Task:
Building a **single CNN** network with multiclass classifier heads for 3 different classification tasks on the same data.

Objective and Dataset:

In this assignment, you are required to implement a CNN in Python (using Pytorch) to classify CIFAR100 images. CIFAR100 has 100 classes, with 20 superclasses as described here. You have to further classify on another super classification (**groups**) as follows:

1) Plants/Parts of plants (Superclasses “flowers”, “trees”, “fruits and vegetables”)
2) Vehicles (Superclasses “vehicles1” and “vehicles2”)
3) Invertebrates (Superclasses “non-insect invertebrates” and “insects”)
4) Aquatic animals (Superclasses “fish” and “aquatic mammals”)
5) Large animals (Superclasses “large carnivores” and “large omnivores and herbivores”)
6) Man-made articles (Superclasses “food containers”, “household electrical devices”, “household furniture”, and “large man-made outdoor things”
7) People (Superclass “people”)
8) Normal Terrestrial Animals (Superclass “reptiles”, “medium-sized mammals” and “small mammals”)
9) Outdoor scenes (Superclass “large natural outdoor scenes”)

This is the nomenclature we are following for the taxonomy:

CLASSES(100) -> SUPERCLASSES(20) -> GROUPS(9)

Note that for the above synthesized classes, the dataset is unbalanced.

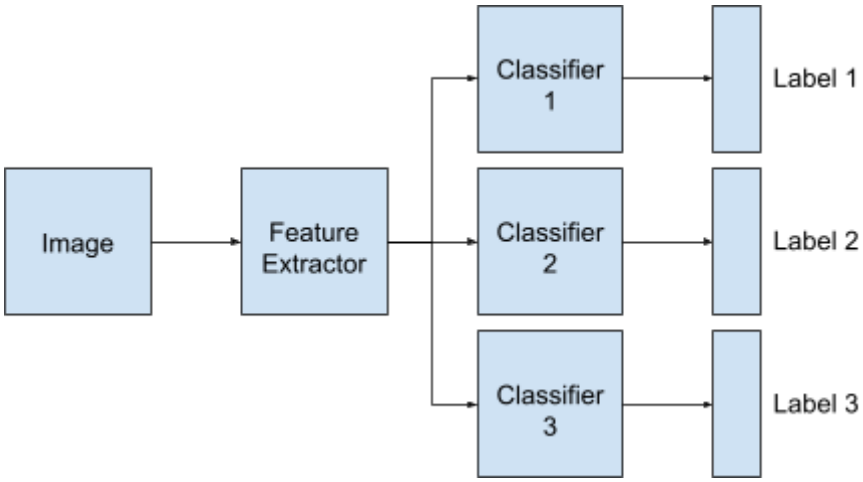

Figure 1: Example of a Vanilla Multihead classifier (You **DO NOT** need to stick to this architecture, experiment with other architectures)

Network Architecture:

1. You have to train a single model that can determine the class, superclass or (synthesized) groups of any given image as required. **Note that you should not create 3 separate models for each task.**
2. You are free to explore your own architecture for your (Deep) CNN.
3. Use Train-test splits as randomized 70:30, 80:20 and 90:10, and report the obtained model performance (Accuracy, Loss, Confusion Matrix) for each split.
4. Use an appropriate loss function and number of epochs.
5. Report total trainable and non-trainable parameters.

BONUS: Modify the CNN training process to reduce the severity of misclassification errors: Severity is defined as follows: if a misclassification happens within the same superclass, assign severity 1; within the same group but not the same superclass, assign severity 2; and in a different group, assign severity 3. (Follow the above nomenclature for better understanding). Train the model using an extra loss function, regularization term, or constraint that gives higher penalties for more severe misclassifications.

NOTE: For the bonus task, you need to have one classification head only to predict the 100 classes. You have to write the logic for incorporating severity for that head only.

Grading Rubrics:

- Proper dataset “groups” synthesis (5 points)
- Handling of the unbalanced dataset (5 points)
- Correctness and completeness of the convolutional neural network implementation (25 points)
- Proper handling of training and evaluation (10 points)
- Documentation and clarity of code (5 points)
- **Bonus (20 points)**