CSE 574: Introduction to Machine Learning, Section C Spring 2022

Assignment 3: Building Neural Networks and CNN

checkpoint on April 3rd

Yvonne Huang (Yon-Chien Huang) #50432184

ACADEMIC INTERGITY STATEMENT

"I certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that I did not receive any external help, coaching or contributions during the production of this work."

Part 1: Building a Basic NN

Nature of dataset

In this assignment, the dataset of income is used. It contains the information of 'age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.status', 'occupation', 'relationship', 'capital.gain', 'capital.loss', 'hours.per.week', 'native.country', 'income'. The datatype is listed as figure 1. The shape of this dataset is (32561, 13), which means there are 423293 information given. While there are 13 columns, we use 12 variables in the dataset to build the NN model to predict the income.

```
age                  int64
workclass           object
fnlwgt             float64
education           object
education.num        int64
marital.status      object
occupation          object
relationship        object
capital.gain         int64
capital.loss         int64
hours.per.week       int64
native.country      object
income              object
```

figure 1 datatype

```
MAX

age                               90
workclass                Without-pay
fnlwgt                     1484705.0
education               Some-college
education.num                     16
marital.status               Widowed
occupation          Transport-moving
relationship                    Wife
capital.gain                   99999
capital.loss                    4356
hours.per.week                    99
native.country            Yugoslavia
income                          >50K
dtype: object
```

figure 2 data maximum

```
min

age                      17
workclass                 ?
fnlwgt              12285.0
education              10th
education.num             1
marital.status     Divorced
occupation                ?
relationship        Husband
capital.gain              0
capital.loss              0
hours.per.week            1
native.country            ?
income                <=50K
dtype: object
```

figure 3 data minimum

```
median
age                   37.0
fnlwgt            178363.0
education.num         10.0
capital.gain           0.0
capital.loss           0.0
hours.per.week        40.0
dtype: float64
```

figure 4 data median

```
mean

age                   38.581647
fnlwgt            189780.114312
education.num         10.080679
capital.gain        1077.648844
capital.loss          87.303830
hours.per.week        40.437456
dtype: float64
```

figure 5 data mean

```
stdev

age                   13.640433
fnlwgt            105551.127393
education.num          2.572720
capital.gain        7385.292085
capital.loss         402.960219
hours.per.week        12.347429
dtype: float64
```

figure 6 data standard deviation

Visualization graph

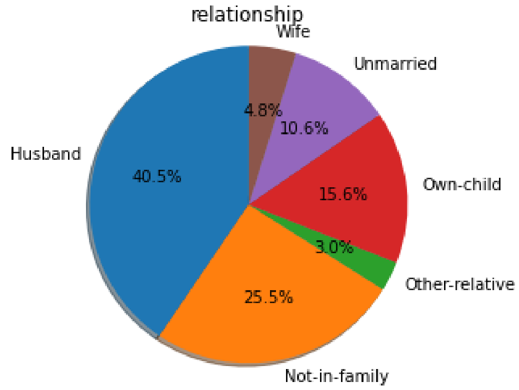


figure 7 relationship graph

This dataset does not include sex information; however, we can acquire some information from the relationship. figure 7 shows there are more than 40% people are husband while there are only less than 5% people are wife. Therefore, the model we built might tend to represent the income situation of male.

