# Question(1)

## Q(a)

Download and load dataset.

Normalized images data into [0,1] and split **10000** of 60000 of the whole train datasets as validation dataset.

The final size of train and validation is **50000, 10000.**

Since kmnist dataset size is **28 \* 28** and only 1 channel(greyscale), the input shape is **28\*28\*1**.

A computer screen shot of a program

Description automatically generated

## Q(b)

Build an ANN with input layer and 2 hidden layers.

Flatten input image in input layer.

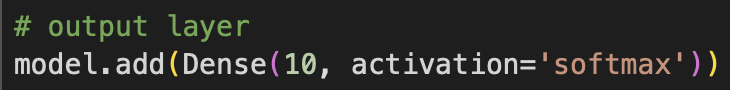
Add two hidden layers with **256** and **128** neurons, each hidden layer utilizes **relu** as activation function, and each layer add **BatchNormalization** and **Dropout** to prevent overfit.

A computer screen with text on it

Description automatically generated

## Q(c)

Add output layer with **10** neurons with **softmax** as activation function.



Each neuron represents one class of classifier.

The **softmax** activation function is ideal for multi-class classification problems, since the sum of output after transformed by softmax is 1, and each output is between [0,1], it can represent the probability of predicting each class.

## Q(d)

A computer screen with many white and green text

Description automatically generated

Utilize **SparseCategoricalCrossentropy** as loss function. Because the labels are integers between [0,9], not one-hot format.

As for optimizer, used **Adam** and **RMSprop**.

**Adam** combines the benefits of **AdaGrad** and **RMSprop** by using running averages of both the gradients (first moment) and the squared gradients (second moment). This results in individual adaptive learning rates for different parameters.

**RMSprop** maintains a moving average of the squared gradients to adjust the learning rate for each parameter.

From the result on test dataset, we can see that **RMSprop** has outperformed **Adam** slightly, achieving a test accuracy of 89.40% compared to Adam’s 88.72%. However, the differences are not substantial, indicating both optimizers are effective for this problem.

## Q(e)

Code for plotting.

A computer screen shot of a program

Description automatically generated

Result:

A graph of a number of training and validation error

Description automatically generated

From plot we can see that losses of both validation and training are decreasing with number of epochs, which means no overfitting.

## Q(f)

Code snippets:

A screen shot of a computer screen

Description automatically generated

A screen shot of a black screen

Description automatically generatedA graph with numbers and a row of squares

Description automatically generatedResult:

We can see that model’s prediction on all classes are above 87% which means the model is well generalization.

Classes 3 ,6 and 8 have particularly high accuracy (95% ,94% and 92% respectively), indicating the model performs very well on these three classes.

Class 1 and Class 9 have a moderate number of misclassifications with other classes comparing to other classes.

# Question(2)

## Q(A)

Initialization of environment

A computer screen shot of a program code

Description automatically generated

Class **GridWorld** initialize with grid size(default=5),start state(0,0),goal state(4,4) and obstacles(2,2),(3,3). And actions U,D,L,R represent move of agent.

Step function calculate the next state, reward/penalty and whether agent finish episode.

A grid with different colored squares

Description automatically generatedLeft is the visualization of Grid.

## Q(B)

1.The size of Q-table should be (number of state) \* (number of actions), in this case, It’s 5\*5\*4.All values are initialized as 0.



2.a Exploration strategy

A screen shot of a computer code

Description automatically generated

2.b Learning rate and discount factor

A screen shot of a computer

Description automatically generated

2.c Update rule for Q values

A screen shot of a computer code

Description automatically generated

To be more specific:

New\_Q = Old\_Q + alpha [reward + gamma(max(value of next state)) – Old\_Q]

3.Code for training:

A screen shot of a computer

Description automatically generated

Training log:

A screen shot of a number of steps

Description automatically generated

Both episode and total steps are recorded.

Code for plot:

A screen shot of a computer program

Description automatically generated

Plot of cumulative reward:

A graph with numbers and lines

Description automatically generated

## Q(C)

Code for experiment and plot:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Following are the policies in different parameters:

A grid with a black square in center

Description automatically generatedA grid with different colored squares

Description automatically generated

Baseline model Higher alpha

A grid with different colored squares

Description automatically generated A grid with different colored squares

Description automatically generated

Lower discount factor Higher exploration rate

So for different parameters:

1. Higher alpha: The agent learns faster and may exhibit more aggressive policy updates.
2. Lower discount factor: The agent focuses more on immediate rewards.
3. Higher exploration rate: The policy is more varied and can capture a wider range of possible strategies.

To conclude:

Learning Rate (alpha): Determines the size of updates to Q-values; higher α results in faster but potentially less stable learning, while lower α leads to slower but more stable learning.

Discount Factor (gamma): Controls the importance of future rewards; higher γ values make the agent prioritize long-term gains, while lower γ values make it focus more on immediate rewards.

