COMP472: Artificial Intelligence

Due: 11:59 PM (EST), Nov 15, 2024, submit on Moodle.

Include your name, group number (max. 2 people/group), and student number!

1: Project Introduction: Image Classification

In this project, your task is to perform image classification on the CIFAR-10 dataset using some of the AI models learned in this class. You will first analyze the CIFAR-10 dataset, and pre-process the images. You will then build four AI models, Naive Bayes, a decision tree, a multi-layer perception, and a convolutional neural network (CNN) using Python and Pytorch to detect the 10 object classes in the CIFAR-10 dataset. You will then apply basic evaluation metrics: accuracy, confusion matrix, precision, and recall.

Why these tasks matter: Your pathway to AI Mastery By undertaking this project, you're immersing yourself in the world of different traditional and modern AI models that have applications in autonomous vehicles to medical imaging. Mastering image classification with Python and PyTorch is not just an academic exercise; it reflects the real-world demands of industries in pursuit of cutting-edge AI solutions. The tasks you engage with, from designing custom AI model architectures to conducting thorough evaluations, echo the challenges faced by AI professionals deploying solutions in ever-changing real-world settings, such as object detection in autonomous vehicles, obstacle avoidance in robotics, etc. Navigating through this project, understand that each skill and insight you acquire not only deepens your understanding but also strategically positions you for future discussions and initiatives in the rapidly evolving AI landscape.

2: Dataset Overview

The CIFAR-10 dataset contains 50,000 training and 10,000 test RGB images belonging to 10 object classes. Images are of size $32 \times 32 \times 3$.

- In this project, you will only use 500 training images and 100 test images per class. Therefore, your first task is to load the dataset and use the first 500 training images, and the first 100 test images of each class.
- The Naive Bayes, decision trees, and MLPs are not well-suited for direct application to high-dimensional RGB image data. Therefore, you will need to convert them into low-dimensional vectors through feature extraction. Pre-trained CNNs can serve as good feature extractors for image classification tasks. You will use a pre-trained ResNet-18 CNN to extract 512×1 feature vectors for the RGB images. For this, you will first need to resize the images to $224 \times 224 \times 3$ and normalize them, because ResNet-18 pre-trained on ImageNet expects the images in a certain format. You will also need to remove the last layer of ResNet-18. Once these steps are finished, you can pass pre-processed RGB images through the pre-trained ResNet-18 to extract feature vectors. More details about using pre-trained CNNs in Pytorch can be found here: `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`.
- Next, use PCA in scikit learn to further reduce the size of feature vectors from 512×1 50×1.

3: Naive Bayes

1. Implement the Gaussian Naive Bayes algorithm in Python. You are only allowed to use the basic Python and Numpy libraries. Fit the Naive Bayes on the training feature vectors of all 10 classes.
2. Next, repeat the above step but using the Scikit's Gaussian Naive Bayes classifier.
3. Evaluate both of these models on the test set of CIFAR-10 (feature vectors of the images in the test set). Evaluation details are stated below in Section 7.

4: Decision Tree

You are only allowed to use the basic Python and Numpy libraries for all parts unless specified otherwise.

1. Develop a decision tree classifier with the Gini coefficient and maximum depth of 50 and train it on the training set of CIFAR-10 (feature vectors).

2. Experiment by varying the depth of the tree. Observe and document how the depth of the tree influences the model's ability to learn and generalize from the data.
3. Next, repeat step 1 using the Scikit's implementation of a Decision Tree.
4. Evaluate all of the above models on the test set of CIFAR-10 (feature vectors of the images in the test set).

5: Multi-Layer Perceptron (MLP)

You can use the basic Python, Numpy, and Pytorch libraries for this part. Use the feature level test set of CIFAR-10 for all the evaluations.

1. Implement a three-layer MLP (details below) and train it on the feature vectors of the CIFAR-10's training set. The details of the MLP architecture are:

   ```
   - Linear(50, 512) - ReLU
   - Linear(512, 512) - BatchNorm(512) - ReLU
   - Linear(512, 10)
   ```

   You should use the cross-entropy loss `torch.nn.CrossEntropyLoss` for training. Also, use the SGD optimizer with momentum=0.9.
2. Experiment by varying the depth of the network by adding or removing layers. Observe and document how the depth of the MLP influences the model's ability to learn and generalize from the data.
3. Vary the sizes of the hidden layers. Experiment with larger and smaller sizes. Analyze the trade-offs in computational cost and performance of the model.

6: Convolutional Neural Network

You can use the basic Python, Numpy, and Pytorch libraries for this project. Perform your evaluations on the test images of CIFAR-10.

1. Implement and train a VGG11 net on the training set of CIFAR-10. Use the training images directly for this part. VGG11 was an earlier version of VGG16 and can be found as model A in Table 1 of this paper, whose Section 2.1 also gives you all the details about each layer.

