HW 3: CIFAR-10 Classification: PyTorch Implementation

In this assignment, you will be classifying the contents of the ten types of images in the CIFAR10 dataset. The implementation will be in PyTorch.

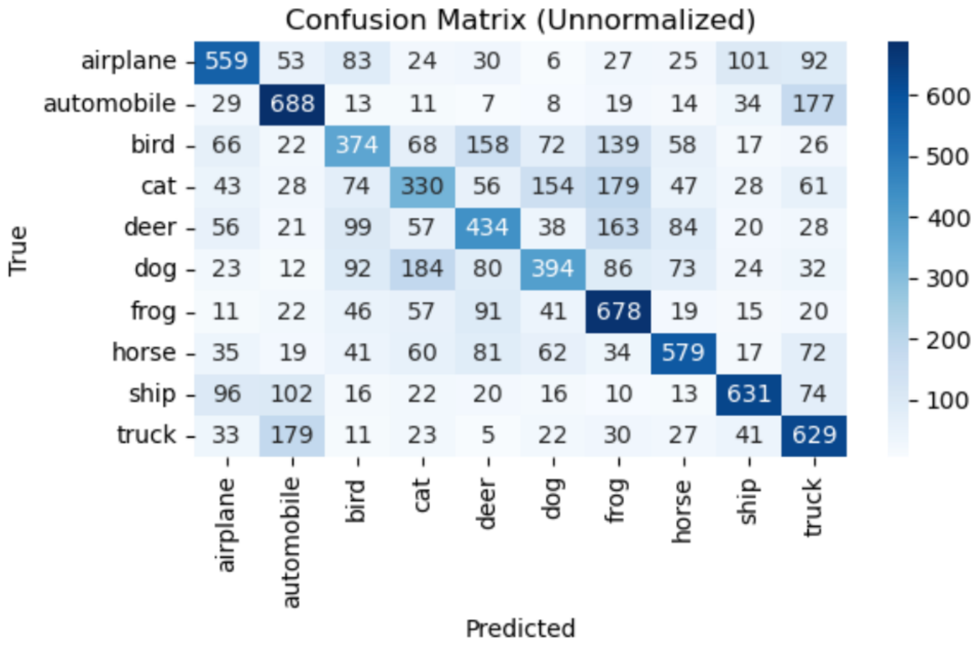
THe learning objectives of the assignment are:

```
Learn to use the PyTorch ML framework for creating and training deep networks.
Learn a structured approach to setting up models, training and testing.
Experiment with and explain different types of regularization in training networks.
```