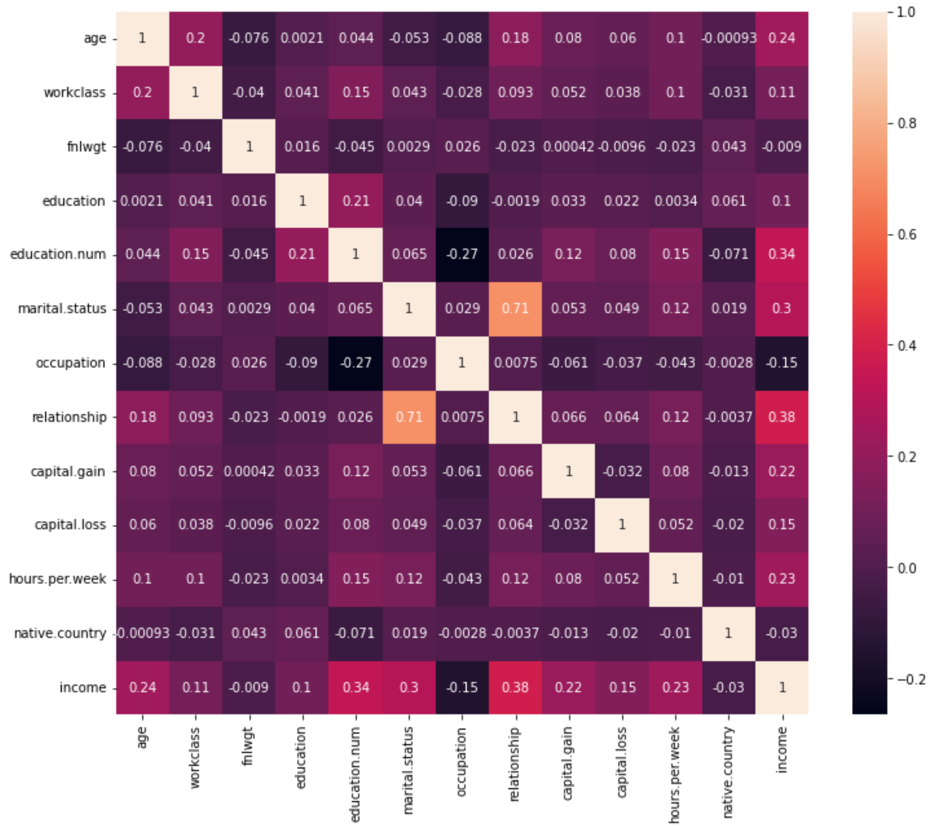


figure 8 heat map

figure 8 represents that most of the variable have low correlation, only relationship and marital.status shows high correlation. It is because the relationship contain the information of marital status, for example, people who are "husband" or "wife" must have the marital status of "marrige".

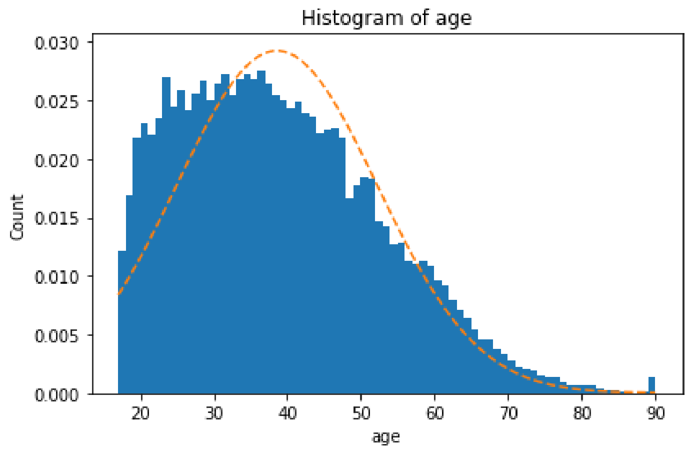


figure 9 histogram of age

figure 9 shows that the data of age are similar to a left-skewed normal distribution, which is the distribution of world population age. Although this data is normally distributed, the amount of people who are 90 years old are more than we expected.

Preprocessing

In preprocessing process, the rows contain "?" or NAN were deleted, and the string categories were transformed into numbers by using pd.factorize function. Since the categorization from tf.keras.layers.Hashing showed some incorrect result, the tools from Keras were not used in this part. After data being categorized, they were normalized to 0-1.

Architecture structure

Hidden layers	Number of nodes	Activation function
#1	60	Sigmoid
#2	10	Sigmoid
output	NAN	Sigmoid

Loss function: tf.keras.losses.BinaryCrossentropy()

