SEMESTER 2 SESSION 2023/2024

SECB4313 BIOINFORMATICS MODELING AND SIMULATION

ASSIGNMENT 2

LECTURER : DR AZURAH BTE SAMAH

SUBMITTED BY : GROUP 4

NAME	MATRIC NO
ERICA DESIRAE MAURITIUS	A20EC0032
INDRADEVI A/P VIKNESHWARAN	A20EC0050
NUR HAZNIRAH BINTI HAZMAN	A20EC0114
SAYANG ELYIANA AMIERA BINTI HELMEY	A20EC0143

1. Identify 4 hyperparameters and propose two values for each hyperparameter. Justify the selection of the hyperparameters and its corresponding values.

Table 1 Proposed hyperparameters values and their justifications

Hyperparameter	**Proposed Value I**	**Proposed Value II**	**Justification**
Activation Function	ReLU	LeakyReLU	ReLU and LeakyReLU enhance learning efficiency and generalization because of the introduction to sparsity through the setting of negative values to zero or a small slope, which enhances. It is also better at handling the vanishing gradient problem and accelerates training time. Thus, gradients are well maintained during backpropagation. Computationally, ReLU and LeakyReLU are much simpler than softmax, which is better suited for the output layer to provide probabilistic class predictions.
Optimizer	AdamMax	RMSprop	RMSprop adapts learning rates more effectively in changing environments, while Adamax handles large and sparse gradients more robustly using the infinity norm. Both optimizers can enhance training performance in specific scenarios as it could be less sensitive to hyperparameter settings and offer better numerical stability.

Learning Rate	0.001	0.0001	A learning rate of 0.001 enables more significant updates every iteration, which helps the model rapidly approach optimal solutions and overcome local minima, facilitating faster convergence. On the other hand, a learning rate of 0.0001 guarantees steady convergence with more precise adjustments, lowering the possibility of divergence and overshooting, particularly when fine-tuning. These two learning rates are utilized as they are more stable compared to the learning rate of 0.01.
Batch Size	32	64	Larger batch sizes (64) allow for faster training but may result in less stable updates, whereas smaller batch sizes (32) can offer more stable training with higher generalization.

