cs0.789MA0.153Pd0.363Sn0.144Ca0.493Br0.88I2.12	1.671
FA0.355Cs0.322MA0.323Ca0.159Pd0.354Sn0.487I2.014Br0.986	1.6714
FA0.386MA0.221Cs0.393Sr0.333Sn0.351Pd0.315Br1.337I1.663	1.6715
MA0.25Cs0.538FA0.212Sn0.198Sr0.341Pd0.461I.823Br1.177	1.6685
FA0.381Cs0.582MA0.037Pd0.497Sn0.319Sr0.184I2.046Br0.954	1.6685
FA0.017Cs0.436MA0.547Cr0.063Sr0.168Sn0.769I1.595Br1.405	1.6683
Cs0.437MA0.007FA0.557Cr0.361Sn0.4Ge0.239I2.175Br0.825	1.668
Cs0.424MA0.392FA0.184Ge0.162Sr0.626Ca0.212Br0.921I2.079	1.6673
Cs0.296MA0.136FA0.567Ca0.353Sr0.466Sn0.181Br1.297I1.703	1.6672
FA0.421Cs0.165MA0.414Sn0.137Sr0.465Ca0.398I1.708Br1.292	1.6671
Cs0.292FA0.159MA0.549Pd0.215Ca0.514Sn0.271I2.257Br0.743	1.6666
FA0.315Cs0.388MA0.297Ca0.222Pd0.363Sn0.415I1.896Br1.104	1.6734
Cs0.311MA0.406FA0.284Sr0.563Sn0.358Pd0.079I1.418Br1.582	1.6665
Cs0.652MA0.084FA0.264Sr0.638Ge0.151Cr0.211I2.336Br0.664	1.6664
FA0.162MA0.342Cs0.497Cr0.36Ge0.28Sn0.361I.721Br1.279	1.6658
Cs0.853MA0.049FA0.098Sr0.677Sn0.063Cr0.26I2.365Br0.635	1.6657
MA0.501Cs0.192FA0.307Sn0.315Pd0.299Sr0.387I1.597Br1.403	1.6656
FA0.367Cs0.525MA0.108Sr0.098Cr0.351Sn0.551I1.798Br1.202	1.6655

MA0.27Cs0.518FA0.212Sn0.209Cr0.331Pd0.46I1.746Br1.254	1.6654
FA0.241Cs0.485MA0.274Cr0.095Sr0.218Sn0.687I1.614Br1.386	1.6654
FA0.42Cs0.482MA0.099Sr0.169Cr0.322Sn0.509I1.805Br1.195	1.6653
FA0.078Cs0.495MA0.427Ge0.333Pd0.294Sr0.374I2.907Br0.093	1.6651
MA0.191FA0.229Cs0.58Ge0.179Sn0.467Ca0.355I2.003Br0.997	1.6752
MA0.43Cs0.383FA0.187Ge0.246Sr0.339Ca0.415I2.151Br0.849	1.6756
FA0.184Cs0.589MA0.227Ge0.038Cr0.222Ca0.739I2.443Br0.557	1.6758
MA0.094FA0.478Cs0.428Ca0.312Sn0.399Cr0.289Br0.568I2.432	1.6637
MA0.253FA0.408Cs0.339Ge0.328Pd0.062Sr0.609I2.654Br0.346	1.6634
Cs0.405MA0.115FA0.481Cr0.356Sn0.383Ge0.26I2.758Br0.242	1.6633
Cs0.837MA0.074FA0.089Sr0.608Ge0.23Pd0.162I2.995Br0.005	1.6631
Cs0.245FA0.12MA0.635Sn0.48Pd0.436Sr0.084I2.147Br0.853	1.6629
FA0.316Cs0.511MA0.173Ca0.266Cr0.365Sn0.37I1.964Br1.036	1.6629
MA0.489FA0.116Cs0.395Sr0.89Pd0.002Ge0.108Br1.063I1.937	1.6774
MA0.273Cs0.496FA0.231Sn0.187Sr0.379Pd0.435I1.927Br1.073	1.662
Cs0.795MA0.094FA0.111Sr0.658Pd0.272Ge0.069I2.581Br0.419	1.6619
MA0.366Cs0.494FA0.139Pd0.136Sr0.491Sn0.372I1.454Br1.546	1.6782
Cs0.658MA0.1FA0.242Pd0.325Sr0.273Sn0.402I2.032Br0.968	1.6786
MA0.124FA0.378Cs0.498Ca0.213Sn0.639Sr0.148Br1.457I1.543	1.6787
FA0.057Cs0.658MA0.285Ge0.388Sr0.358Ca0.254I2.417Br0.583	1.6788
MA0.402FA0.241Cs0.356Sr0.24Sn0.263Ca0.497Br1.354I1.646	1.6789
FA0.382Cs0.504MA0.114Sr0.136Cr0.422Sn0.442I1.701Br1.299	1.661
FA0.3Cs0.518MA0.182Sr0.112Cr0.266Sn0.622I1.852Br1.148	1.6609
FA0.003Cs0.798MA0.199Pd0.369Sn0.151Ca0.48Br0.981I2.019	1.6793
FA0.088Cs0.821MA0.091Ge0.376Sn0.44Sr0.184I1.997Br1.003	1.6799