Training data: 80%, test data: 20%, epoch: 20, batch size: 10

Loss and Accuracy

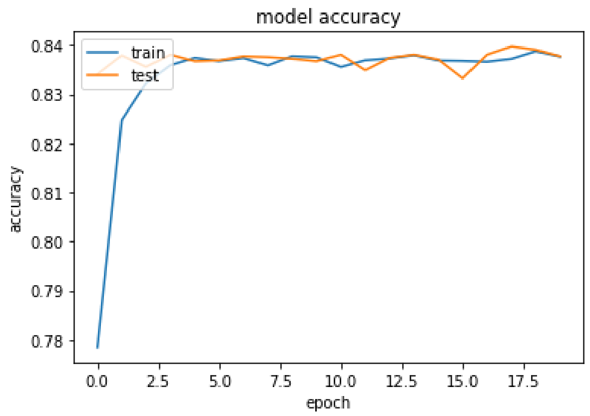


figure 10 model accuracy

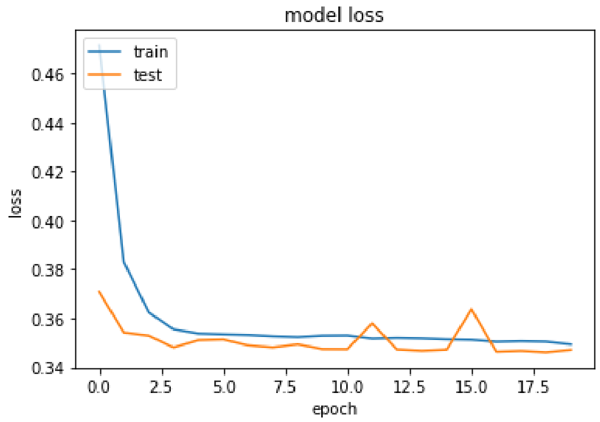


figure 11 model loss

Final result:

	loss	Accuracy
Training dataset	0.3494	0.8377
Testing dataset	0.3471	0.8377

The plot shows that the accuracy increases, and the loss decreases by time in both datasets.

Part 2: Optimizing NN

hyperparameters

table 1 Dropout tuning

	Setup 1	accuracy	Setup 2	accuracy	Setup 3	accuracy
Dropout	0.2	0.8341	0.1	0.8377	0.15	0.8371
Optimizer	adam		adam		adam	
Activation function	sigmoid		sigmoid		sigmoid	
initializer	uniform		uniform		uniform	

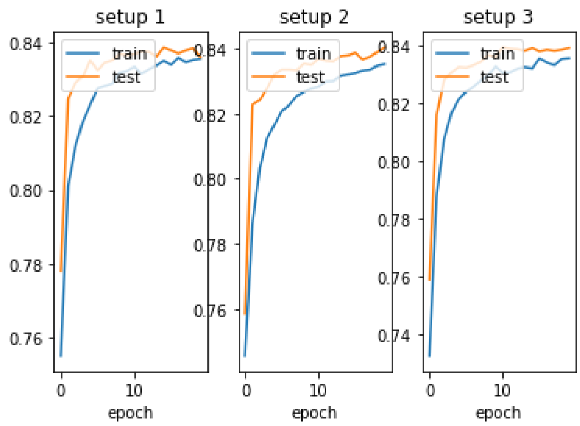


Figure 12 setup 1-3 accuracy

For setup 1-3, the hyperparameter we change is dropout. In each step, the neurons are deactivated randomly to minimize calculation and prevent the result from over fitting. This parameter is usually set as 0.1 to 0.2, therefore, the setup of 0.1, 0.15, and 0.2 are used. Although accuracy of three setup shows no significant difference, the accuracy of setup 2 is relatively more stable and higher.

table 2 Optimizer tuning

	Setup 4	accuracy	Setup 5	accuracy	Setup 6	accuracy
Dropout	0.1	0.8377	0.1	0.7525	0.1	0.7525
Optimizer	adam		SGD		Adagrad	
Activation function	sigmoid		sigmoid		sigmoid	
initializer	uniform		uniform		uniform	