   For your convenience, we list the details of the VGG11 architecture here. The convolutional layers are denoted as `Conv(number of input channels, number of output channels, kernel size, stride, padding)`; the batch normalization layers are denoted as `BatchNorm(number of channels)`; the max-pooling layers are denoted as `MaxPool(kernel size, stride)`; the fully-connected layers are denoted as `Linear(number of input features, number of output features)`; the drop out layers are denoted as `Dropout(dropout ratio)`:

   ```
   - Conv(001, 064, 3, 1, 1) - BatchNorm(064) - ReLU - MaxPool(2, 2)
   - Conv(064, 128, 3, 1, 1) - BatchNorm(128) - ReLU - MaxPool(2, 2)
   - Conv(128, 256, 3, 1, 1) - BatchNorm(256) - ReLU
   - Conv(256, 256, 3, 1, 1) - BatchNorm(256) - ReLU - MaxPool(2, 2)
   - Conv(256, 512, 3, 1, 1) - BatchNorm(512) - ReLU
   - Conv(512, 512, 3, 1, 1) - BatchNorm(512) - ReLU - MaxPool(2, 2)
   - Conv(512, 512, 3, 1, 1) - BatchNorm(512) - ReLU
   - Conv(512, 512, 3, 1, 1) - BatchNorm(512) - ReLU - MaxPool(2, 2)
   - Linear(0512, 4096) - ReLU - Dropout(0.5)
   - Linear(4096, 4096) - ReLU - Dropout(0.5)
   - Linear(4096, 10)
   ```

You should use the following in your training process unless specified otherwise: cross-entropy loss `torch.nn.CrossEntropyLoss`, and optimize using SGD optimizer with momentum=0.9.

2. Experiment by adding or removing convolutional layers in your architecture. Observe and document how the depth of the network influences the model's ability to learn and generalize from the data.
3. Adjust the kernel sizes used in your convolutional layers. Experiment with larger kernels (e.g., 5×5 or 7×7) as well as smaller ones (e.g., 2×2 or 3×3). Analyze the trade-offs in terms of spatial granularity versus computational cost and how different kernel sizes influence the recognition of broader features versus finer details.

7: Evaluation

Evaluate the performance of your four models and their variants on the test set of CIFAR-10.

- For each model, generate a confusion matrix to visualize classification performance. Ensure that classes are clearly labeled, either directly on the matrix or using an accompanying legend.
- Summarize your findings in a table detailing the metrics accuracy, precision, recall, and F1- measure. The table must have separate rows for the four models and their variants.
- Only use the libraries clearly stated for each model. If you are unsure about the use of a specific module, please ask using the Moodle Discussion forum.
- It is strongly recommended to run the evaluation using a saved model that you load and test in your program. This avoids having to re-train a model when making any changes to the evaluation part of your code.

8: Report

Please include the following in your report:

- **Model Architectures and Training**
  - Highlight the changes made for variants of each model, specifically noting how each deviates from the main model.
  - Detail the training methodology: number of epochs, learning rate, loss function used, and other relevant training hyperparameters.
  - Mention any optimization algorithms or techniques used, like mini-batch gradient descent, Adam optimizer, etc.
- **Evaluation** Elaborate on the evaluation of your system:
  - Present the metrics (accuracy, precision, recall, and F1-measure) of the four ML models and their variants.
  - Offer insights into each model's performance relative to the others. For instance, if one model has a higher recall but lower precision, discuss its implications in the context of facial image analysis.
  - Display the confusion matrices for each model.
  - Identify which classes were most frequently confused and discuss any model-specific reasons that might be behind these misclassification.
  - Highlight well-recognized classes and speculate on the reasons behind their success.
  - Reflect on how depth (for the decision tree, MLP, and CNN) influenced performance. Did it seem to make the model capture more detailed features or overfit?
  - Discuss how layer size (for MLP) and kernel size (for CNN) variations affected the model's recognition abilities.
  - Summarize the primary findings: which model performed best and why?

9: Deliverables

You are expected to submit your complete project on Moodle. Additionally, your complete project (code, data, files, report, etc.) must be stored in a private repository on Github. Ensure you bundle all the necessary items specified below into a single .zip or .tgz archive for submission on Moodle:

- **Python Code** All Python scripts developed for this project:
  - This encompasses scripts for data visualization and dataset processing.
  - Your PyTorch code for the four models, including the variants, code for evaluation as well as saving, loading, and testing the models.
  - Your code should be well-commented and modular to facilitate easy understanding and evaluation.
  - Ensure that your code is fully functional and runnable. If the markers encounter persistent errors or unresolvable issues, this may impact your grade
- **AI Models** Include all saved models that your trained in your submission:
  - This includes the main model, as well as the saved variants.
  - Running your evaluation code with these saved models on your dataset must result in the same numbers as shown in your report.
- **README** A comprehensive readme.md file:
  - It must enumerate the contents and describe the purpose of each file in your submission.
  - Clearly outline the steps to execute your code for data pre-processing.
  - Describe the steps for running your code to train, evaluate, and apply the models.
  - If your instructions are incomplete and your code cannot be run you might not receive marks for your work.
- **Report** Your finalized project report:
  - Must be structured adhering to the guidelines provided earlier.
  - Submit your report as a PDF file.
  - Hand-written reports will not be accepted.
- **Submission Procedure** You must submit your code electronically on Moodle by the due date.