Double divalent cations HOIP candidate	Predicted bandgap (eV)
MA0.126FA0.495Cs0.379Ca0.357Sn0.643Br1.367I1.633	1.6701
MA0.215Cs0.526FA0.259Sr0.603Sn0.396I1.635Br1.365	1.6696
MA0.329FA0.149Cs0.522Sn0.307Sr0.694Br1.338I1.662	1.6694
MA0.17FA0.451Cs0.379Ca0.438Sn0.562Br1.342I1.658	1.6694
MA0.343Cs0.498FA0.159Sn0.543Sr0.458I1.672Br1.328	1.6693
MA0.365Cs0.507FA0.129Sn0.562Sr0.437I1.583Br1.417	1.6686
MA0.201Cs0.547FA0.252Sr0.619Sn0.381I1.62Br1.38	1.6679
MA0.358Cs0.492FA0.15Sn0.514Sr0.486I1.575Br1.425	1.6677
MA0.257Cs0.436FA0.308Pd0.277Sn0.723I1.28Br1.72	1.6732
Cs0.78FA0.017MA0.203Sr0.576Ge0.425Br0.582I2.418	1.6665
Cs0.428MA0.571FA0.001Sr0.847Sn0.153I1.826Br1.174	1.6665
MA0.404FA0.06Cs0.537Sn0.459Cr0.541Br1.759I1.241	1.6663
MA0.355Cs0.614FA0.031Sn0.524Sr0.476I1.734Br1.266	1.6658
MA0.197FA0.384Cs0.42Cr0.255Sn0.745Br1.876I1.124	1.6657
MA0.034Cs0.323FA0.643Sn0.694Cr0.306I1.34Br1.66	1.6752
Cs0.763FA0.021MA0.216Sr0.58Ge0.42Br0.388I2.612	1.6644
Cs0.374MA0.491FA0.135Sn0.212Sr0.788I1.855Br1.145	1.6644

Cs0.715MA0.034FA0.252Sr0.576Ge0.424Br0.644I2.356	1.6762
Cs0.419MA0.228FA0.353Sn0.613Ge0.387I2.139Br0.861	1.6634
FA0.217MA0.21Cs0.572Sn0.651Sr0.349Br1.382I1.618	1.6631
MA0.186Cs0.606FA0.208Ca0.677Ge0.323I2.901Br0.099	1.6776
FA0.311Cs0.629MA0.06Sr0.153Sn0.847I1.672Br1.328	1.6777
Cs0.544MA0.165FA0.291Sn0.712Sr0.288I1.915Br1.085	1.6622
Cs0.433FA0.325MA0.242Sn0.344Cr0.656I2.984Br0.016	1.6779
FA0.177Cs0.375MA0.448Ca0.742Sn0.258Br1.378I1.622	1.6782
MA0.319Cs0.424FA0.257Sn0.662Sr0.339I1.578Br1.422	1.6785
MA0.165Cs0.527FA0.309Sn0.617Sr0.383I1.666Br1.334	1.6787
MA0.161FA0.49Cs0.349Ca0.49Sn0.509Br1.508I1.492	1.6787
Cs0.557FA0.05MA0.393Sr0.545Sn0.455Br1.4I1.6	1.6611
Cs0.973FA0.024MA0.003Sr0.584Ge0.416Br0.354I2.646	1.6607
MA0.371Cs0.624FA0.005Sn0.524Sr0.476I1.602Br1.398	1.6794
MA0.243Cs0.618FA0.138Sn0.368Sr0.632Br1.415I1.585	1.6603
Cs0.925MA0.002FA0.072Sr0.726Ge0.274I2.996Br0.004	1.6602
FA0.28Cs0.483MA0.238Sn0.602Ca0.399I1.5Br1.5	1.6798
MA0.425Cs0.521FA0.054Sn0.51Sr0.49I1.586Br1.414	1.66

Table S11 Searched candidates for 1.2 eV.

Double divalent cations HOIP candidate	Predicted bandgap (eV)
MA0.815FA0.185Ge0.073Sn0.927I3.0	1.2190
FA0.145MA0.856Ge0.074Sn0.925I3.0	1.2191
MA0.86FA0.139Sn0.914Ge0.086I3.0	1.2227
MA0.851FA0.149Sn0.92Ge0.08I3.0	1.2235
MA0.851FA0.148Sn0.984Ge0.017I3.0	1.2263
MA0.859FA0.141Sn0.986Ge0.015I3.0	1.2264
MA0.86FA0.139Sn0.987Ge0.014I3.0	1.2265
MA0.862FA0.138Sn0.988Ge0.013I3.0	1.2266
MA0.858FA0.142Sn0.987Ge0.014I3.0	1.2266
MA0.861FA0.139Sn0.987Ge0.014I3.0	1.2266
MA0.851FA0.149Ge0.012Sn0.989I3.0	1.2266
MA0.855FA0.145Ge0.013Sn0.988I3.0	1.2266
MA0.852FA0.148Ge0.013Sn0.988I3.0	1.2266
MA0.858FA0.142Ge0.012Sn0.989I3.0	1.2266
MA0.861FA0.139Sn0.988Ge0.013I3.0	1.2266
MA0.848FA0.151Sn0.987Ge0.014I3.0	1.2267
MA0.781FA0.22Sn0.976Ge0.024I3.0	1.2276
MA0.78FA0.221Sn0.975Ge0.025I3.0	1.2276
MA0.777FA0.224Ge0.024Sn0.976I3.0	1.2277
MA0.788FA0.213Sn0.975Ge0.025I3.0	1.2279
MA0.782FA0.219Sn0.976Ge0.024I3.0	1.2279
MA0.784FA0.217Sn0.975Ge0.025I3.0	1.2279
MA0.782FA0.219Ge0.025Sn0.975I3.0	1.2279

FA0.515Cs0.486Sn0.952Ge0.047I3.0	1.2293
Cs0.486FA0.515Ge0.051Sn0.948I3.0	1.2293
MA0.823FA0.176Sn0.989Ge0.012I3.0	1.2295
FA0.516Cs0.485Ge0.048Sn0.952I3.0	1.2299

