New Interpretable Deep Learning Model to Monitor Real-Time $PM_{2.5}$ Concentrations from Satellite Data

Xing Yan[1], Zhou Zang[1], Nana Luo [1, 2], Yize Jiang [1], Zhanqing Li [3*]

1. Additional detail for EntityDenseNet

1.1 Scattering angle Θ_t (Rahman and Dedieu, 1994):

$$\Theta_t = \cos^{-1}(-\cos(\theta_0)\cos(\theta) + \sin(\theta_0)\sin(\theta)\cos(\phi - \phi_0))$$

where θ_0 is the solar zenith angle; θ is the satellite zenith angle, $\phi - \phi_0$ is the relative azimuth angle between the viewing ϕ and solar direction ϕ_0 angles.

1.2 Normalized Vegetation Index(NDVI):

$$\mathrm{NDVI} = \frac{\rho_{0.86} - \rho_{2.26}}{\rho_{0.86} + \rho_{2.26}}$$

Where $\rho_{0.86}$ and $\rho_{2.26}$ are the Himawari-8 measured reflectances in the 0.86 μm and 2.26 μm bands.

1.3 The ReLU activation function:

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x < 0 \end{cases}$$

1.4 Z-score method normalization:

$$Normalized(x_i) = \frac{x_i - mean(x)}{std(x)}$$

$$std(x) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}[x_i - mean(x)]^2}$$

