COL774: Assignment 3

Enhancing Accuracy in High-Confidence Predictions

1 Overview

In critical applications such as **medical diagnostics**, **autonomous driving**, and **financial systems**, the risks associated with incorrect predictions can be significant. For instance, a self-driving car that fails to detect an obstacle may cause a serious accident, while an incorrect medical diagnosis could lead to harmful treatments. Therefore, in such applications, it is essential for models to make predictions only when they are highly confident in their accuracy. The model should be allowed to abstain from making a prediction if it lacks sufficient confidence in any of the possible options.

1.1 Assignment Objective

This assignment aims to develop machine learning models that not only achieve **high overall accuracy** but also ensure that **all predictions are made with high confidence**. You will train models using the **CIFAR-100 dataset**, which consists of **60,000 32x32 color images** in **100 classes**, with **400 training images** and **200 testing images per class**. Out of these, you will use **40,000 images for training**. Each image is associated with one of the 100 fine-grained class labels, such as "apple," "car," or "tree."

You will also be given an **"accuracy threshold"** as input. Your goal is to make predictions on the test set such that:

1. The overall test accuracy for all classes is higher than the specified **accuracy threshold**.
2. The number of examples classified into any of the classes is maximized.

If your model cannot confidently predict a class label for a given example, you may indicate a **failure to predict** by assigning a label of **-1**.

1.2 Evaluation Metric

The performance of your models will be evaluated using a custom metric that rewards the number of examples for which your model was able to confidently predict a class. It will also penalize cases where the model could not make a confident prediction. **If a model predicts a class label of -1 for an example, it will not be considered for evaluation**. This ensures that only high-confidence predictions are included in the evaluation, focusing on reliability and accuracy.

1.3 Metric Components

1. **Notations**:
   - $\hat{y}_i$: Predicted class label for the i-th example.
   - y_i: True class label for the i-th example.
   - $\mathcal{P}_c$: Prediction set for class c, defined as the set of examples that your model labels as belonging to class c:
   $$\mathcal{P}_c = \{i : \hat{y}_i = c\}$$
   - α: Accuracy threshold for distinguishing between high and low accuracy classes.
   - γ: Penalty factor applied to low-accuracy counts, emphasizing the importance of minimizing incorrect high-confidence predictions.
2. **Class-wise Accuracy**:
   $$\text{Accuracy}(c) = \frac{\sum_{i \in \mathcal{P}_c} 1_{\{\hat{y}_i = y_i\}}}{|\mathcal{P}_c|} \tag{1}$$
3. **Classification Quality Metrics**:
   - **High Accuracy Class Count**:
   $$\text{High Accuracy Count} = \sum_{c \in C} 1_{\{\text{Accuracy}(c) \geq \alpha\}} \times |\mathcal{P}_c| \tag{2}$$
   - **Low Accuracy Class Count**:
   $$\text{Low Accuracy Count} = \sum_{c \in C} 1_{\{\text{Accuracy}(c) < \alpha\}} \times |\mathcal{P}_c| \tag{3}$$
4. **Final Evaluation Score**:
   $$\text{Final Score} = \text{High Accuracy Count} - \gamma \times \text{Low Accuracy Count} \tag{4}$$
   - For the Kaggle competition leaderboard, we will use $\alpha = 0.99$ and $\gamma = 5$.

- For private testing, we may use α values ranging from $[0.98, 0.999]$ and γ ranging from $[5, 20]$.
- Ideally, you should aim to ensure that the low accuracy count is 0, allowing your model performance to become independent of γ.

This metric aims to maximize the number of examples for which your model makes high-accuracy predictions while discouraging low-accuracy predictions. If your model is not very confident about an example, it should tag it with a -1 instead of assigning it a class label.

2 Assignment Structure and Requirements

2.1 Use of Existing Models and Code

For this assignment, you are allowed to use any open-source tool or library, as long as you mention it in the report and can explain how it works and why it works. You may experiment with multiple models and open-source codes, but you must thoroughly understand the final model and code that you submit for this assignment. Your understanding will be checked in the demos.

Moreover, you are not allowed to use pre-trained weights for any model that you use. All the weights must be trained from scratch using the provided dataset. If any models or codes are found to use pre-trained weights, this will be considered a violation of the honor code, and all team members will be subject to disciplinary actions (in line with the course policy on plagiarism).

2.2 Submission Instructions

You will need to make submissions in three places:

1. On Kaggle, submit your predictions for every example in the test set.
2. On Moodle, submit your training and testing scripts in Python, along with your report.
3. On the Baadal machine, submit your model file, which will be used to make predictions on the evaluation set.

On Kaggle, there will be a public leaderboard that will show your scores and ranks on the public test data. This will provide an estimate of your model's score. You may use this information to fine-tune your model, but you must exercise caution to avoid overfitting to the public leaderboard examples.