10: Project Contribution and Grading Policy

In the spirit of collaboration and team unity, by default, all team members will receive the same grade for the project. This default policy is based on the expectation that all team members actively contribute, collaborate, and collectively drive the project to completion. However, we recognize that team dynamics can vary, and there might be instances where contributions are disproportionate. In the event of a dispute regarding contributions:

1. The team must first attempt to resolve the dispute internally. Open dialogue and clear communication are encouraged to ensure all members are aligned on expectations and deliverables.
2. If the internal resolution is unsuccessful and team members believe that the contributions have been significantly uneven, the team should approach the POD (acting as their "project manager") with their concerns. The POD will provide guidance and possibly mediate to help resolve the issue.
3. If after the POD's intervention the dispute remains unresolved, the team may ask the TA to escalate the matter to the course instructor.
4. For a dispute to be considered at this stage, the team must provide clear evidence that delineates individual contributions. This evidence should ideally be in the form of version control records, such as Github change logs, commit messages, pull requests, and code reviews. Merely having more commits

doesn't necessarily indicate greater contribution; the quality, relevance, and impact of the changes will be evaluated.

5. Along with the evidence, a written statement detailing the nature of the dispute, reasons for the perceived uneven contribution, prior attempts to resolve the matter internally, and the involvement of the POD should be submitted.

6. The course instructor will review the submitted evidence and statements. After consideration, they may adjust individual grades to reflect the contributions more accurately. This decision is final.

7. Any claims towards the contribution made after the final (late) submission deadline for each project part will not be considered.

Teams are strongly encouraged to maintain regular communication with the POD throughout the project to preemptively address any potential conflicts and ensure smooth progression. The goal of this policy is not to encourage disputes but to provide a fair mechanism for resolution in the rare event it's needed.

11: Academic Integrity Guidelines for the Project

Upholding the principles of academic integrity is paramount to the learning process. To ensure fairness, clarity, and ethical behavior throughout this project, the following guidelines have been set:

1. **Originality of Work:** All submissions must be the original work of the team members. Copying or adapting work from other teams is strictly prohibited. Using external sources without proper citation is also strictly prohibited (this includes ChatGTP and similar tools, see below). Such actions will be considered academic dishonesty and may lead to a failing grade for the project or other disciplinary actions as deemed appropriate by the university's academic integrity policies. Please make sure you review the academic code of conduct at `https://www.concordia.ca/conduct/academic-integrity.html`. Not knowing the code is not a valid defence for violating it!

2. **Citing External Sources:** Should you use external sources, such as datasets, code snippets, or any other resources, you must provide clear citations. Ensure that you give appropriate credit by adding the source into your report's reference section as defined above (using the IEEE format). Remember, acknowledging sources not only maintains academic integrity but also highlights your diligence in research.

3. **Usage of Large Language Models (LLMs) like ChatGPT:** Recognizing the relevance of LLMs in modern AI, the use of tools like ChatGPT is permitted with specific restrictions:

   - LLMs should complement your work, not serve as the main source of your solutions or content.
   - When using LLMs, both the prompt you provided and the response from the model must be clearly displayed in your report. This ensures transparency and allows for a clear differentiation between student work and LLM-generated content.
   - While LLMs can offer insights or clarify concepts, relying heavily on them diminishes the learning experience. Aim to understand and articulate in your words, using LLMs as a supportive tool, not a primary crutch.

4. **Collaboration vs. Copying:** While collaboration is encouraged for brainstorming and problem-solving, always ensure that what you submit is your team's authentic work. Sharing code, data, or report content between teams is considered a breach of these guidelines.

5. **Ensure Private Repositories:** You are required to make use of an online repository to coordinate and store your team's work, e.g., using Github or Gitlab. However, you must make your repository private (make sure you give access to your team's TA). If your repository is public and another team uses your work, both your team (for sharing) and the copying team (for using) will be held accountable for academic misconduct.

Remember, the purpose of this project is to immerse yourself in the world of AI, develop skills, and gain a profound understanding of the challenges and responsibilities that come with the domain. Adhering to these academic integrity guidelines ensures a level playing field and a genuine learning experience for all