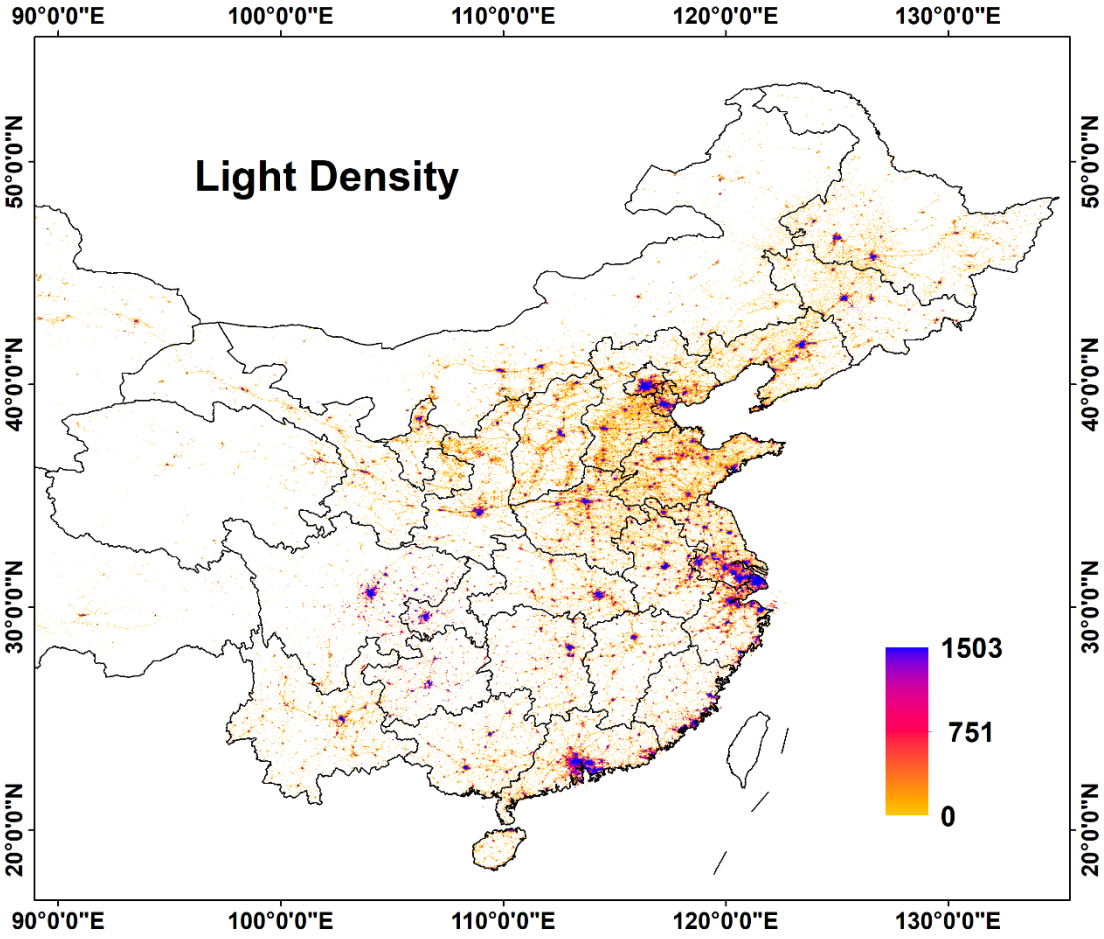


Figure S1. Light density data from Version 1 Nighttime VIIRS Day/Night

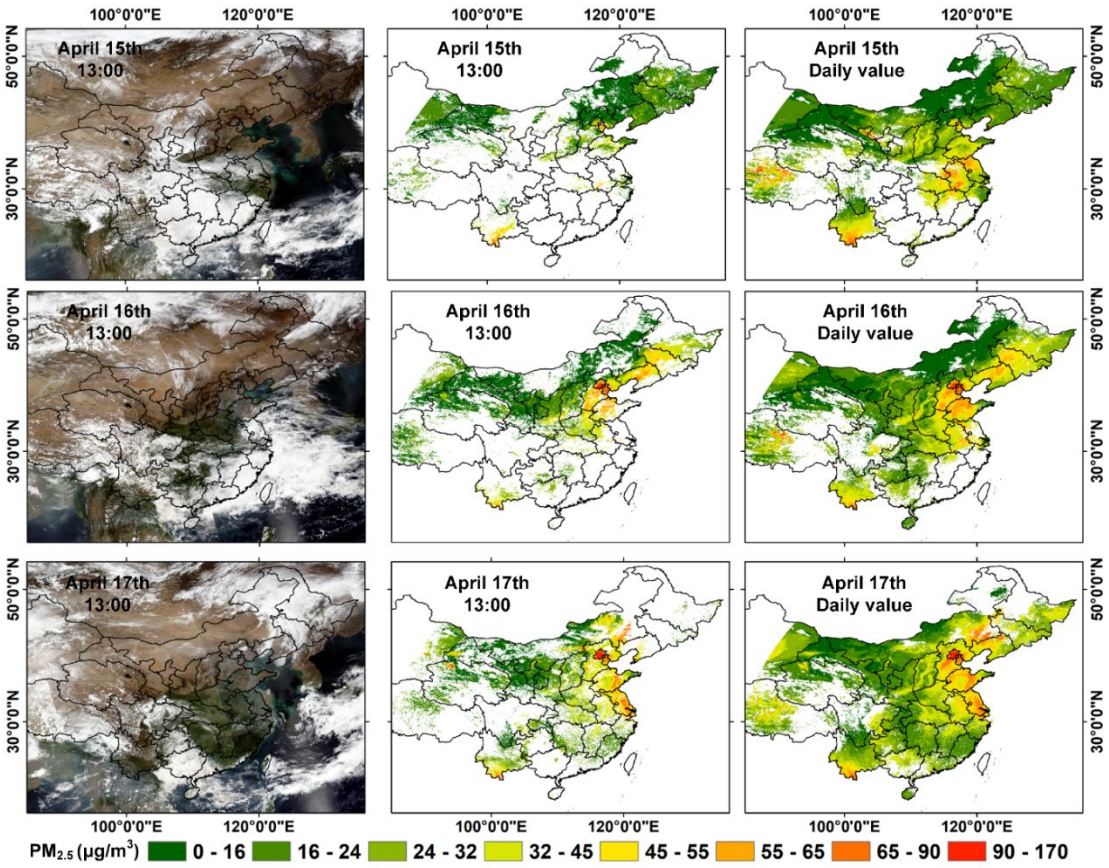


Figure S2. EntityDenseNet $PM_{2.5}$ over mainland China on April 15 –17, 2019. The left column is the true color satellite image. The middle column is the $PM_{2.5}$ concentrations at 13:00 (local time). The right column is the daily averaged $PM_{2.5}$ concentrations.

Table S1. The provincial RMSE for 5 machine learning models.