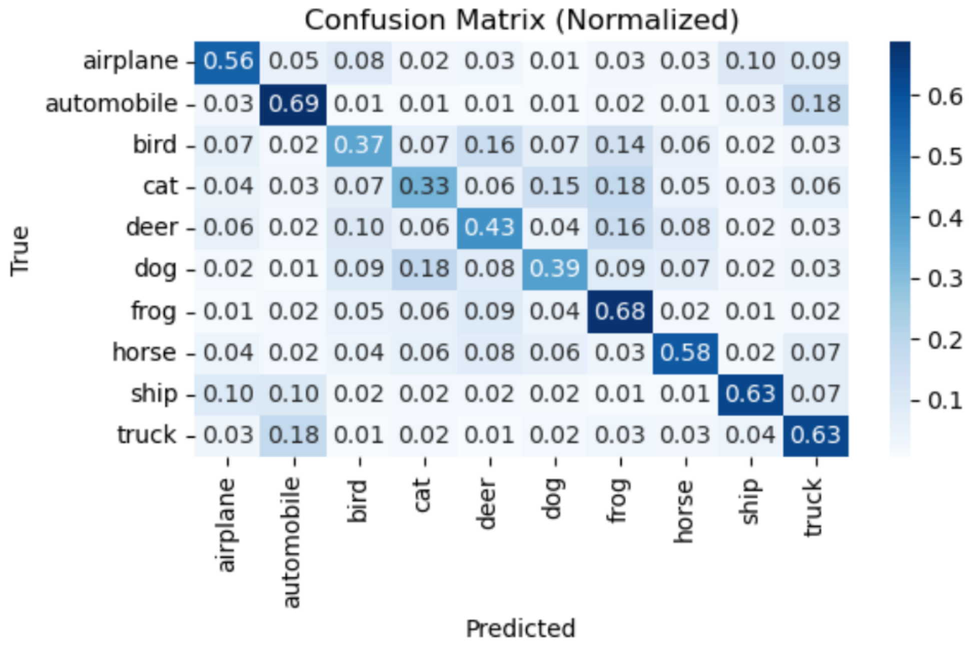
In the last section of the assignment, there are questions that you need to answer about model performance and the concepts for characterizing performance. You may answer the questions in the run notebook that you submit or in a separate pdf document. Since much of the model has been created for you, most of the points in the project will come from your answers.

```
Enter batch size: 32
Files already downloaded and verified
Files already downloaded and verified
Epoch 1, Loss: 1.9488393546104432, Training Accuracy: 31.5975%, Validation Accuracy: 38.39%
Epoch 2, Loss: 1.6632169169425965, Training Accuracy: 41.345%, Validation Accuracy: 43.24%
Epoch 3, Loss: 1.5483962034225465, Training Accuracy: 45.54%, Validation Accuracy: 45.84%
Epoch 4, Loss: 1.4637012629985808, Training Accuracy: 48.6725%, Validation Accuracy: 48.03%
Epoch 5, Loss: 1.3907947424411773, Training Accuracy: 51.36%, Validation Accuracy: 48.65%
Epoch 6, Loss: 1.331476707792282, Training Accuracy: 53.25%, Validation Accuracy: 50.78%
Epoch 7, Loss: 1.275592298603058, Training Accuracy: 55.605%, Validation Accuracy: 51.05%
Epoch 8, Loss: 1.2240527666091918, Training Accuracy: 57.34%, Validation Accuracy: 52.51%
Epoch 9, Loss: 1.173939116191864, Training Accuracy: 58.975%, Validation Accuracy: 52.44%
Epoch 10, Loss: 1.1276487730026246, Training Accuracy: 60.4475%, Validation Accuracy: 52.98%
Finished Training
Test Accuracy: 52.96%
```

```
              precision    recall  f1-score   support

    airplane       0.59      0.56      0.57      1000
  automobile       0.60      0.69      0.64      1000
        bird       0.44      0.37      0.40      1000
         cat       0.39      0.33      0.36      1000
        deer       0.45      0.43      0.44      1000
         dog       0.48      0.39      0.43      1000
        frog       0.50      0.68      0.57      1000
       horse       0.62      0.58      0.60      1000
        ship       0.68      0.63      0.65      1000
       truck       0.52      0.63      0.57      1000
```