Your final score will be determined by evaluating your models on a private test set. To achieve this, we will run your training and prediction scripts to generate output on the private dataset. Your team will be ranked based on the performance of your model, and this ranking will be used to assign a score to

your team.

Your report for each submission must include the report from the previous submission and describe any additional optimizations made. You need to mention the experiments you conducted (and their results) and explain why you finalized a particular model for each submission.

2.3 Submission Schedule

The assignment spans **6 weeks**, with deadlines outlined below. Late submissions are not allowed, so it is crucial to adhere to these deadlines. If you miss a submission deadline, we will use the last submitted model and evaluate it for that period. Your final score will be the average of your submission scores, multiplied by your demo score. As mentioned earlier, your submission score will depend on your model ranking (based on the evaluation metric described above).

The demo score will be based on the quality of your report, your ability to explain your approach, and the experiments done to improve model performance. If you have used existing models and codes, you must thoroughly understand them. You are strongly encouraged to read the corresponding papers for the models you used. This understanding will be checked during the demos.

Tentative Submission Deadlines:

1. **Submission 1**: 6th October
2. **Submission 2**: 20th October
3. **Submission 3**: 27th October
4. **Submission 4**: 10th November

Please note that all deadlines are inclusive of the specified dates, and all submissions must be made by 7 PM on the respective date.

2.4 Submission Format

The assignment will be conducted on Kaggle, where you will be provided with input data for training and testing. You can access the Kaggle competition here. You will receive two files:

- **train.pkl**: Contains the training data (40,000 images).
- **test.pkl**: Contains the test data, including an `ID` column and the associated images (20,000 images).

Your submission file should be a CSV file containing two columns:

- `ID`: A unique identifier for each test example, matching the IDs from the `test.pkl` file.
- `Predicted_label`: The predicted class label, in the range of $[-1, 99]$. Use -1 to indicate cases where you choose not to make a prediction; values from 0 to 99 correspond to the 100 class labels in the dataset.

To see your performance on Kaggle, submit your prediction file on the platform in this format.

The deliverables for Moodle and Baadal are as follows:

1. **Training Script**:
   - **Filename**: `train.py`
   - **Command**: `python train.py train.pkl alpha gamma`
   - **Output**: Generate a file named `model.pth` containing the trained model.
   - **Constraints**:
     - **Training Time**: The training process should not exceed 5 hours on the IIT Delhi HPC cluster.
     - **Model Size**: The `model.pth` file must be less than 1 GB.
2. **Model File**:
   - **Filename**: `model.pth`
   - **Description**: Your trained model file for predictions.
   - **Constraints**: The model size must not exceed 1 GB.
3. **Prediction Script**:
   - **Filename**: `predict.py`
   - **Command**: `python predict.py model.pth test.pkl alpha gamma`
   - **Output**: Generate a file named `submission.csv` containing your predictions.
   - **Constraints**:
     - **Inference Time**: The prediction process should not exceed 30 minutes on the IIT Delhi HPC cluster.
   - **Note**: Ensure that the script runs successfully on the HPC environment provided.
4. **Report Submission**:
   - **Filename**: `report.pdf`

- **Content**: A detailed report describing your approach, model architecture, calibration techniques, and any preprocessing or postprocessing steps. Include all libraries used and discuss your results. The report should detail submission-wise changes and document everything you have tried.
- **Report Format**:
  - **Date of Submission**: Include the date of your report submission. The report must be updated weekly, describing each submission. After the first submission, append the subsequent week's report to the previous one. For each update, mention the new submission date and describe the experiments and changes made.
  - **Approach**:
    - **Objective Function**: Describe the objective function in detail. Explain whether it is inspired by existing literature or self-defined, and provide reasoning for its use.
    - **Model Architectures**: Justify the choice of model architectures and explain why they are suitable for generating reliable high-confidence predictions. Provide relevant references.
    - **Justification of Approach and Experiments**: Offer a detailed rationale for your chosen approach, including any experiments performed and their outcomes. Discuss how these led to your final approach.
  - **Use of Open Source Libraries, Codes, and Models**:
    - **Libraries and Tools**: List all open-source tools and libraries used in your project. Provide references and hyperlinks where applicable.
    - **Weight Initialization for Existing Models**: If using pretrained models or existing code, explain how they were initialized, adapted, and used in your solution.
    - **Additional Resources**: Mention any additional resources, papers, or repositories used in your project. Include links and references.
  - **Training**:
    - **Data Preparation**: Summarize your data preparation steps, including preprocessing methods such as normalization and augmentation techniques like rotations and flips. Specify your dataset splitting strategy for training and validation.
    - **Training Methodology**: Outline your training process, detailing the model architecture, hyperparameters (e.g., learning rate, batch size), and any regularization techniques used. Discuss how you optimized the objective function, including

the loss function and any tuning strategies applied during training.

- **Inference**:
  - **Prediction Methodology**: Describe the inference process, detailing how predictions are generated and any postprocessing steps applied.
  - **Confidence Threshold**: If applicable, explain how a confidence threshold was used to filter predictions, including reasoning behind the chosen threshold value.