Province	EntityDenseNet	XGBoost	RF	LightGBM	BPNN
Zhejiang	21.35	22.11	25.18	22.02	27.53
Yunnan	14.68	15.63	20.13	17.00	21.19
Xinjiang	21.07	20.92	24.78	22.49	38.60
Xizang	16.81	20.10	27.46	23.68	23.16
Sichuan	18.58	20.63	25.37	22.36	28.80
Shaanxi	23.79	26.20	30.06	27.52	33.82
Shanxi	31.00	35.21	38.85	35.28	43.67
Shandong	32.88	33.78	40.35	34.45	44.94
Qinghai	20.40	24.96	24.48	25.37	26.57
Ningxia	20.62	22.91	28.11	24.09	29.69
NeiMongol	26.76	28.22	31.78	28.80	35.49
Liaoning	30.60	29.49	35.42	30.13	43.08
Jiangxi	18.66	22.95	23.39	23.94	24.72
Jilin	26.45	27.27	30.40	27.01	34.33
Hunan	19.13	21.49	23.87	22.08	24.29
Hubei	22.11	24.93	26.97	25.42	30.35
Heilongjiang	37.79	38.61	42.88	37.94	47.78
Henan	33.95	36.13	42.46	37.43	48.16
Beijing	26.47	28.62	38.37	30.75	47.45
Tianjin	36.56	42.61	49.57	42.60	58.04
Hainan	8.69	10.07	36.53	14.44	18.37
Guizhou	15.54	17.80	21.14	17.79	20.02
Guangxi	20.31	24.21	29.56	25.31	26.59
Gansu	15.70	18.30	24.21	20.19	26.19
Fujian	16.37	16.14	21.01	17.37	22.78
Anhui	24.09	25.27	28.18	25.63	31.13
Shanghai	20.45	22.85	26.81	23.11	30.37
Chongqing	15.23	18.89	24.29	20.12	26.30
Jiangsu	23.70	25.36	29.53	25.83	33.28
Guangdong	16.02	18.48	25.53	19.83	21.27
Hebei	36.73	39.72	47.66	40.30	55.95

Table S2. The provincial R^2 for Individual machine learning model.

Province	EntityDenseNet	XGBoost	RF	LightGBM	BPNN
Zhejiang	0.47	0.45	0.30	0.47	0.17
Yunnan	0.59	0.56	0.26	0.47	0.17
Xinjiang	0.28	0.23	0.05	0.14	0.16
Xizang	0.29	0.32	0.13	0.23	0.06
Sichuan	0.54	0.53	0.34	0.48	0.33
Shaanxi	0.63	0.60	0.44	0.60	0.33
Shanxi	0.64	0.56	0.45	0.58	0.29
Shandong	0.63	0.59	0.45	0.58	0.30
Qinghai	0.43	0.37	0.17	0.33	0.09
Ningxia	0.39	0.34	0.15	0.31	0.14
NeiMongol	0.46	0.38	0.23	0.35	0.14
Liaoning	0.65	0.62	0.45	0.62	0.17
Jiangxi	0.48	0.40	0.23	0.38	0.19
Jilin	0.47	0.44	0.30	0.45	0.14
Hunan	0.41	0.36	0.21	0.34	0.21
Hubei	0.56	0.51	0.36	0.51	0.29
Heilongjiang	0.57	0.49	0.34	0.49	0.14
Henan	0.64	0.58	0.43	0.55	0.25
Beijing	0.70	0.66	0.37	0.64	0.17
Tianjin	0.68	0.56	0.42	0.57	0.20
Hainan	0.28	0.20	0.00	0.10	0.04
Guizhou	0.27	0.24	0.09	0.22	0.07
Guangxi	0.33	0.26	0.02	0.21	0.06
Gansu	0.48	0.43	0.22	0.35	0.26
Fujian	0.21	0.24	0.11	0.16	0.01
Anhui	0.52	0.49	0.33	0.48	0.22
Shanghai	0.58	0.47	0.26	0.45	0.15
Chongqing	0.50	0.40	0.30	0.37	0.27
Jiangsu	0.56	0.49	0.30	0.48	0.17
Guangdong	0.37	0.22	0.04	0.18	0.09
Hebei	0.69	0.62	0.50	0.62	0.29

Reference

Rahman, H. , & Dedieu, G. . (1994). Smac: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum. International Journal of Remote Sensing, 15(1), 123-143.

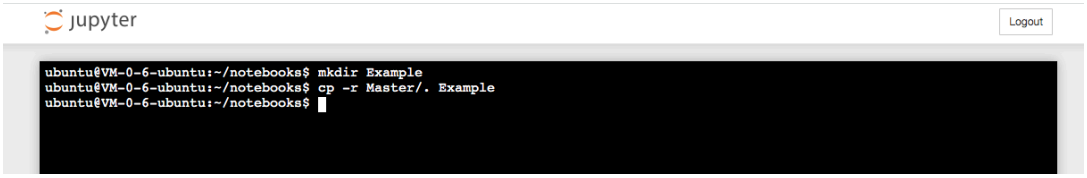
EntityDenseNet Cloud Platform Guide

This Cloud Platform is based on a CPU sever, it allows maximum 2-3 users to run the EntityDenseNet model simultaneously. The website of this Platform is http://49.233.1.40:8888/ . Please contact Dr. Yan (email: yanxing@bnu.edu.cn) to get the access and follow the schedule to use. The EntityDenseNet Cloud Platform can be used to extract spatial and temporal information from the data. Follow our release notes to stay updated on the latest EntityDenseNet releases: https://yanxingemail.wixsite.com/group .