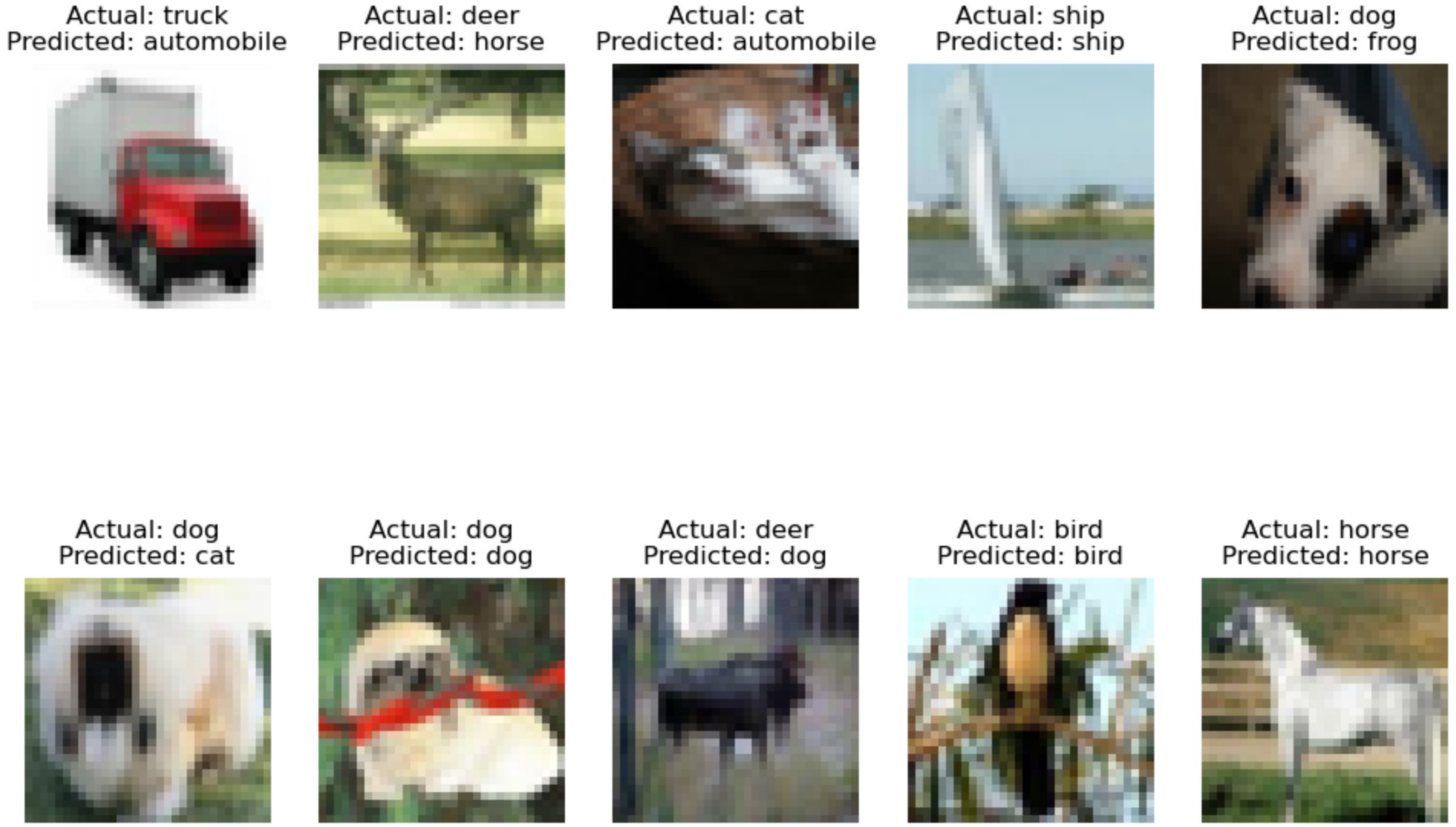


	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	559	53	83	24	30	6	27	25	101	92
automobile	29	688	13	11	7	8	19	14	34	177
bird	66	22	374	68	158	72	139	58	17	26
cat	43	28	74	330	56	154	179	47	28	61
deer	56	21	99	57	434	38	163	84	20	28
dog	23	12	92	184	80	394	86	73	24	32
frog	11	22	46	57	91	41	678	19	15	20
horse	35	19	41	60	81	62	34	579	17	72
ship	96	102	16	22	20	16	10	13	631	74
truck	33	179	11	23	5	22	30	27	41	629



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	0.56	0.05	0.08	0.02	0.03	0.01	0.03	0.03	0.10	0.09
automobile	0.03	0.69	0.01	0.01	0.01	0.01	0.02	0.01	0.03	0.18
bird	0.07	0.02	0.37	0.07	0.16	0.07	0.14	0.06	0.02	0.03
cat	0.04	0.03	0.07	0.33	0.06	0.15	0.18	0.05	0.03	0.06
deer	0.06	0.02	0.10	0.06	0.43	0.04	0.16	0.08	0.02	0.03
dog	0.02	0.01	0.09	0.18	0.08	0.39	0.09	0.07	0.02	0.03
frog	0.01	0.02	0.05	0.06	0.09	0.04	0.68	0.02	0.01	0.02
horse	0.04	0.02	0.04	0.06	0.08	0.06	0.03	0.58	0.02	0.07
ship	0.10	0.10	0.02	0.02	0.02	0.02	0.01	0.01	0.63	0.07
truck	0.03	0.18	0.01	0.02	0.01	0.02	0.03	0.03	0.04	0.63

```
Randomly Select 10 Samples for Testing:
```