Table S12 Predictions for the candidates from ref⁴¹

Formula	Bandgap (eV)	τ_f	O_f	t_f
ASnI3	1.2864	4.162	0.523	0.871
AGSnI3	1.2894	3.828	0.523	0.939
XQSnI3	1.2994	3.884	0.523	0.923
ATmI3	1.3506	4.124	0.468	0.903
MASmI3	1.3584	3.748	0.555	0.939
AGSmI3	1.3850	3.816	0.555	0.92
XQTmI3	1.3867	3.963	0.468	0.957
XQSmI3	1.3879	3.891	0.555	0.904
ASmI3	1.3918	4.263	0.555	0.853
AGTmI3	1.4048	3.931	0.468	0.974
AYbI3	1.4335	4.128	0.464	0.906
AGEuI3	1.4420	3.821	0.532	0.934
MAEuI3	1.4538	3.765	0.532	0.953
AEuI3	1.4562	4.184	0.532	0.866
XQEuI3	1.4591	3.882	0.532	0.917
XQYbI3	1.4782	3.975	0.464	0.96
AGSrI3	1.5007	3.818	0.536	0.931
ACaI3	1.5067	4.139	0.455	0.912
XQSrI3	1.5217	3.882	0.536	0.914
AScI3	1.5222	4.675	0.339	0.99
AGYbI3	1.5256	3.945	0.464	0.977
ASrI3	1.5269	4.198	0.536	0.863
XQCaI3	1.5521	4.001	0.455	0.966
MATmI3	1.5748	3.904	0.468	0.994
ANbI3	1.6656	4.774	0.327	0.999
MAYbI3	1.6839	3.92	0.464	0.997
XQTiI3	1.7123	4.284	0.391	1.01
AASmI3	1.7285	3.906	0.555	0.901
AGCaI3	1.7294	3.974	0.455	0.983
AMgI3	1.7704	4.774	0.327	0.999
AGTiI3	1.8425	3.999	0.439	1.056
AGScI3	1.8717	4.363	0.38	1.101
MAScI3	1.8972	4.697	0.339	1.09
AASnI3	1.9007	3.895	0.523	0.92
AZrI3	1.9103	5.081	0.301	1.077
ASmF3	1.9188	3.55	0.917	0.903
XQScI3	1.9255	4.678	0.339	1.05

AGZrI3	1.9400	4.103	0.444	1.319
XQZrI3	1.9735	5.553	0.268	1.1
ATmBr3	1.9825	3.891	0.526	0.919
ASnF3	1.9827	3.406	0.865	0.928
AGMgI3	1.9849	4.465	0.367	1.111
AATmI3	1.9909	3.969	0.468	0.954
ASnBr3	1.9916	3.953	0.587	0.883
AGNbI3	2.0059	4.465	0.367	1.111
AGNbI3	2.0059	3.62	0.541	1.293
XQSnBr3	2.0097	3.675	0.587	0.939
AAScI3	2.0180	4.354	0.38	1.076
MANbI3	2.0182	4.813	0.327	1.1
AATiI3	2.0199	4.286	0.391	1.007
AGSnBr3	2.0236	3.619	0.587	0.957
ATiBr3	2.0257	4.057	0.439	0.974
XQMgI3	2.0262	4.788	0.327	1.058
XQNbI3	2.0366	4.788	0.327	1.058
AAEuI3	2.0524	3.894	0.532	0.914
AGTmBr3	2.0542	3.698	0.526	0.996
AASrI3	2.0558	3.895	0.536	0.911
AScF3	2.0576	3.507	0.56	1.109
ASmBr3	2.0610	4.066	0.622	0.864
XQTmBr3	2.0620	3.73	0.526	0.977
XQSmBr3	2.0649	3.694	0.622	0.919
AAZrI3	2.0715	4.888	0.326	1.169
AAybI3	2.0732	3.981	0.464	0.957
XQSrBr3	2.0752	3.679	0.602	0.93
ACaBr3	2.0761	3.899	0.51	0.928
AMgF3	2.0786	3.566	0.541	1.123
XBNbI3	2.0846	4.832	0.327	1.124
AGSmBr3	2.0869	3.619	0.622	0.936
ASrBr3	2.0926	3.994	0.602	0.875
AACaI3	2.1017	4.006	0.455	0.963
AGSrBr3	2.1076	3.615	0.602	0.948
AEuBr3	2.1088	3.979	0.597	0.878
AYbBr3	2.1102	3.893	0.52	0.922
MASmBr3	2.1112	3.551	0.622	0.956
MAEuBr3	2.1166	3.56	0.597	0.972
XQEuBr3	2.1167	3.677	0.597	0.933
AAMgI3	2.1251	4.453	0.367	1.086
AASnF3	2.1285	3.139	0.865	0.994
XQSnF3	2.1317	3.127	0.865	0.998
ASrF3	2.1327	3.46	0.887	0.917
AANbI3	2.1345	4.453	0.367	1.086