1. When you login to the EntityDenseNet Cloud Platform, you will see:

Create a **new Terminal** and enter the following code in the **Terminal** to create your folder. For example, suppose the name of your folder is called “Example”, enter the code in the Terminal:

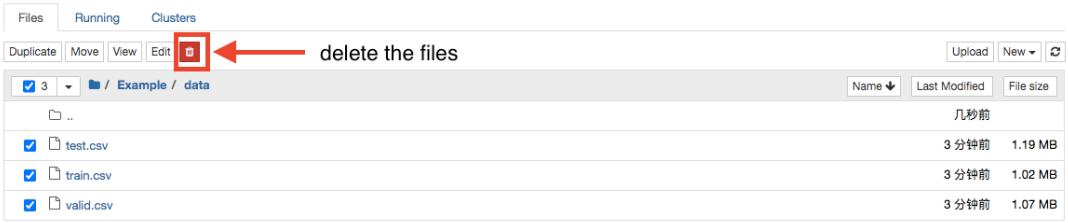
mkdir Example
cp -r Master/. Example



Refresh the website you will see:

Then the "Example" folder is created, and in the folder there are four items generated automatically. The "**data**" is to upload your **training, validating and testing datasets**, the "**output**" is to store the **results from the model training and testing**, and the "**run.sh**" includes all **parameters** of the EntityDenseNet model. After all files are set up, you can run the model on your data by running the "**experiment.ipynb**".

In the "data", there are three example datasets including "test.csv", "train.csv" and "valid.csv", you can easily delete them and upload your own data (or you can directly run the example data to see the result).



Before using the EntityDenseNet model, we should prepare the **training, validation and test datasets**. The **training data** is used to train the model, the **validation data** is used for model hyperparameter optimization or tuning, and the **test data** is used to assess the performance of the trained model. The test data should be "never seen before".

The names of your uploaded **training, validation and test datasets** must be "**train.csv", "valid.csv" and "test.csv"**, and the three datasets should be "**.csv**". The detailed description of the three datasets is as follows:

	A	B	C	D	E	F	G	H	I	J
1	X_1	X_2	X_3	X_4	X_5	X_6	C_1	C_2	C_3	Y
2	0.52	61.3	55	5.22	5.19	3.19	Ideal	E	VS2	1.767
3	0.9	62.8	57	6.07	6.16	3.84	Good	I	VS1	3.398
4	1.02	58	60	6.7	6.64	3.87	Premium	F	SI2	3.996
5	1.09	61.5	57	6.62	6.65	4.08	Ideal	G	VVS2	8.97
6	1.12	61	56	6.76	6.72	4.11	Ideal	G	VS1	7.632
7	1.54	60.3	60	7.47	7.45	4.5	Premium	H	VS2	9.081
8	0.53	63.3	56	5.13	5.17	3.26	Very Good	H	VS1	1.561
9	1.01	63.1	60	6.35	6.39	4.02	Good	D	VS2	6.999
10	0.54	62	57	5.2	5.25	3.24	Very Good	H	VVS1	1.846
11	1	56.4	61	6.58	6.55	3.7	Fair	I	SI1	3.92
12	0.38	60.7	59	4.69	4.73	2.86	Premium	E	VS1	1
13	0.71	59.3	56	5.88	5.82	3.47	Premium	F	VS2	2.84
14	1.04	61.8	55	6.54	6.5	4.03	Ideal	H	SI2	4.368
15	0.37	61.1	57	4.63	4.66	2.84	Very Good	E	VS2	0.811
16	0.91	62.6	55	6.2	6.23	3.89	Ideal	F	VS2	5.261
17	0.43	62.3	54	4.84	4.85	3.02	Ideal	H	VVS1	1.094
18	0.7	59.2	59	5.76	5.79	3.42	Good	D	VVS1	4.198

The **continuous variables in your three datasets should be named as** "X_*", and the **categorical variables in your three datasets should be named as** "C_*". The name of the dependent variable should be "Y". **Null value is not allowed** in the datasets.