2. Construct a tree diagram that displays the proposed hyperparameters and its corresponding values.

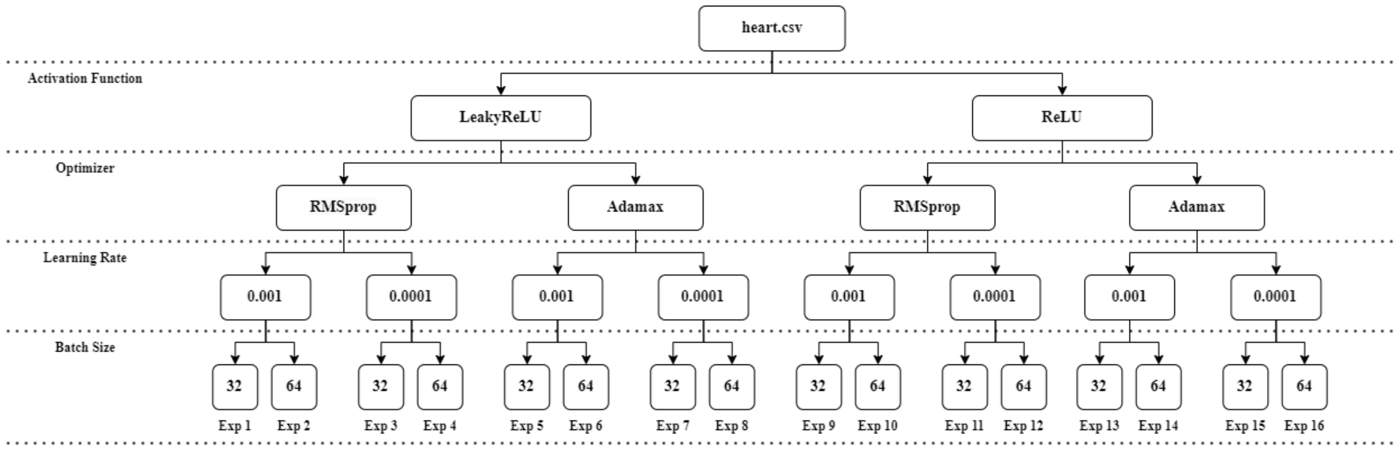


Figure 1 Tree Diagram of proposed hyperparameter values

3. Tabulate the proposed experimental design based on your tree diagram

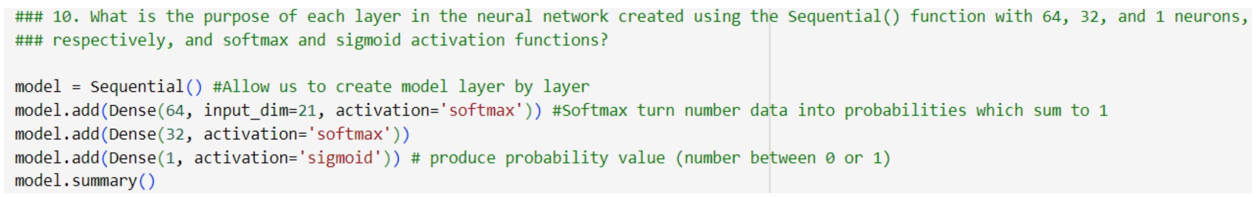
Table 2 Proposed hyperparameter combinations

No.	Activation Function	Optimizer	Learning Rate	Batch Size
1.	LeakyReLU	RMSprop	0.001	32
2.	LeakyReLU	RMSprop	0.001	64
3.	LeakyReLU	RMSprop	0.0001	32
4.	LeakyReLU	RMSprop	0.0001	64
5.	LeakyReLU	Adamax	0.001	32
6.	LeakyReLU	Adamax	0.001	64
7.	LeakyReLU	Adamax	0.0001	32
8.	LeakyReLU	Adamax	0.0001	64
9.	ReLU	RMSprop	0.001	32
10.	ReLU	RMSprop	0.001	64
11.	ReLU	RMSprop	0.0001	32
12.	ReLU	RMSprop	0.0001	64
13	ReLU	Adamax	0.001	32
14.	ReLU	Adamax	0.001	64
15.	ReLU	Adamax	0.0001	32
16.	ReLU	Adamax	0.0001	64

4. Perform *hyperparameter tuning* to improve current results. (Simulate the model and collect the results)

Original Code:

a) Activation Function

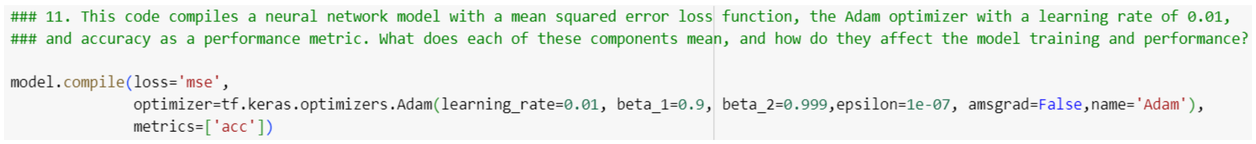


```
### 10. What is the purpose of each layer in the neural network created using the Sequential() function with 64, 32, and 1 neurons,
### respectively, and softmax and sigmoid activation functions?

model = Sequential() #Allow us to create model layer by layer
model.add(Dense(64, input_dim=21, activation='softmax')) #Softmax turn number data into probabilities which sum to 1
model.add(Dense(32, activation='softmax'))
model.add(Dense(1, activation='sigmoid')) # produce probability value (number between 0 or 1)
model.summary()
```

Figure 3 Original Code of Activation Function

b) Optimizer and Learning Rate

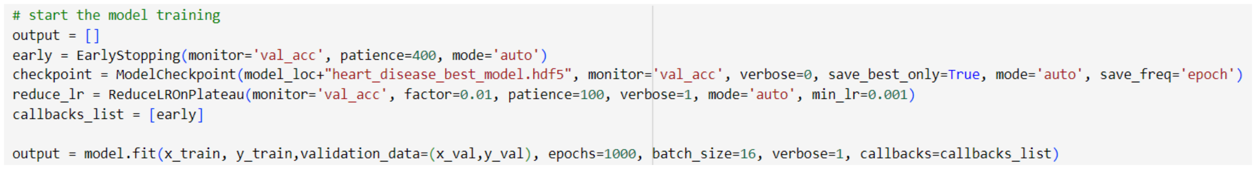


```
### 11. This code compiles a neural network model with a mean squared error loss function, the Adam optimizer with a learning rate of 0.01,
### and accuracy as a performance metric. What does each of these components mean, and how do they affect the model training and performance?

model.compile(loss='mse',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.01, beta_1=0.9, beta_2=0.999,epsilon=1e-07, amsgrad=False,name='Adam'),
              metrics=['acc'])
```