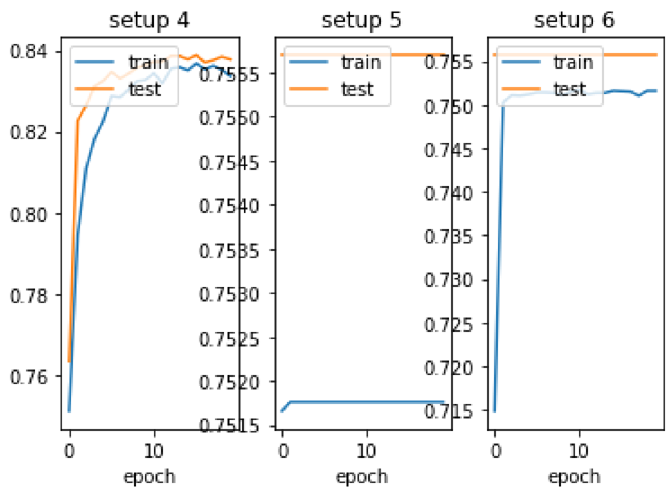


figure 13 setup 4-6 accuracy

Optimizers are used to change the attributes of NN such as weights and learning rate. The purpose of using optimizers is to minimize the loss. In “Adam”, the default learning rate is 0.001, while the SGD learning rate is 0.01 and the “Adagrad” learning rate is 0.001. Although Adam and Adagrad share the same learning rate, the way they update are different. Adagrad reflet the frequency of updating, the update tends to be small if it is updated more frequently. Adam is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. The SGD updating based on the moments, if it is greater than0, SGD accelerates gradient descent in the relevant direction and dampens oscillations.

In figure 13, setup 4 (Adam) shows the accuracy increases stably while setup 6 (Adagrad) increases massively at the beginning of the training. It is because the Adagrad is updated less frequently at the early stage of training. Setup 5 (SGD) shows the accuracy did not significantly improve along the epoch; it is because the parameter w is: $w = w - learning\_rate * g$, therefore, the accuracy will not be updated if the gradian g is extremely small.

table 3 Activation function tuning

	Setup 7	accuracy	Setup 8	accuracy	Setup 9	accuracy
Dropout	0.1	0.8377	0.1	0.8346	0.1	0.7518
Optimizer	adam		adam		adam	
Activation function	sigmoid		relu		tanh	
initializer	uniform		uniform		uniform	

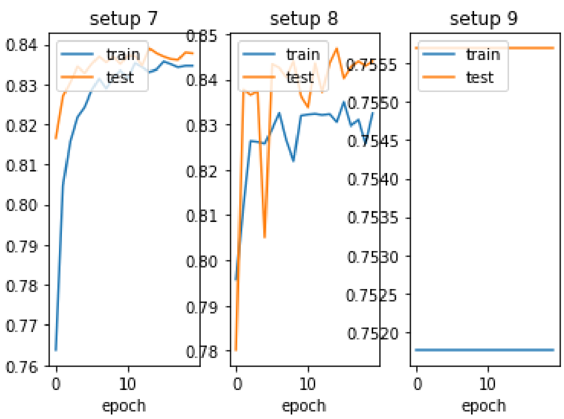


figure 14 setup 7-9 accuracy

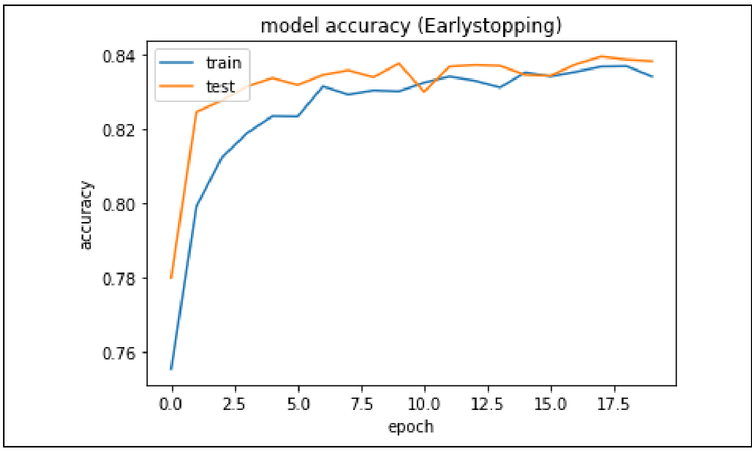
figure 14 represents the activation function change. For setup 7, the activation function is set as sigmoid function $\frac{x}{1+e^{-x}}$, which means most of the $sigmoid(x)$ are positive while $RELU(x)$ only contain the positive value. Comparing to setup 8, the accuracy in setup 7 show higher and more stable accuracy. As for setup 9, the activation function is tanh, which is $\frac{e^x - e^x}{e^x + e^x}$, therefore, the range of $\tanh\,(x)$ is -1 to 1. Since it does not have a certain direction (positive or negative), the prediction result and accuracy. does not improve significantly.