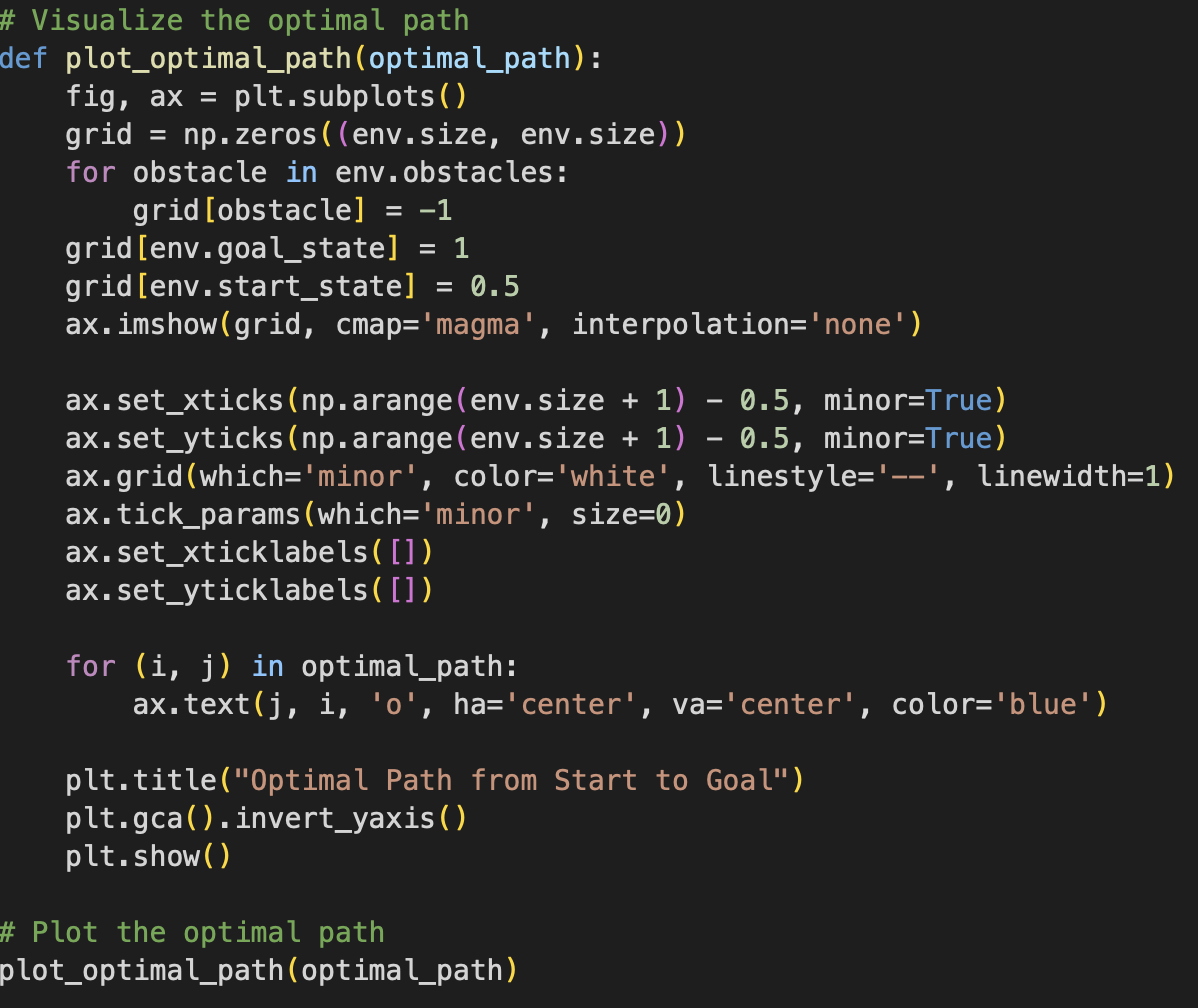
Exploration Rate (epsilon): Balances exploration and exploitation; higher ε values encourage more exploration of the environment, while lower ε values favor exploiting known information.

## Q(D)

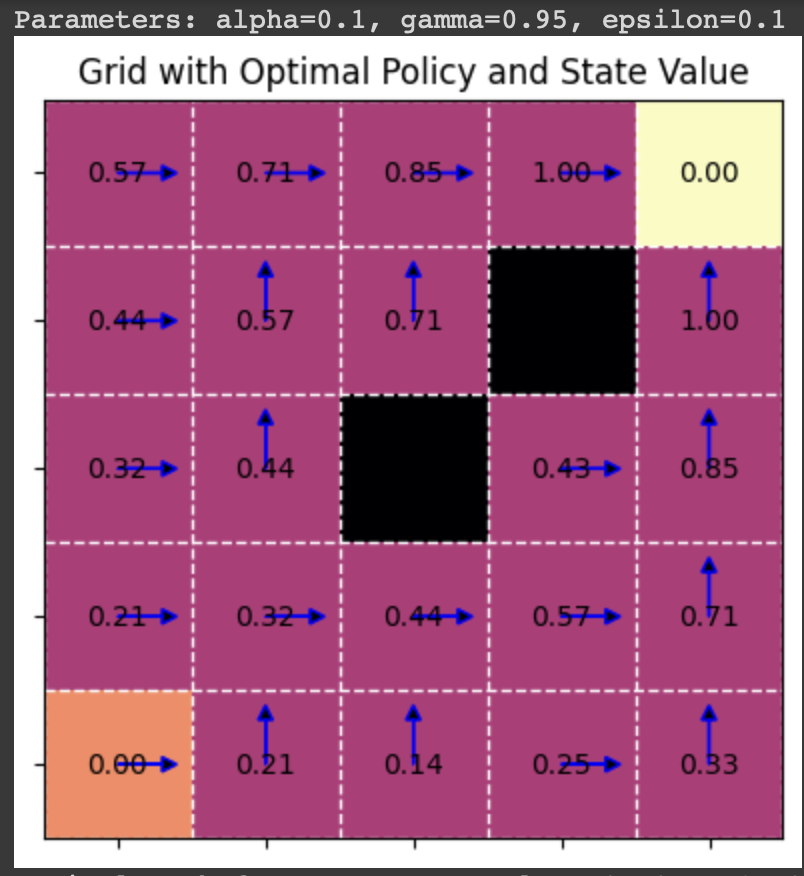
Code:

A screen shot of a computer program

Description automatically generated



Result:

 A screenshot of a computer

Description automatically generated