Deep Learning Assignment - 2

Instructions & Guidelines

- This coding assignment revolves around developing custom deep convolutional network and manually training it. Additional instructions for each question are provided accordingly.
- No extensions will be granted for this assignment under any circumstances.
- You have to submit only changerollno.py. Rename it according to your rollnumber. Ensure that you follow the naming convention as **rollnoA2.py**. Any submissions without adhering to the naming convention, having multiple files in submission including .ipynb files, and any additional files will not be evaluated.
  For instance, if your roll number is 1234567 / MT34567, the file name should be 1234567A2.py / MT34567A2.py.

Coding Guidelines

- Follow Readme.txt to setup the environment and start coding.
- You will receive a pipeline for training the architectures. Within the pipeline folder, locate a file named **changerollno.py**. Your task is to edit only this specific file .
- Do not change any variable/class/method name as the pipeline will break. Modifying any other file's code will result in the pipeline not executing properly, leading to a score of 0 marks.
- You have to write code for saving trained checkpoints in **trainer** class and load the saved checkpoint in network in **evaluator** class. If not done 0 marks.
- Do not add any user input variable in any of the class in your file. It will break the pipeline and again no marks.
- Minimum accuracy you have to achieve is 45% for each question to be eligible to get evaluated.

40 marks (10 marks for each Question.)

All these networks are easily trainable on Google Colaboratory

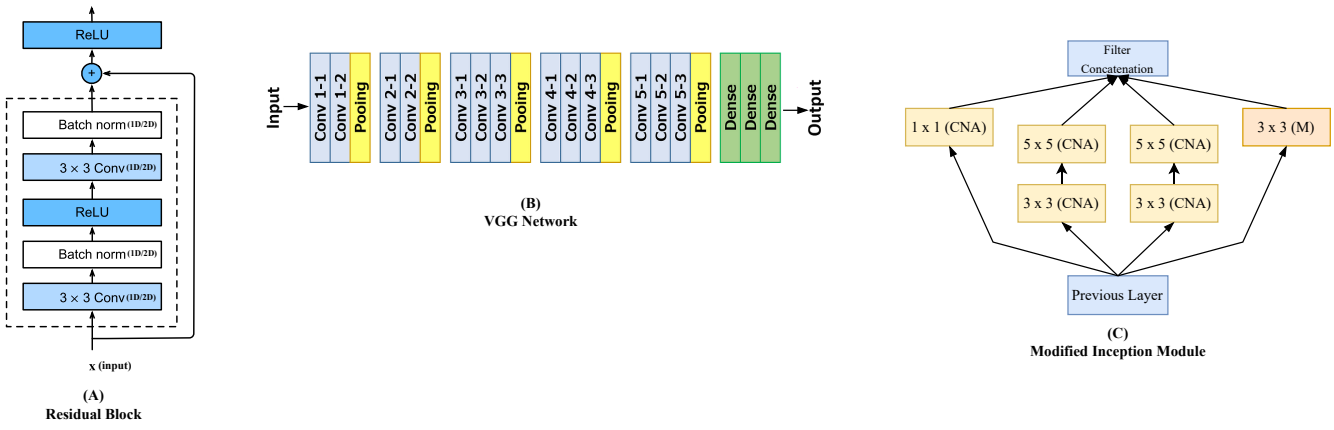

Figure 1: Block diagram for different architectures.

Figure 1 illustrates different blocks from various deep learning architectures. Your task is to implement these blocks and train them using two datasets: **CIFAR-10** and **SpeechCommand V0.02**. Follow the provided instructions for each question. Only use PyTorch's dataset library **torchvision.datasets** for image data and **torchaudio.datasets** for audio data when downloading the datasets.

1. **ResNet**
   In Figure.1, (A) represents a single ResNet block. Here, (**3x3 Conv(1D/2D)**) denotes a single convolution layer with a filter size of 3×3 for 2D data and 1×3 for 1D data, (**Batch norm(1D/2D)**) represents a single Batch normalization layer for 1D or 2D data, and (*ReLU*) represents the rectified linear unit activation function. Your task is to create a network comprising 18 such blocks and train it on both datasets. For the image dataset, use 2D Convolutions and Batch normalization, while for the audio dataset, use 1D Convolutions and Batch normalization. Utilize Cross-Entropy as the loss function and Adam as the optimizer with default parameters specified in the script.

2. **VGG**
   In Figure.1, (B) outlines the modified VGG architecture. After each pooling layer, the number of channels is reduced by 35%, and the kernel size is increased by 25% (with ceil rounding for float calculations). Here, (*Conv n-m*) denotes the n^{th} block and m^{th} layer within that block. Your task is to create a VGG network and train it on both image and audio datasets using the specified loss function and optimizer from the previous question.

3. **Inception**
   In Figure.1, (C) illustrates the configuration of a single inception block, where (*n×n (CNA)*) denotes a sequence of convolution, batch normalization, and ReLU activation with an $n \times n$ convolution filter. Your task is to construct a modified inception network comprising 4 such blocks and train it on both audio and image data, following similar configurations as in the previous two questions.

4. Design a custom network using the following combination of blocks and follow the channel reduction and kernel size increase as in VGG. Train it on both image and audio datasets, adhering to configurations similar to those in the previous questions.

   - Network Architecture
     (a) Input Layer
     (b) Residual Block × 2
     (c) Inception Block × 2
     (d) Residual Block × 1
     (e) Inception Block × 1
     (f) Residual Block × 1
     (g) Inception Block × 1
     (h) Residual Block × 1
     (i) Inception Block × 1
     (j) Classification Network
   - A classification network here refers to a combination of either **nn.Linear** or **nn.Conv1d, nn.Conv2d** layers that produce logits for the classification task.