AGEuBr3	2.1366	3.616	0.597	0.951
XQSmF3	2.1416	3.178	0.917	0.971
AASmF3	2.1505	3.193	0.917	0.967
FANbI3	2.1541	3.627	0.541	1.305
AATiF3	2.1545	3.274	0.647	1.126
AGScBr3	2.1569	4.162	0.412	1.124
XQCaBr3	2.1576	3.761	0.51	0.987
ANbF3	2.1580	3.566	0.541	1.123
XQTiF3	2.1586	3.273	0.647	1.13
AATmF3	2.1679	3.125	0.774	1.045
AGSnF3	2.1685	3.071	0.865	1.021
AGMgBr3	2.1687	4.257	0.398	1.135
XQMgBr3	2.1695	4.454	0.367	1.09
AMgBr3	2.1711	4.441	0.367	1.025
AScBr3	2.1818	4.353	0.38	1.016
AEuF3	2.1819	3.441	0.88	0.921
AZrBr3	2.1880	4.827	0.326	1.1
MAScBr3	2.1933	4.375	0.38	1.124
AGTiBr3	2.1944	3.825	0.475	1.075
AACaF3	2.1954	3.136	0.752	1.058
AGSmF3	2.1959	3.103	0.917	0.993
MACaF3	2.2007	3.082	0.752	1.114
AASrF3	2.2010	3.158	0.887	0.982
ATmF3	2.2020	3.28	0.774	0.975
AAMgBr3	2.2027	4.244	0.398	1.109
AAScBr3	2.2158	4.153	0.412	1.098
MATmF3	2.2162	3.06	0.774	1.1
AATiBr3	2.2165	4.007	0.439	1.032
XQScBr3	2.2225	4.355	0.38	1.08
MATmBr3	2.2244	3.671	0.526	1.017
AGNbBr3	2.2291	4.257	0.398	1.135
XQTiBr3	2.2298	4.005	0.439	1.036
ATiF3	2.2382	3.324	0.647	1.051
AAZrBr3	2.2384	4.074	0.444	1.284
AGCaBr3	2.2454	3.734	0.51	1.006
AAEuF3	2.2493	3.151	0.88	0.986
MASnF3	2.2507	3.777	0.523	0.958
AAybF3	2.2509	3.128	0.767	1.049
ASnCl3	2.2520	3.823	0.635	0.892
XQYbBr3	2.2573	3.739	0.52	0.98
ACaF3	2.2622	3.269	0.752	0.988
ANbBr3	2.2630	4.441	0.367	1.025
MASmF3	2.2650	3.035	0.917	1.018
AGSrF3	2.2676	3.081	0.887	1.009

AGYbBr3	2.2684	3.71	0.52	0.999
AScCl3	2.2696	4.151	0.412	1.034
AGEuF3	2.2738	3.077	0.88	1.013
AMgCl3	2.2819	4.233	0.398	1.044
AGTiCl3	2.2826	3.267	0.647	1.156
AGCaF3	2.2849	3.104	0.752	1.087
XQNbBr3	2.2864	4.454	0.367	1.09
MANbBr3	2.2901	4.48	0.367	1.135
AGScCl3	2.2929	3.518	0.56	1.22
ATiCl3	2.2944	3.883	0.475	0.989
AZrCl3	2.2949	4.013	0.444	1.199
AGTmF3	2.2959	3.087	0.774	1.073
MAYbF3	2.2998	3.067	0.767	1.104
AATmBr3	2.3010	3.736	0.526	0.974
AGMgCl3	2.3052	3.59	0.541	1.235
XQCaF3	2.3086	3.131	0.752	1.062
XQTmF3	2.3119	3.118	0.774	1.049
XQSrF3	2.3152	3.145	0.887	0.986
XQScCl3	2.3176	4.154	0.412	1.102
AATiCl3	2.3177	3.832	0.475	1.051
AASeCl3	2.3182	3.509	0.56	1.188
XQTiCl3	2.3190	3.831	0.475	1.054
AAMgCl3	2.3227	3.578	0.541	1.203
XQMgCl3	2.3251	4.246	0.398	1.113
MASrF3	2.3307	3.023	0.887	1.034
AACaBr3	2.3384	3.766	0.51	0.983
ATmCl3	2.3401	3.746	0.569	0.93
ASmCl3	2.3483	3.943	0.674	0.872
AANbBr3	2.3501	4.244	0.398	1.109
AGYbF3	2.3598	3.092	0.767	1.077
AYbF3	2.3701	3.275	0.767	0.979
XQSnCl3	2.3706	3.545	0.635	0.951
ANbCl3	2.3730	4.233	0.398	1.044
XQSmCl3	2.3733	3.571	0.674	0.929
XQEuF3	2.3736	3.138	0.88	0.99
MAYbBr3	2.3750	3.684	0.52	1.02
AGNbCl3	2.3753	3.59	0.541	1.235
AGCaCl3	2.3854	3.584	0.552	1.022
XQYbF3	2.3929	3.122	0.767	1.053
AGTmCl3	2.3960	3.553	0.569	1.011
MAEuF3	2.4035	3.022	0.88	1.038
ACaCl3	2.4182	3.749	0.552	0.94
AANbCl3	2.4239	3.578	0.541	1.203
MATmCl3	2.4412	3.526	0.569	1.033

XQNbCl3	2.4470	4.246	0.398	1.113
AGYbCl3	2.4524	3.563	0.564	1.015
AATmCl3	2.4559	3.591	0.569	0.988
AACaCl3	2.4597	3.616	0.552	0.998
XQCaCl3	2.4665	3.611	0.552	1.002
XQTmCl3	2.4732	3.584	0.569	0.991
ASrCl3	2.4733	3.867	0.652	0.883
AAyBr3	2.4961	3.745	0.52	0.977
AAyCl3	2.5073	3.598	0.564	0.991
AyCl3	2.5105	3.746	0.564	0.933
AGSnCl3	2.5318	3.489	0.635	0.97
AEuCl3	2.5444	3.851	0.646	0.886
AASnBr3	2.5577	3.687	0.587	0.936
XQsrCl3	2.5628	3.552	0.652	0.941
AGSmCl3	2.5693	3.496	0.674	0.948
MAYbCl3	2.5957	3.537	0.564	1.037
XQYbCl3	2.6447	3.592	0.564	0.995
AASnCl3	2.6492	3.556	0.635	0.948
XQEuCl3	2.6625	3.549	0.646	0.945
AGSrCl3	2.6785	3.488	0.652	0.96
AASmCl3	2.6810	3.587	0.674	0.926
AASmBr3	2.6977	3.709	0.622	0.915
AASrBr3	2.7253	3.692	0.602	0.927
MASmCl3	2.7719	3.428	0.674	0.969
AASrCl3	2.7744	3.565	0.652	0.938
AGEuCl3	2.7978	3.487	0.646	0.964
AAEuCl3	2.8295	3.561	0.646	0.941
AAEuBr3	2.8484	3.689	0.597	0.93
MAEuCl3	3.0856	3.432	0.646	0.985