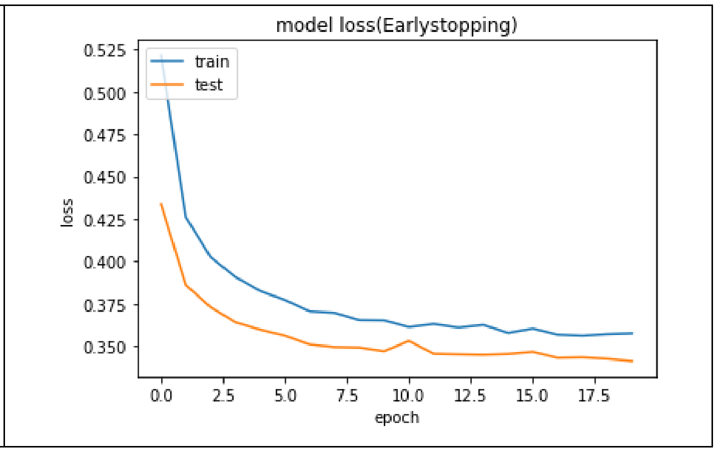
Base and best model:

Dropout = 0.1, optimizer = adam, activation function = sigmoid, initializer = uniform

NN model improvement methods

Early stopping

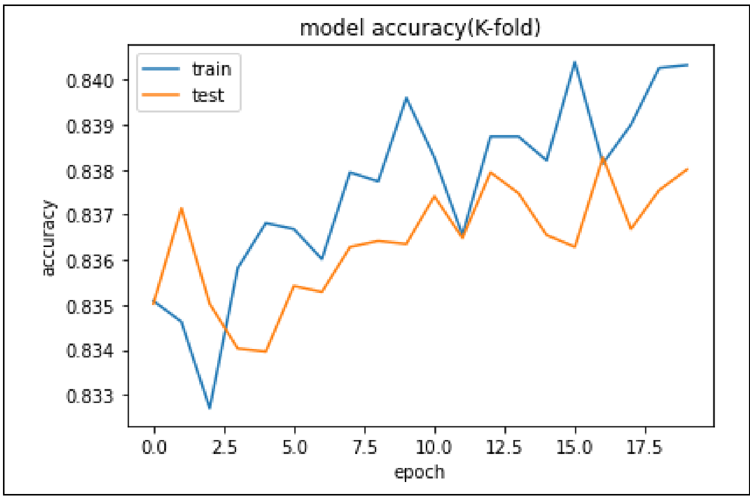


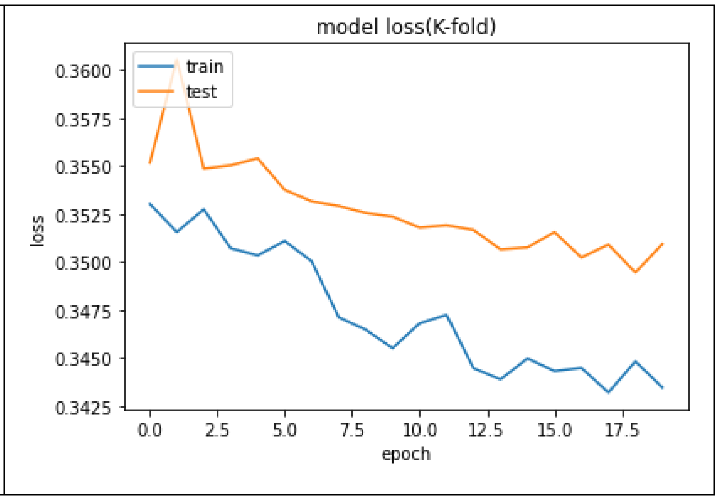


	loss	Accuracy
Training dataset	0.3575	0.8340
Testing dataset	0.3411	0.8382

Early stopping allows the model training step stop early when the model is not improving, which can reduce unnecessary running time. When the epoch is too large, the accuracy stops to increase, and loss stop to decrease. This model result improved along the epoch stably.