6.1 Exercises (40 points)

E1. Using the original file and variables, discuss the overall results of the training. Overall, the system did not do well with an accuracy of just over 50% and the confusion matrices given above. In what situations did the system perform poorly or better? Is the system overfit or underfit? Give two approaches that might help improve system performance based on the results.

E2. Implement one of your approaches in the code. Be sure to highlight what you did and describe it in words in this section. (you can change models, learning rates, optimization, add in regularization, train for different times,etc.) Comment on **why** you made each change. Give the setup that gave you the **best** results in your Python Notebook file with the run results saved.

E3. Compare the difference in the results obtained from two different optimizers. when might you choose one over the other, and why?

6.2 Conceptual Questions: (10 Points)

C1. A “loss landscape” of your neural network’s loss function has a plateau shape, in which there are large areas where the loss is flat. This means the gradient signal will be weak and gradient descent will not update the network parameters very much in each iteration.

What are two activation functions that may cause such a problem, and one activation function that will not?

C2. Which methods may help accelerate the optimization of a model that uses batch gradient descent? a. Using Adam. b. Fine tuning the learning rate using grid search. c. Initializing all the weights to zero. d. Using mini-batch gradient descent.

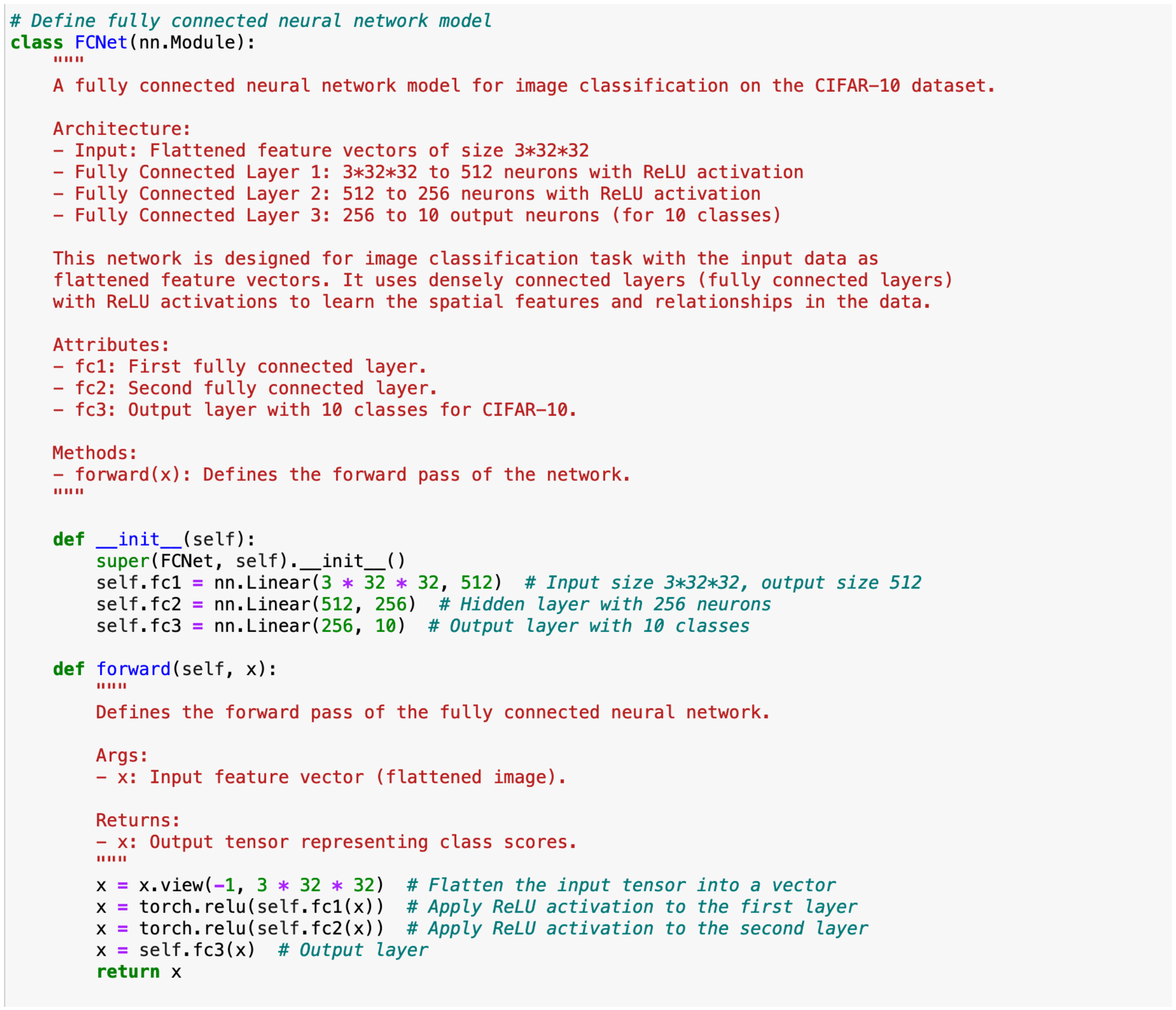
C3. Compute the value of the function $f(\theta)=(\theta-1)^4$ after updating θ in one step of gradient descent, for $\theta = 4$ and $\eta=0.01$. Does it look like the algorithm will converge to an optimum value of the function? Why or why not?