Section S3 Experimental Validation

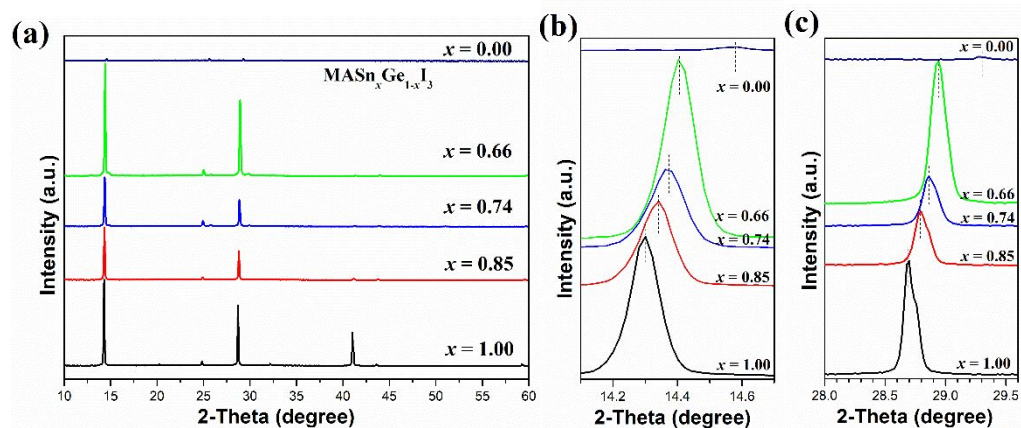


Figure S24 XRD patterns of (a) $\text{MASn}_x\text{Ge}_{1-x}\text{I}_3$ ($x=1, 0.85, 0.74, 0.66, 0$) and partial enlarged plots around (b) 14.30° and (c) 28.70° of 2-Theta.

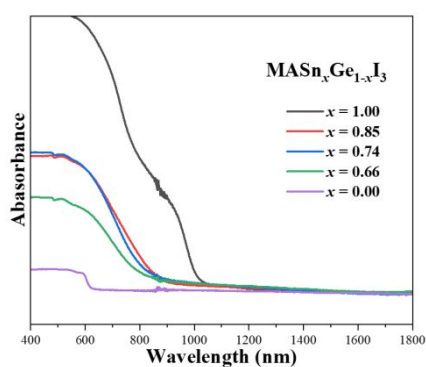


Figure S25 Absorbance spectra of $\text{MASn}_x\text{Ge}_{1-x}\text{I}_3$ ($x=1, 0.85, 0.74, 0.66, 0$).

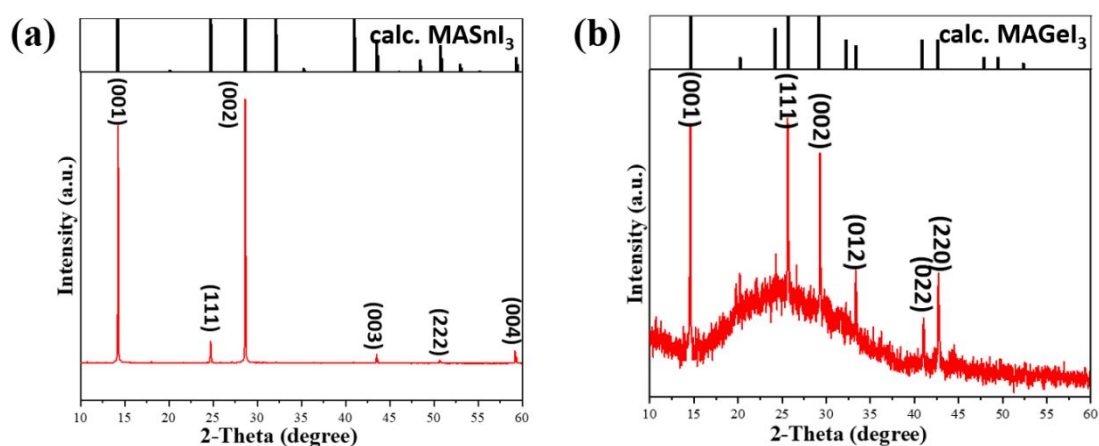


Figure S26 XRD patterns of (a) MASnI_3 and (b) MAGeI_3 . The diffraction peaks of MASnI_3 and MAGeI_3 prepared are consistent with the theoretical values, corresponding to the cubic structure (space group: $\text{Pm}\bar{3}\text{m}$) and tetragonal structure (space group: P4mm), respectively.