5. **Requirements File**:

   - **Filename**: `requirements.txt`
   - **Content**: A text file listing all the required dependencies.
   - **Example** `requirements.txt`:

     ```
     torch==1.9.0
     torchvision==0.10.0
     torchaudio==0.9.0
     numpy==1.21.2
     pandas==1.3.3
     scikit-learn==0.24.2
     matplotlib==3.4.3
     seaborn==0.11.2
     ```

   - **Command**:

     ```
     pip install -r requirements.txt
     ```

   - **Note**: Ensure that your `requirements.txt` includes all the necessary packages and their specific versions that your code depends on.

3 Model Submission Guidelines

3.1 FTP Submission

Students should submit their models using FTP to the following directory:

- **FTP Address**: `col774@<Baadal_IP>:/home/col774/A3`

3.1.1 Folder Naming Convention

1. **Folder Name Format**:

   - The folder name should start with a **12-digit random number** followed by the **entry numbers of all team members**, separated by underscores (_).

- **Example**: `123456789012_2020CS12345_2020CS23456_2020CS13456_2020CS44556/`
  - `123456789012`: A 12-digit random number.
  - `2020CS12345, 2020CS23456, 2020CS13456, 2020CS44556`: The entry numbers of the four team members.

2. **Submission Subdirectories**:
   - Inside your team folder, create separate subfolders for each submission:
     - **Submission 1**: `s1/`
     - **Submission 2**: `s2/`
     - **Submission 3**: `s3/`
     - **Submission 4**: `s4/`

3.1.2 Submission Process

1. **FTP to the Server**:
   - Use the following command to connect to the FTP server:

     ```
     ftp col774@<Baadal_IP>
     ```

   - Password and IP will be shared shortly.

2. **Navigate to the Submission Directory**:
   - Change the directory to the submission folder:

     ```
     cd /home/col774/A3
     ```

3. **Create Your Team Folder**:
   - Create a folder following the naming convention:

     ```
     mkdir 123456789012_2020CS12345_2020CS23456_2020CS13456_2020CS44556
     ```

4. **Upload Your Model Files**:
   - Navigate to your team folder and create the submission folder (e.g., `s1` for Submission 1).
   - Upload your model file (`model.pth`):

     ```
     put model.pth
     ```

5. **Important Notes**:

- The submission folder is **write-only**. Once uploaded, you will not be able to view the files, so double-check them before submission.
- Ensure your folder and filenames follow the conventions to avoid any evaluation issues.
- **Submission Timestamp Check**: Submission timestamps will be verified. Any submission containing files with timestamps later than the deadline will be disqualified. No late submissions will be accepted.

3.2 Moodle Submission

In addition to FTP submission, students must also submit the following files on Moodle:

- **Required Files**:
  - `train.py` - The training script.
  - `test.py` - The prediction script.
  - `report.pdf` - A detailed report of your work.
  - `requirements.txt` - A file listing all the dependencies.
- **Submission Format**:
  - Compress the required files into a single ZIP file named as follows:

    `2020CS12345_2020CS23456_2020CS13456_2020CS44556.zip`

- **Upload to Moodle**:
  - Log in to Moodle and navigate to the submission section for this assignment.
  - Upload the ZIP file before the deadline.

Note: Both FTP and Moodle submissions are mandatory.

3.3 Reproducibility

Ensure your results are reproducible by using fixed seed values in your code. This practice will help maintain consistency across different runs.

4 Baseline Approach

1. **Model Training**
   - Train a robust state-of-the-art Convolutional Neural Network (CNN) on the CIFAR-100 dataset using architectures such as:

  - ResNet
  - DenseNet
  - WideResNet
- Focus on achieving high accuracy while avoiding overfitting by using techniques like cross-validation and regularization.

2. **Calibration**
   - Apply calibration methods to improve confidence reliability:
     - Platt Scaling
     - Temperature Scaling
   - These methods align predicted probabilities with actual outcomes, making high-confidence predictions more reliable.

3. **Filtering Low-Confidence Predictions**
   - Set a confidence threshold τ to filter out low-confidence predictions.
   - Exclude predictions below this threshold to improve performance on high-confidence metrics.

5 Suggested Improvement

1. **Custom Loss Function**
   - Design a custom loss function aligned with the evaluation metrics of this assignment.

6 Important Notes

- **Code of Conduct**: Any form of misconduct, including hard-coding results or using pre-trained models against the guidelines, will result in severe penalties, potentially including failing the course.
- **Team Formation**: Teams of up to 4 members are allowed. Each team must submit their code, trained model, and a detailed report explaining their approach and results.

7 Resources

- **Model Calibration**: `https://arxiv.org/abs/1706.04599`
- **CNN Architectures**: `https://paperswithcode.com/sota/image-classification-on-cifar-100`