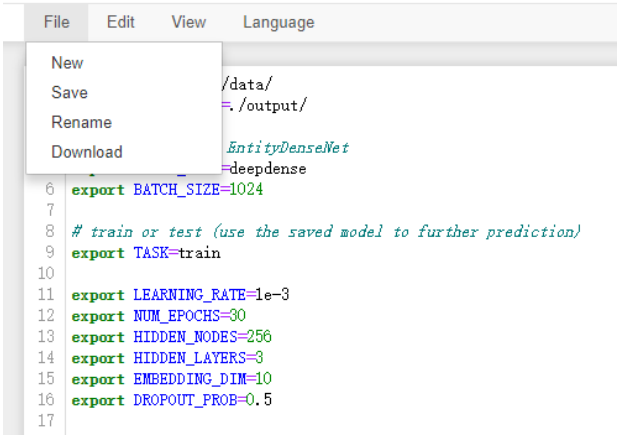
2. Upload your **training, validation and test datasets (in /data)** by the "**upload**" on the upper right of the website.

3. Set up all parameters of the EntityDenseNet model for model training (in **/run.sh)**

```
File  Edit  View  Language

export DATA_DIR=./data/
export OUTPUT_DIR=./output/

# deepdense isthe EntityDenseNet
export MODEL_NAME=deepdense
export BATCH_SIZE=1024

# train or test (use the saved model to further prediction)
export TASK=train

export LEARNING_RATE=1e-3
export NUM_EPOCHS=30
export HIDDEN_NODES=256
export HIDDEN_LAYERS=2
export EMBEDDING_DIM=10
export DROPOUT_PROB=0.5


export NUM_STEPS=5
python ~/github/BNU-Networks/src/run.py --data_dir $DATA_DIR \
--output_dir $OUTPUT_DIR \
--model_name $MODEL_NAME \
--task $TASK \
--learning_rate $LEARNING_RATE \
--num_train_epochs $NUM_EPOCHS \
--batch_size $BATCH_SIZE \
--hidden_nodes $HIDDEN_NODES \
--hidden_layers $HIDDEN_LAYERS \
--dropout_prob $DROPOUT_PROB \
--embedding_dim $EMBEDDING_DIM \
--num_steps $NUM_STEPS
```

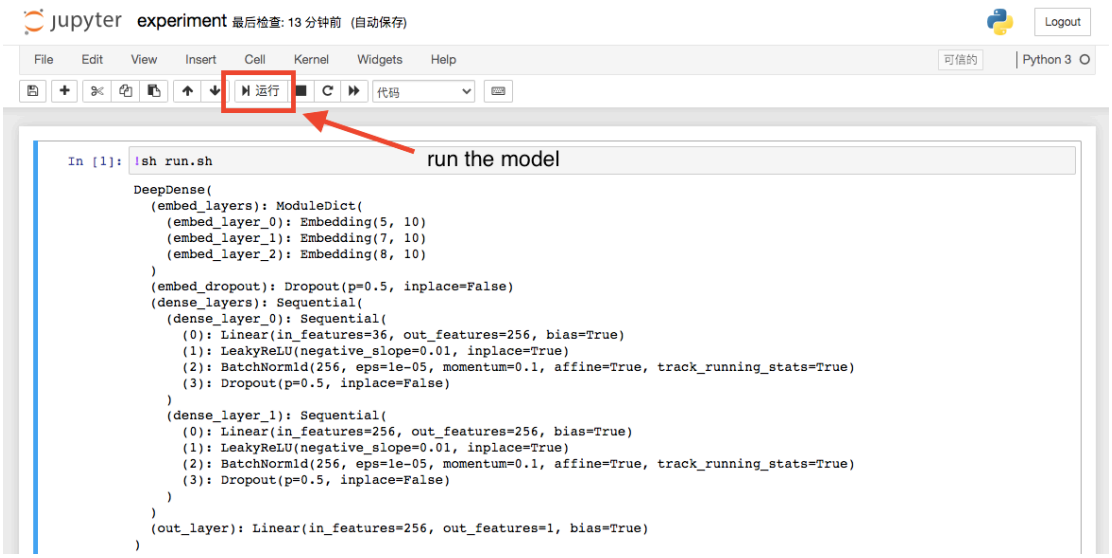
HIDDEN_NODES is the number of neurons in the fully connected layer.
HIDDEN_LAYERS is the number of hidden layer, it should be noted that each hidden layer includes one fully connected layer, one rectified linear unit (ReLU) layer, one batch normalization layer, and one dropout layer.
EMBEEDING_DIM is the D in the Embedding matrix (see paper).
DROPOUT_PROB is the value of the dropout rate in the dropout layer.
LEARNING_RATE is the learning rate.
NUM_EPOCHS is the epochs for model training.