C4. Which statements are true about the step size in gradient descent? a. The step size is the learning rate times the magnitude of the gradient. b. If the step size is too big, gradient descent may oscillate leading to slow or unstable convergence. c. The smaller the step size, the faster we can reach the optimal minima.

C5. State if each statement is true or false and give a brief reason A). L2 regularization encourages sparse weights.

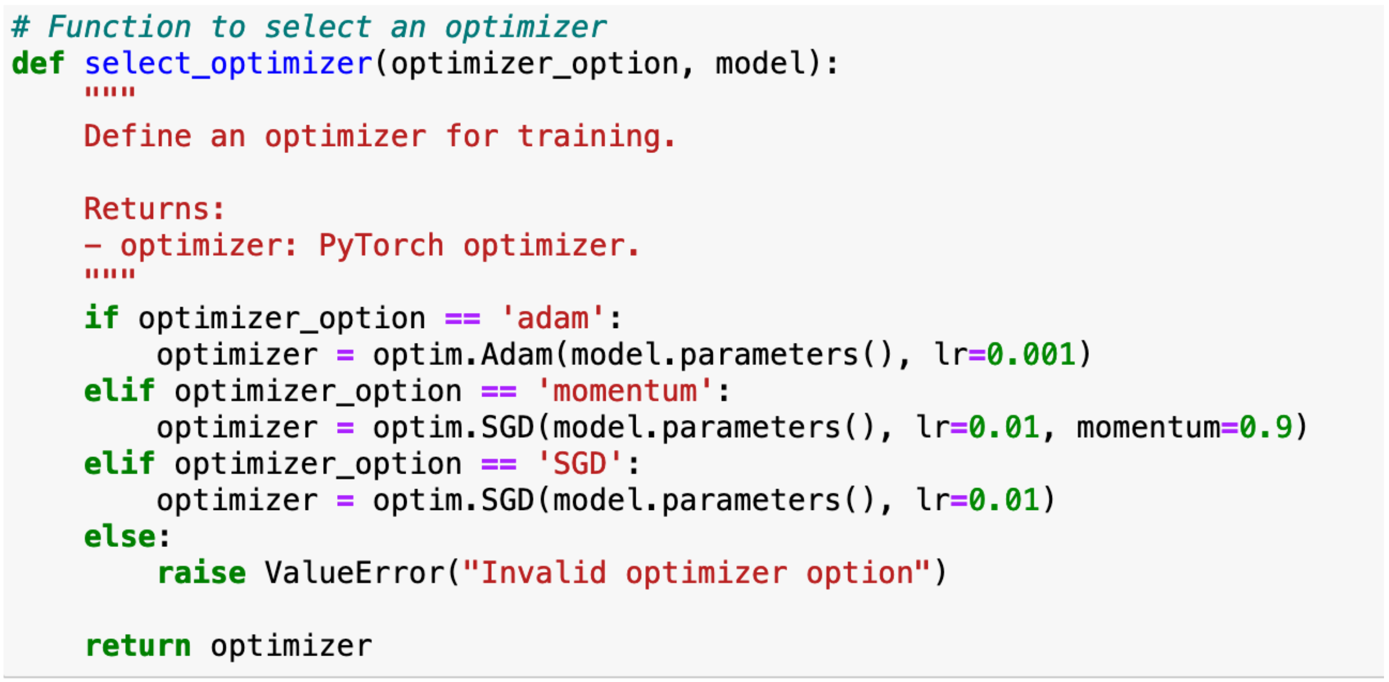
B) You notice that while training your neural network the test loss initially decreases but then starts increasing, while training loss continues to decrease. This means that test loss will only continue to increase if we train further.