K-fold

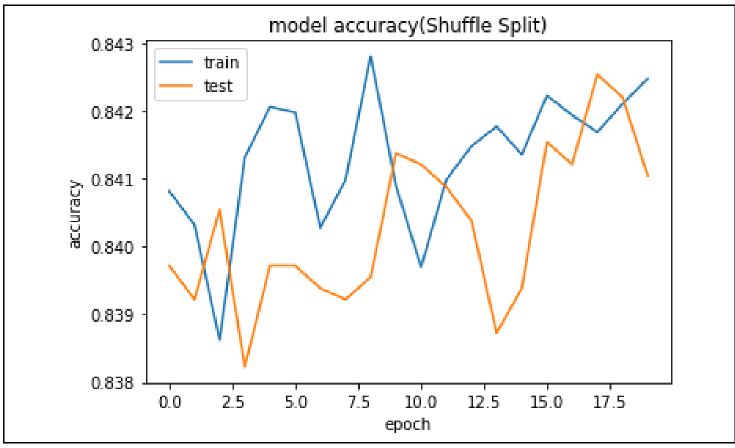


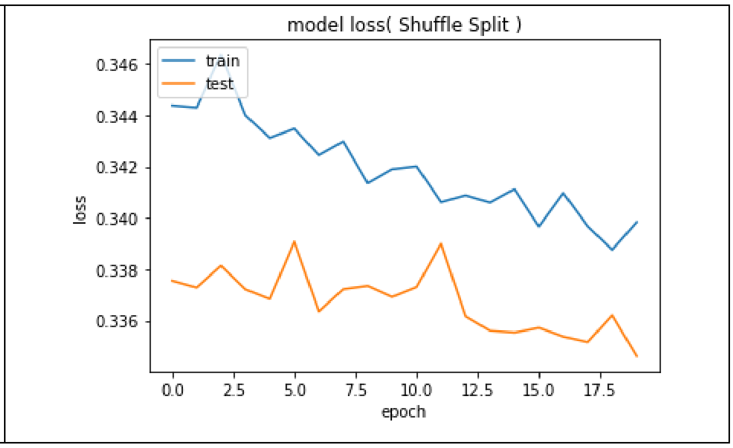


	loss	Accuracy
Training dataset	0.3435	0.8403
Testing dataset	0.3509	0.8380

This method provides test and test indices to split data into subsets, and there is no shuffling by default. The dataset is split into k consecutive folds. Comparing to shuffle split, the data is indices before training, therefore, all the data participate as training and testing data. According to the plots, the accuracy at the beginning of training is higher than original training method, and the loss is also relatively lower.

Shuffle split

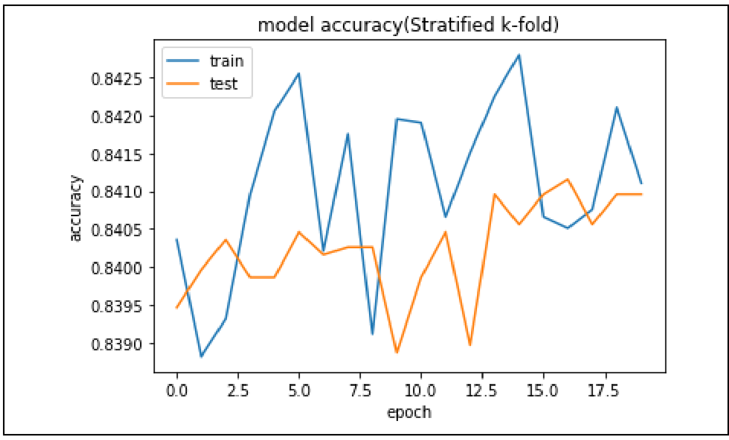




	loss	Accuracy
Training dataset	0.3398	0.8425
Testing dataset	0.3346	0.8410

This method indices the data into training and test sets randomly, the data is shuffle and indices into folds. However, there is a possibility that the folds are the same. The accuracy and loss are less stable because the fold was randomly given, and the similar data may participate as training data multiple times.

Stratified k-fold



	loss	Accuracy
Training dataset	0.3412	0.8411
Testing dataset	0.3386	0.8410

This method is similar to k-fold, but the indices return the stratified folds. The folds are made by preserving the percentage of samples for each class. In this way, the training data includes the same distribution as the whole dataset and other folds. Although the plots seems to be unstable, this model accuracy and loss appears to be the most stable result considering the scaling difference of y.

Reference

Display Deep Learning Model Training History in Keras (machinelearningmastery.com)

Your First Deep Learning Project in Python with Keras Step-By-Step (machinelearningmastery.com)

Dropout Regularization in Deep Learning Models With Keras (machinelearningmastery.com)

Layer weight initializers (keras.io)

machine-learning-articles/how-to-use-k-fold-cross-validation-with-keras.md at main · christianversloot/machine-learning-articles · GitHub

Optimizers - Keras Documentation (faroit.com)

sklearn.model_selection.ShuffleSplit — scikit-learn 1.0.2 documentation