After setting up all parameters, don't forget to save it by the **File section**.



4. Run model (in **/experiment.ipynb)**

Open the experiment.ipynb, and click "run".



```
Epoch 18 -> Train Loss: 8.128909103393555
Epoch 18 -> Valid Loss: 4.186536224365234, RMSE: 2.046102756100886, R2 Score: 0.7449936186473325
Epoch 19 -> Train Loss: 7.071280047098796
Epoch 19 -> Valid Loss: 3.4168413410186766, RMSE: 1.8484698845049101, R2 Score: 0.7918765930966121
Epoch 20 -> Train Loss: 6.238333388010661
Epoch 20 -> Valid Loss: 2.8580145225524904, RMSE: 1.690566353975158, R2 Score: 0.8259152972197636
Epoch 21 -> Train Loss: 5.656378407796224
Epoch 21 -> Valid Loss: 2.355764276504517, RMSE: 1.5348498566763766, R2 Score: 0.8565079091856121
Epoch 22 -> Train Loss: 4.958307954152425
Epoch 22 -> Valid Loss: 1.8117036237716675, RMSE: 1.3459954073678853, R2 Score: 0.8896472084422483
Epoch 23 -> Train Loss: 4.689443752288819
Epoch 23 -> Valid Loss: 1.3841305418014527, RMSE: 1.1764908243053724, R2 Score: 0.915691132321313
Epoch 24 -> Train Loss: 4.346971508026123
Epoch 24 -> Valid Loss: 1.4780635919570924, RMSE: 1.2157564055571284, R2 Score: 0.9099695749266438
Epoch 25 -> Train Loss: 4.0108202463785805
Epoch 25 -> Valid Loss: 1.2124627513885498, RMSE: 1.1011188063535255, R2 Score: 0.9261476144889583
Epoch 26 -> Train Loss: 3.880736406326294
Epoch 26 -> Valid Loss: 1.2507275419235229, RMSE: 1.118359345656196, R2 Score: 0.923816852703279
Epoch 27 -> Train Loss: 3.9868891989390054
Epoch 27 -> Valid Loss: 1.0390084533691406, RMSE: 1.0193175566015071, R2 Score: 0.9367129027276917
Epoch 28 -> Train Loss: 3.605681941350301
Epoch 28 -> Valid Loss: 1.0896088886260986, RMSE: 1.0438432847642227, R2 Score: 0.9336307712252606
Epoch 29 -> Train Loss: 3.6344506969451906
Epoch 29 -> Valid Loss: 0.9871351871490478, RMSE: 0.9935467940896237, R2 Score: 0.9398725452125495
Epoch 30 -> Train Loss: 3.506718782424927
Epoch 30 -> Valid Loss: 1.0135009970664979, RMSE: 1.006727878037666, R2 Score: 0.9382665770565809
Prediction -> Test RMSE: 1.0165454290324079, R2 Score: 0.9357394727355208
```