Figure 4 Original Code of Optimizer and Learning Rate

c) Batch Size



```
# start the model training
output = []
early = EarlyStopping(monitor='val_acc', patience=400, mode='auto')
checkpoint = ModelCheckpoint(model_loc+"heart_disease_best_model.hdf5", monitor='val_acc', verbose=0, save_best_only=True, mode='auto', save_freq='epoch')
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.01, patience=100, verbose=1, mode='auto', min_lr=0.001)
callbacks_list = [early]

output = model.fit(x_train, y_train,validation_data=(x_val,y_val), epochs=1000, batch_size=16, verbose=1, callbacks=callbacks_list)
```

Figure 5 Original Code of Batch Size

The figures above illustrate the original model. The hyperparameters in the original model were tuned using the softmax activation function, employing the Adam optimizer, utilizing a learning rate of 0.01 and using a batch size of 16. The proposed values are presented and justified in Table 1.

Hyperparameter tuning was performed according to the experimental design outlined in Table 2. The performance measurements, such as accuracy, precision, recall, and F1-Score, are then used to evaluate the results of the experiment.

5. Tabulate the results based on your proposed experimental design

Table 3 Results of Hyperparameter Tuning Based on Proposed Experimental Design

No.	**Test Loss**	**Test Accuracy**	**Precision**	**Recall**	**F1-Score**
1.	0.13	0.84	0.80	0.86	0.83
2.	0.12	0.84	0.80	0.86	0.83
3.	0.17	0.81	0.72	0.93	0.81
4.	0.19	0.81	0.75	0.86	0.80
5.	0.12	0.84	0.80	0.86	0.83
6.	0.12	0.87	0.81	0.93	0.87
7.	0.20	0.77	0.73	0.79	0.76
8.	0.24	0.55	0.00	0.00	0.00
9.	0.12	0.84	0.80	0.86	0.83
10.	0.12	0.87	0.81	0.93	0.87
11.	0.22	0.77	0.82	0.64	0.72
12.	0.21	0.74	0.71	0.71	0.71
13	0.12	0.84	0.80	0.86	0.83
14.	0.13	0.87	0.81	0.93	0.87
15.	0.24	0.55	0.00	0.00	0.00
16.	0.24	0.55	0.00	0.00	0.00

Notes:

- Highlights in yellow indicate the best hyperparameter combinations.

6. Analyze the results. Which combination of hyperparameters generate the most improved result?

The original model, which had an F1-score of 0.79, accuracy of 0.81, precision of 0.79, and recall of 0.79, was to be improved by hyperparameter tuning experiments. A total of sixteen different hyperparameter combinations involving adjustments to the activation function, optimizer, learning rate, and batch size were examined.

Among all of the experiments, the greatest significant enhancement was obtained from Experiments 6 and 10. Experiment 6 made use of a LeakyReLU activation function, Adamax optimizer, a learning rate of 0.001, and 64 batch sizes. Experiment 10 utilized a ReLU activation function, RMSprop optimizer, a learning rate of 0.001, and a batch size of 64. Both configurations yielded an accuracy of 0.87, precision of 0.81, recall of 0.93, and an F1-score of 0.87. These results indicate improvements over the original model of 0.06, 0.02, 0.14, and 0.08, respectively. These improvements show that the model performs much better when these particular hyperparameters are set.

On the other hand, Experiments 8, 15, and 16 produced significantly lower outcomes than the original model, with an accuracy of 0.55 and an F1-score of 0.00 for each experiment. This illustrates how various combinations of hyperparameters can impact the performance of the model in numerous ways, sometimes resulting in significant decreases.

Overall, the analysis highlights the essential role of hyperparameter tuning in optimizing machine learning models. By systematically testing various configurations, the optimal combination of hyperparameters was discovered, resulting in significant performance improvements. This thorough approach guarantees that the model is robust, dependable, and accurate in its predictions.