Section 2: model definition



```python
# Define fully connected neural network model
class FCNet(nn.Module):
    """
    A fully connected neural network model for image classification on the CIFAR-10 dataset.

    Architecture:
    - Input: Flattened feature vectors of size 3*32*32
    - Fully Connected Layer 1: 3*32*32 to 512 neurons with ReLU activation
    - Fully Connected Layer 2: 512 to 256 neurons with ReLU activation
    - Fully Connected Layer 3: 256 to 10 output neurons (for 10 classes)

    This network is designed for image classification task with the input data as
    flattened feature vectors. It uses densely connected layers (fully connected layers)
    with ReLU activations to learn the spatial features and relationships in the data.

    Attributes:
    - fc1: First fully connected layer.
    - fc2: Second fully connected layer.
    - fc3: Output layer with 10 classes for CIFAR-10.

    Methods:
    - forward(x): Defines the forward pass of the network.
    """

    def __init__(self):
        super(FCNet, self).__init__()
        self.fc1 = nn.Linear(3 * 32 * 32, 512)  # Input size 3*32*32, output size 512
        self.fc2 = nn.Linear(512, 256)  # Hidden layer with 256 neurons
        self.fc3 = nn.Linear(256, 10)  # Output layer with 10 classes

    def forward(self, x):
        """
        Defines the forward pass of the fully connected neural network.

        Args:
        - x: Input feature vector (flattened image).

        Returns:
        - x: Output tensor representing class scores.
        """
        x = x.view(-1, 3 * 32 * 32)  # Flatten the input tensor into a vector
        x = torch.relu(self.fc1(x))  # Apply ReLU activation to the first layer
        x = torch.relu(self.fc2(x))  # Apply ReLU activation to the second layer
        x = self.fc3(x)  # Output layer
        return x
```

4 Optimizer



```python
# Function to select an optimizer
def select_optimizer(optimizer_option, model):
    """
    Define an optimizer for training.

    Returns:
    - optimizer: PyTorch optimizer.
    """
    if optimizer_option == 'adam':
        optimizer = optim.Adam(model.parameters(), lr=0.001)
    elif optimizer_option == 'momentum':
        optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
    elif optimizer_option == 'SGD':
        optimizer = optim.SGD(model.parameters(), lr=0.01)
    else:
        raise ValueError("Invalid optimizer option")

    return optimizer
```