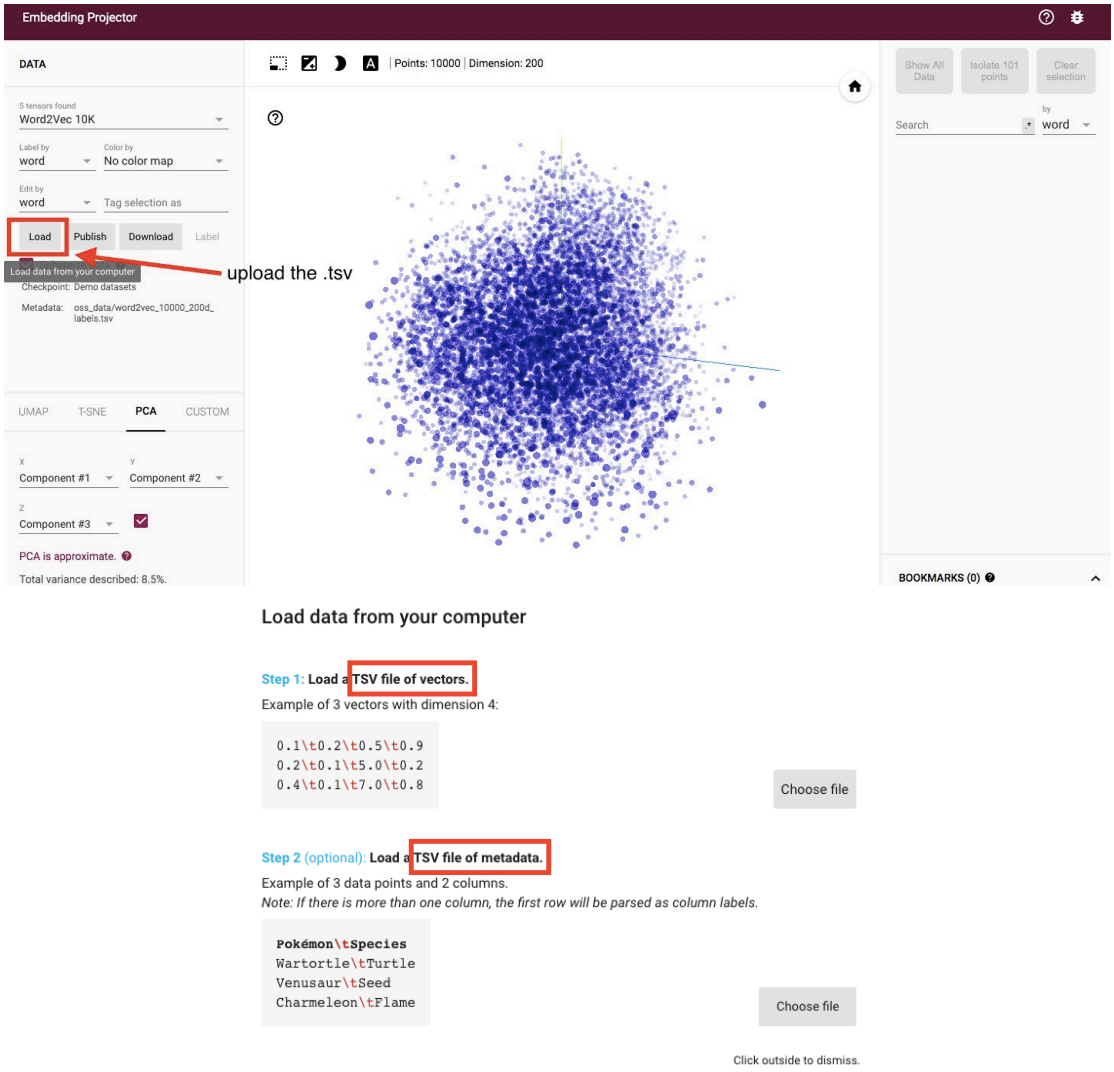
In all outputs, the "**Prediction -> Test RMSE", and "R2 Score"** are the **Root Mean Square Error (RMSE) and Determination Coefficient (R^2)** for the **testing** dataset. The **RMSE and R2** for the **validating** dataset is presented in each epoch.

You can improve the RMSE and R^2 by tuning all parameters in **/run.sh.**

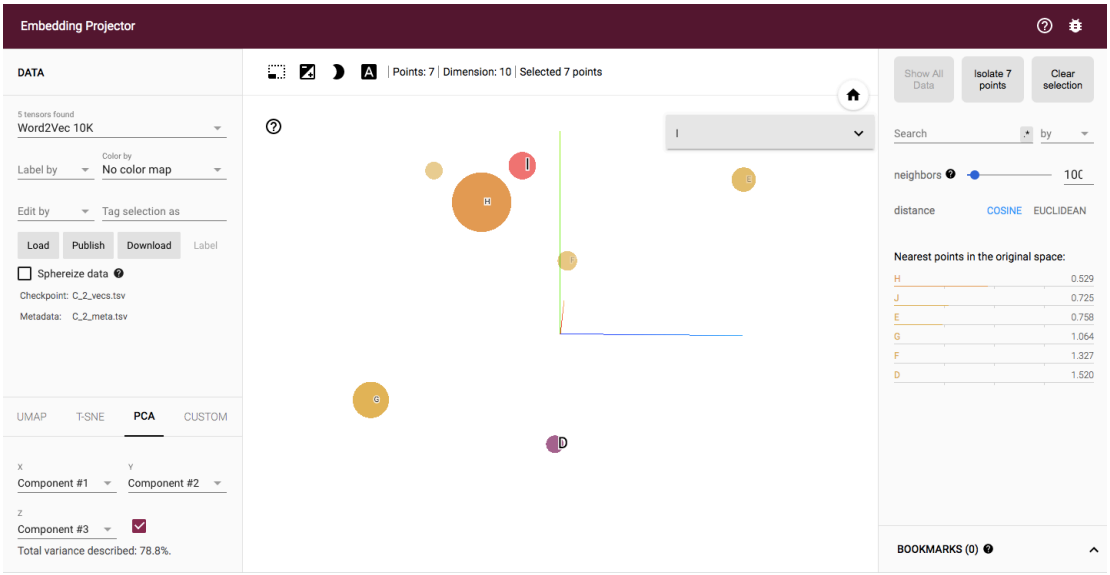
Finally, all results are stored in **/output,** such as the trained model, the prediction for your testing dataset.

