MILS – Assignment I

Prof. Chih-Chung Hsu

TA Head Chia-Ming Lee and Ming-Hsuan Wu

I. Dataset

- Dataset URL: https://cchsu.info/files/images.zip
- The mini-ImageNet dataset is a smaller version of the ImageNet dataset, created to reduce the computational complexity while retaining the essential properties of the larger dataset.

- **The data is organized as follows:**
  - train.txt: Contains the file paths and labels for the training images.
  - validation.txt: Contains the file paths and labels for the validation images.
  - test.txt: Contains the file paths and labels for the test images.

- **Validation Performance:** During training, you must evaluate your model using the validation set. Report the performance metrics (e.g., accuracy, loss) on the validation set to monitor the model's performance and to tune hyperparameters.

- **Final Performance**: After completing the training, evaluate your model using the test set. Report the final performance metrics based on the test set. This ensures that the reported performance reflects the model's ability to generalize to unseen data.

II. Task A: Designing a Convolution Module for Variable Input Channels

Description:

Design a special convolutional module that is spatial size invariant and can handle an arbitrary number of input channels. You only need to design this special module, not every layer of the CNN. After designing, explain the design principles, references, additional costs (such as FLOPS or #PARAMS), and compare with naive models. To simulate the practicality of your method, use the ImageNet-mini dataset for training, and during the inference process, test images with various channel combinations (such as RGB, RG, GB, R, G, B, etc.) and compare the performance.

Detailed Example:

1. Design Principles:

Dynamic Convolution: Design a dynamic convolution module that can adjust its weights based on the number of input channels. This can be achieved by learning a weight-generating network that takes the input channels as an input and generates corresponding convolution kernels.

2. Implementation Details:

- Convolution Module: Use a dynamic weight-generating network. This network receives the number of input channels and generates convolution kernels accordingly. These kernels can then be used for standard convolution operations.
- Reference: The concept of dynamic convolution can be referenced from the paper 'Dynamic Convolution: Attention over Convolution Kernels' by Wu et al., CVPR 2020. Cost Analysis: Compared to naive models, this design might increase additional computational costs (such as FLOPS) and parameters but can significantly reduce the need for training and storing models for different channel numbers.

3. Experiment Design:

Training and Inference: Use the ImageNet-mini dataset for training and test the model on images with various channel combinations during inference. Comparison: Compare the performance of the model using the dynamic convolution module with naive models across different input channel combinations, evaluating accuracy and computational cost.

III. Task B: Designing a Two-Layer Network for Image Classification

Description:

Design a network using CNN, Transformer, or RNN architectures with 2 to 4 *effective layers* that achieves at least 90% of the performance of a ResNet34 model on the ImageNet-mini dataset (i.e., with no more than a 10% drop in performance). There are no restrictions on the parameter count or FLOPS; however, the number of input and output layers in your architecture must not exceed 4 to 6 layers.

Important Clarification:

For the purposes of this assignment, an *effective layer* is defined as a single basic unit (e.g., a convolutional layer, a GCN layer, or an RNN cell). If you decide to use composite blocks (for example, RRDB or similar structures) that internally consist of multiple operations, you must clearly specify the effective layer count by detailing the structure of each block.

Instructions:

- **Design Strategy:**
  - **Receptive Field Enhancement:**
    Incorporate techniques such as self-attention mechanisms, Graph Convolutional Networks (GCN), or other approaches to effectively increase the network's receptive field.
  - **Architectural Strategies:**
    You may consider using ideas from Wide ResNet or ResNeXt to widen the filters, but ensure that your design adheres to the effective layer count.
- **Implementation Details:**
  - **Architecture Description:**
    Clearly describe your network architecture. Explain what you consider as one effective layer and provide a detailed breakdown if using composite blocks.
  - **Baseline Comparison:**
    Train a ResNet34 from scratch on the ImageNet-mini dataset as a baseline. Avoid using any pre-trained ResNet34 models.
  - **Resource Consideration:**
    Although there are no explicit restrictions on parameter count or computational cost, include a discussion on these aspects, comparing your network with the baseline.
- **Documentation and Reporting:**
  - **Design Rationale:**
    Provide a clear explanation of your design principles, citing relevant literature where appropriate.
  - **Layer Definition:**
    Explicitly define what constitutes an effective layer in your architecture. If using composite blocks, detail their internal structure and the resulting effective layer count.
  - **Experimental Findings:**
    Present both quantitative and qualitative results, and include any ablation studies that help justify your design choices.

IV. Evaluation Criteria

- Theoretical Justification (30%)
  - Provide a clear and coherent explanation of the proposed solution, highlighting how the design addresses the task requirements. Justify the design choices and assumptions made in the proposed approach. Discuss the potential benefits and limitations of the proposed solution compared to traditional methods.

- Experimental Verification (40%)
  - Implement the proposed solution and conduct experiments to validate its effectiveness. Compare the performance of the proposed approach with baseline models in terms of either accuracy, computational cost (such as FLOPS), or both. Provide quantitative metrics to support the claims, such as accuracy, precision, recall, F1-score, or computational cost. Present qualitative results showcasing the visual quality or other relevant performance indicators of the models. Analyze the experimental results and discuss the observed improvements or trade-offs compared to the baseline methods.

- Ablation Studies and Analysis (30%)
  - Conduct ablation studies to investigate the impact of different components or hyperparameters in the proposed solution. Vary the key parameters, such as the structure of the convolution module, the number of layers, or the attention mechanisms, and evaluate their influence on the performance. Provide insights and interpretations based on the ablation studies, justifying the chosen configurations.

- Note:

The focus of this assignment is on demonstrating the effectiveness of the designed convolution module and two-layer network, rather than achieving state-of-the-art performance. The proposed solution should show improvements in either accuracy, computational cost, or both, compared to traditional methods. The claims made in the solution should be supported by theoretical justifications and experimental verification. The evaluation criteria emphasize the importance of providing a clear theoretical justification for the proposed approach, conducting thorough experiments to validate its effectiveness, and presenting the work in a well-organized and understandable manner. You are expected to provide quantitative and qualitative results, perform ablation studies to analyze the impact of different components, and discuss the observed improvements or trade-offs compared to the baseline methods. By meeting these evaluation criteria, you can demonstrate your understanding of the design and implementation of convolutional modules and two-layer networks, as well as your skills in conducting rigorous experiments and analysis.

V. Submission Requirements

4.1. GitHub Repository (50%):

- Create a GitHub repository to host your implementation and related files.
- Include well-documented and organized code for your proposed solution.
- Provide clear instructions in the README.md file on how to run the code and reproduce the experiments.
- Use appropriate git commit messages and branches to track the development progress.
- Ensure that the repository is accessible to the instructor and teaching assistants.
- Readme in markdown or drop the Docker image instead for reproducibility.

4.2. Report (50%):

- Write a comprehensive report describing your proposed solution, experiments, and findings.
- The report should be in a format of your choice (e.g., PDF, Markdown, LaTeX) and can be written in any preferred language.

4.3. Submission Deadline:

- The GitHub repository link and the report should be submitted via Moodle.
- Late submissions will be subject to the course's late submission policy.