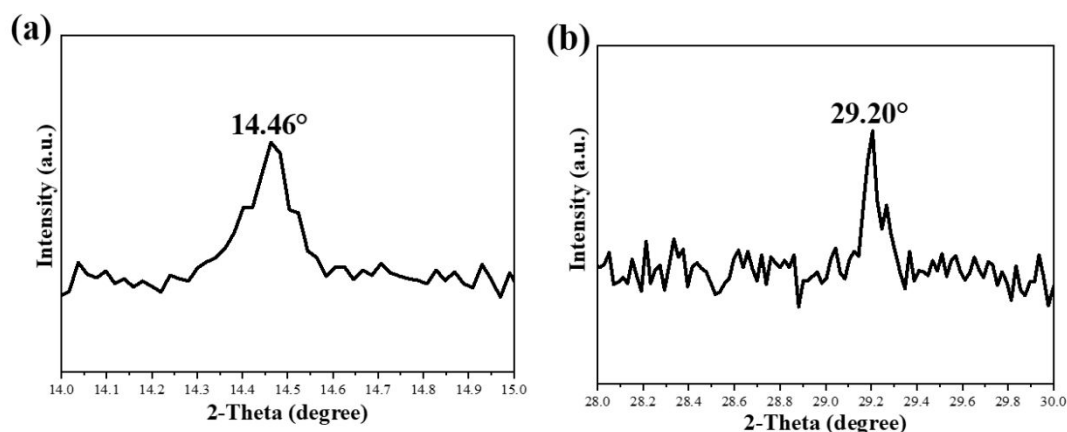


Figure S27 Diffraction peaks of MAGeI_3 at (a) 14.46° and (b) 29.20° of 2-Theta.

Experimental

Methylammonium iodide (MAI, 99.8%) was purchased from GreatCell. Tin (II) iodide (SnI_2 , 99.99%) and tin (II) fluoride (SnF_2 , 99%) were purchased from Sigma-Aldrich. N, N-dimethylformamide (DMF, 99.8%) and dimethylsulfoxide (DMSO, 99.9%) were purchased from Admas-beta. Germanium powder (Ge, 99.999%) and iodine pellets (I_2 , 99.999%) were purchased from Aladdin and Admas-beta, respectively. All the above materials were used as received. GeI_2 was synthesized by sintering a sealed quartz tube containing equimolar amounts of germanium powder and iodine pellets at 600°C for 10 hours. The MASnI_3 precursor solution was prepared by dissolving 15.9 mg MAI, 37.2 mg SnI_2 and 3.13 mg SnF_2 additives in a mixed solvent of 100 μL DMF and 25 μL DMSO while stirring at 60°C for 2 hours. The MAGeI_3 precursor solution was prepared by dissolving 15.9 mg MAI and 32.6 mg GeI_2 in a solvent of 125 μL DMF while stirring at 60°C for 2 hours. The $\text{MASn}_x\text{Ge}_{1-x}\text{I}_3$ ($x=0.85, 0.74, 0.66$) precursor solution was prepared by dissolving 15.9 mg MAI and different molar ratios of SnI_2 and GeI_2 in a mixed solvent of 100 μL DMF and 25 μL DMSO while stirring at 60°C for 2 hours. The MASnI_3 , MAGeI_3 and $\text{MASn}_x\text{Ge}_{1-x}\text{I}_3$ films were deposited by spin-coating the precursor solution onto a glass substrate at 1000-4000 rpm for 45 s in a glove box. All the films were annealed at 100°C for 10 minutes.

Material Characterization:

The crystal structure of perovskite films was examined by a D2-phaser diffractometer from Bruker with a $\text{CuK}\alpha$ radiation. The UV-Vis-IR measurement was performed using a Lambda 750 UV/Vis/NIR spectrophotometer from PerkinElmer. The optical band gap (E_g) of the material was obtained using the following Tauc formula:

$$(\alpha h\nu)^\gamma = A(h\nu - E_g)$$

where α , $h\nu$ and A are absorption coefficient, Planck's constant, the incident photon frequency and proportionality constant, respectively. γ represents the nature of the electronic transition and is equal to 2 for direct allowed transitions condition.

References:

1. Villars, P. Materials Platform for Data Science. <https://mpds.io/> (accessed December).
2. Mentel, Ł. mendeleev -- A Python resource for properties of chemical elements, ions and isotopes. <https://github.com/lmmentel/mendeleev> (accessed December).
3. Frisch, M. J.; Trucks, G. W.; Schlegel, H. B.; Scuseria, G. E.; Robb, M. A.; Cheeseman, J. R.; Scalmani, G.; Barone, V.; Petersson, G. A.; Nakatsuji, H.; Li, X.; Caricato, M.; Marenich, A. V.; Bloino, J.; Janesko, B. G.; Gomperts, R.; Mennucci, B.; Hratchian, H. P.; Ortiz, J. V.; Izmaylov, A. F.; Sonnenberg, J. L.; Williams; Ding, F.; Lipparini, F.; Egidi, F.; Goings, J.; Peng, B.; Petrone, A.; Henderson, T.; Ranasinghe, D.; Zakrzewski, V. G.; Gao, J.; Rega, N.; Zheng, G.; Liang, W.; Hada, M.; Ehara, M.; Toyota, K.; Fukuda, R.; Hasegawa, J.; Ishida, M.; Nakajima, T.; Honda, Y.; Kitao, O.; Nakai, H.; Vreven, T.; Throssell, K.; Montgomery Jr., J. A.; Peralta, J. E.; Ogliaro, F.; Bearpark, M. J.; Heyd, J. J.; Brothers, E. N.; Kudin, K. N.; Staroverov, V. N.; Keith, T. A.; Kobayashi, R.; Normand, J.; Raghavachari, K.; Rendell, A. P.; Burant, J. C.; Iyengar, S. S.; Tomasi, J.; Cossi, M.; Millam, J. M.; Klene, M.; Adamo, C.; Cammi, R.; Ochterski, J. W.; Martin, R. L.; Morokuma, K.; Farkas, O.; Foresman, J. B.; Fox, D. J. *Gaussian 16 Rev. C.01*, Gaussian, Inc.: Wallingford, CT, 2016.
4. Lu, T.; Chen, F., Multiwfn: A multifunctional wavefunction analyzer. *J. Comput. Chem.* **2012**, *33* (5), 580-592.
5. Sánchez-Coronilla, A.; Navas, J.; Gallardo, J. J.; Martín, E. I.; De los Santos, D.; Hernández, N. C.; Alcántara, R.; Toledo, J. H.; Fernández-Lorenzo, C., Hybrid Perovskite, CH₃NH₃PbI₃, for Solar Applications: An Experimental and Theoretical Analysis of Substitution in A and B Sites. *J. Nanomater.* **2017**, *2017*, 9768918.
6. Li, Z.; Najeeb, M. A.; Alves, L.; Sherman, A. Z.; Shekar, V.; Cruz Parrilla, P.; Pendleton, I. M.; Wang, W.; Nega, P. W.; Zeller, M.; Schrier, J.; Norquist, A. J.; Chan, E. M., Robot-Accelerated Perovskite Investigation and Discovery. *Chem. Mater.* **2020**, *32* (13), 5650-5663.
7. Lu, S.; Zhou, Q.; Ouyang, Y.; Guo, Y.; Li, Q.; Wang, J., Accelerated discovery of stable lead-free hybrid organic-inorganic perovskites via machine learning. *Nat. Commun.* **2018**, *9* (1), 3405.
8. Ramli, A.; Bakar, M. N. A.; Osman, N.; Ismail, W. I. N. W.; Sepeai, S., Characterization of novel nitrogen-less derived 2D hybrid perovskite of C₆H₈N₂PbBr₃ as a light-harvesting material for perovskite solar cell application. *Mater. Lett.* **2018**, *227*, 62-65.
9. Laref, A.; Al-Enazi, M.; Al-Qahtani, H. R.; Laref, S.; Wu, X., Impact of fluorine on organic cation for determining the electronic and optical properties of CH_{3-x}F_xNH₃PbI₃ (x = 0, 1, 2, 3) hybrid perovskites-based photovoltaic devices. *Sol. Energy* **2019**, *177*, 517-530.
10. Tsai, C.-M.; Lin, Y.-P.; Pola, M. K.; Narra, S.; Jokar, E.; Yang, Y.-W.; Diao, E. W.-G., Control of Crystal Structures and Optical Properties with Hybrid Formamidinium and 2-Hydroxyethylammonium Cations for Mesoscopic Carbon-Electrode Tin-Based Perovskite Solar Cells. *ACS Energy Lett.* **2018**, *3* (9), 2077-2085.
11. Kevorkyants, R.; Sboev, M. N.; Chizhov, Y. V., Hybrid organic-inorganic lead and tin halide perovskites with saturated heterocyclic cations (CH₂)_nNH₂⁺ and (CH₂)_nOH⁺, (n=2–6): Ab initio study. *Comput. Mater. Sci.* **2017**, *138*, 99-104.
12. Kevorkyants, R.; Emeline, A. V.; Bahnemann, D. W., Novel hybrid semiconducting lead and tin halide perovskites with saturated heterocyclic cations (CH₂)_nPH₂⁺ and (CH₂)_nSH⁺, (n=2–6): Ab initio study. *Mater. Chem. Phys.* **2019**, *229*, 387-391.

13. Yang, Q.; Wu, M.; Zeng, X. C., Constructing Stable and Potentially High-Performance Hybrid Organic-Inorganic Perovskites with “Unstable” Cations. *Research* **2020**, *2020*, 1986576.
14. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E., Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825--2830.
15. Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T. E., Array programming with NumPy. *Nature* **2020**, *585* (7825), 357-362.
16. Komer, B.; Socastro, M. T.; Kim, W. *Hyperopt: Distributed Hyperparameter Optimization*, 0.2.5; 2012-2021.
17. Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S.-I., From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2020**, *2* (1), 56-67.
18. (a) Ding, C.; Peng, H., MINIMUM REDUNDANCY FEATURE SELECTION FROM MICROARRAY GENE EXPRESSION DATA. *J. Bioinf. Comput. Biol.* **2005**, *03* (02), 185-205; (b) Hanchuan, P.; Fuhui, L.; Ding, C., Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE TPAMI* **2005**, *27* (8), 1226-1238.
19. Dorogush, A. V.; Ershov, V.; Gulin, A. CatBoost: gradient boosting with categorical features support. <https://catboost.ai/docs> (accessed December).
20. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. <https://xgboost.readthedocs.io/en/latest/install.html> (accessed December).
21. Guolin, K.; Qi, M.; Thomas, F.; Taifeng, W.; Wei, C.; Weidong, M.; Qiwei, Y.; Tieyan, L., LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, CA, USA, 2017.
22. Friedman, J. H., Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* **2001**, *29* (5), 1189-1232.
23. Chang, C.; Lin, C., LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2* (3), Article 27.
24. Wu, X.; Kumar, V.; Ross Quinlan, J.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G. J.; Ng, A.; Liu, B.; Yu, P. S.; Zhou, Z.; Steinbach, M.; Hand, D. J.; Steinberg, D., Top 10 algorithms in data mining. *KAIS* **2008**, *14* (1), 1-37.
25. Goldstein, A.; Kapelner, A.; Bleich, J.; Pitkin, E., Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation. *arXiv* **2013**, arXiv:1309.6392.
26. Jerome, H. F., Greedy function approximation: A gradient boosting machine. *Ann. Stat* **2001**, *29* (5), 1189-1232.
27. (a) Kumawat, N. K.; Dey, A.; Kumar, A.; Gopinathan, S. P.; Narasimhan, K. L.; Kabra, D., Band Gap Tuning of CH₃NH₃Pb(Br_{1-x}Cl_x)₃ Hybrid Perovskite for Blue Electroluminescence. *ACS Appl. Mater. Interfaces* **2015**, *7* (24), 13119-13124; (b) Kumawat, N. K.; Tripathi, M. N.; Waghmare, U.; Kabra, D., Structural, Optical, and Electronic Properties of Wide Bandgap Perovskites: Experimental and Theoretical Investigations. *J. Phys. Chem. A* **2016**, *120* (22), 3917-3923; (c) Aharon, S.; Cohen, B. E.; Etgar, L., Hybrid Lead Halide Iodide and Lead Halide Bromide in Efficient Hole Conductor Free