5. Interpret the results from the **EntityDenseNet model**

The "C_*_meta.tsv" contains the category information for the categorical variable C_* and the "C_*_vecs.tsv" contains the embedding-based vector for the categorical variable C_*. As for the example data, there are three categorical variables (C_1, C_2, and C_3) in the "Example" data, and the model will output three "C_*_meta.tsv" files and three "C_*_vecs.tsv" files. To visualize the categorical variable C_1's embedding result, TensorFlow, originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization, provides a useful tool. To achieve this aim, we will upload the "C_1_meta.tsv" and "C_1_vecs.tsv" to the website http://projector.tensorflow.org/.



Mapping the matrix of embedding layer from the trained EntityDenseNet down to 3D with UMAP enables us to calculate the Cosine Distance between different variables in this 3D Coordinate System. The smaller the Cosine Distance, the higher the correlation between two features. It reveals the **intrinsic properties of the categorical variables**. This function greatly improves interpretability of the EntityDenseNet inversion result.



The "**history.csv**" is the **validation result** for each epoch and the "**result.csv**" is the **prediction result** based on test set. In "**history.csv**", the **train loss, valid loss, valid RMSE and valid R^2** for every epoch is provided. In "**result.csv**", the **original test dataset** ("X_*", "C_*" and "Y") and **prediction** of Y ("predicted") are provided. The "Y" and "predicted" can be used to see the performance of model by scatter plot.

	X_1	X_2	X_3	X_4	X_5	X_6	C_1	C_2	C_3	Y	predict
2	1.5	62	53	7.3	7.34	4.54	Very Good	I	VVS2	10.873	11.4514484
3	0.4	62.3	54	4.77	4.74	2.96	Ideal	H	VVS1	1.088	0.77828532
4	1.02	60.3	59	6.5	6.53	3.93	Very Good	F	SI2	4.241	4.05704689
5	0.6	62.1	56	5.4	5.45	3.37	Ideal	G	VS2	1.795	1.34303772
6	0.3	61.6	56	4.27	4.3	2.64	Very Good	G	VVS2	0.552	0.99489659
7	0.32	64.1	54	4.37	4.34	2.79	Good	D	SI2	0.571	0.41562802
8	0.37	60.9	57	4.63	4.67	2.83	Ideal	H	SI1	0.552	0.93075627
9	1	62.4	59	6.42	6.37	3.99	Premium	H	SI1	4.816	3.8075676
10	0.24	58.8	58	4.08	4.05	2.39	Premium	D	VVS2	0.668	0.84181851
11	0.94	60.6	54	6.39	6.34	3.86	Ideal	F	SI2	5.159	4.21691465
12	0.41	62.8	56	4.72	4.77	2.98	Very Good	H	SI1	0.683	0.55674481
13	0.41	62.3	57	4.76						56	0.83420622
14	0.36	61.6	55	4.6						23	0.86150181
15	0.3	56.3	62	4.47						64	0.95895749
16	0.9	63.7	57	6.09						92	3.52185464
17	0.32	61.3	60	4.41						89	0.45817333
18	0.41	60.5	61	4.8						69	0.51076698
19	1.53	58.5	59	7.52						88	10.0867958
20	0.31	61.7	56	4.34						01	0.51556325
21	0.54	61.5	56	5.31						94	1.03735507
22	1.51	58.8	59	7.47						44	8.61129856
23	0.94	62.3	59	6.23						15	3.20311689
24	0.74	61.3	56	5.85						13	3.11414266
25	1.21	62.6	56	6.84						24	6.62377977
26	1.22	62.2	56	6.83						14	9.06010914
27	0.7	62.1	55	5.63						71	1.94997871
28	0.42	61.5	56	4.83	4.79	2.96	Ideal	G	SI2	0.682	0.70774907
29	0.4	61.6	59	4.68	4.74	2.9	Very Good	D	SI2	0.666	0.80713201
30	0.81	61.9	57	6.01	5.97	3.71	Ideal	D	SI1	3.353	2.91730618

The "**model.pkl**" is the **trained model** which can be saved for other new test data. When applying this trained model to the new test data (upload the new test data to "data" file), change the "**export TASK=train**" as "**export TASK=test**" in **/run.sh** and run in **experiment.ipynb** directly.