Perovskite Solar Cell. *J. Phys. Chem. C* **2014**, *118* (30), 17160-17165; (d) Tu, Y.; Wu, J.; Lan, Z.; He, X.; Dong, J.; Jia, J.; Guo, P.; Lin, J.; Huang, M.; Huang, Y., Modulated CH₃NH₃PbI₃-xBr_x film for efficient perovskite solar cells exceeding 18%. *Sci. Rep.* **2017**, *7* (1), 44603; (e) Zhang, Z. L.; Men, B. Q.; Liu, Y. F.; Gao, H. P.; Mao, Y. L., Effects of precursor solution composition on the performance and I-V hysteresis of perovskite solar cells based on CH₃NH₃PbI₃-xCl_x. *Nanoscale Res. Lett.* **2017**, *12* (1), 84.

28. Zunger, A., Pseudopotential and all-electron atomic core size scales. *J. Chem. Phys.* **1981**, *74* (7), 4209-4211.

29. Allred, A. L.; Rochow, E. G., A scale of electronegativity based on electrostatic force. *J. Inorg. Nucl. Chem.* **1958**, *5* (4), 264-268.

30. Bartel, C. J.; Sutton, C.; Goldsmith, B. R.; Ouyang, R.; Musgrave, C. B.; Ghiringhelli, L. M.; Scheffler, M., New tolerance factor to predict the stability of perovskite oxides and halides. *Sci. Adv.* **2019**, *5* (2), eaav0693.

31. Kim, H.-S.; Lee, C.-R.; Im, J.-H.; Lee, K.-B.; Moehl, T.; Marchioro, A.; Moon, S.-J.; Humphry-Baker, R.; Yum, J.-H.; Moser, J. E.; Grätzel, M.; Park, N.-G., Lead Iodide Perovskite Sensitized All-Solid-State Submicron Thin Film Mesoscopic Solar Cell with Efficiency Exceeding 9%. *Sci. Rep.* **2012**, *2* (1), 591.

32. Hanusch, F. C.; Wiesenmayer, E.; Mankel, E.; Binek, A.; Angloher, P.; Fraunhofer, C.; Giesbrecht, N.; Feckl, J. M.; Jaegermann, W.; Johrendt, D.; Bein, T.; Docampo, P., Efficient Planar Heterojunction Perovskite Solar Cells Based on Formamidinium Lead Bromide. *J. Phys. Chem. L* **2014**, *5* (16), 2791-2795.

33. Jokar, E.; Chien, C.-H.; Tsai, C.-M.; Fathi, A.; Diau, E. W.-G., Robust Tin-Based Perovskite Solar Cells with Hybrid Organic Cations to Attain Efficiency Approaching 10%. *Adv. Mater.* **2019**, *31* (2), 1804835.

34. Chen, Z.; Zheng, X.; Yao, F.; Ma, J.; Tao, C.; Fang, G., Methylammonium, formamidinium and ethylenediamine mixed triple-cation perovskite solar cells with high efficiency and remarkable stability. *J. Mater. Chem. A* **2018**, *6* (36), 17625-17632.

35. Rebecca, L. W. X.; Burhanudin, Z. A.; Abdullah, M.; Saheed, M. S. M., Structural changes and band gap tunability with incorporation of n-butylammonium iodide in perovskite thin film. *Heliyon* **2020**, *6* (2), e03364.

36. Mittal, M.; Jana, A.; Sarkar, S.; Mahadevan, P.; Sapra, S., Size of the Organic Cation Tunes the Band Gap of Colloidal Organolead Bromide Perovskite Nanocrystals. *J. Phys. Chem. L* **2016**, *7* (16), 3270-3277.

37. Weber, S.; Rath, T.; Kunert, B.; Resel, R.; Dimopoulos, T.; Trimmel, G., Dependence of material properties and photovoltaic performance of triple cation tin perovskites on the iodide to bromide ratio. *abrégeé en Monatssh. Chem.* **2019**, *150* (11), 1921-1927.

38. Xu, A. F.; Wang, R. T.; Yang, L. W.; Jarvis, V.; Britten, J. F.; Xu, G., Pyrrolidinium lead iodide from crystallography: a new perovskite with low bandgap and good water resistance. *Chem. Commun.* **2019**, *55* (22), 3251-3253.

39. Kesava, S. V.; Hassan, Y.; Privitera, A.; Varambhia, A.; Snaith, H. J.; Riede, M. K., Azetidinium as cation in lead mixed halide perovskite nanocrystals of optoelectronic quality. *AIP Adv.* **2020**, *10* (2), 025001.

40. Huang, Z.; Chen, B.; Sagar, L. K.; Hou, Y.; Proppe, A.; Kung, H.-T.; Yuan, F.; Johnston, A.;

Saidaminov, M. I.; Jung, E. H.; Lu, Z.-H.; Kelley, S. O.; Sargent, E. H., Stable, Bromine-Free, Tetragonal Perovskites with 1.7 eV Bandgaps via A-Site Cation Substitution. *ACS Mater. Lett.* **2020**, *2* (7), 869-872.

41. Zhang, S.; Lu, T.; Xu, P.; Tao, Q.; Li, M.; Lu, W., Predicting the Formability of Hybrid Organic–Inorganic Perovskites via an Interpretable Machine Learning Strategy. *J. Phys. Chem. L* **2021**, *12* (31), 7423-7430.