COMP 6721 Applied Artificial Intelligence (Fall 2023)

Project Assignment, Part II

Due date (Moodle Submission): Friday, November 17
Counts for 35% of the course project

In this phase, you'll focus on creating an AI capable of analyzing facial images for classification. Your primary tool will be a deep learning *Convolutional Neural Network* (CNN), developed using *PyTorch*. Your goal is to design and train this CNN on the classes as outlined in Part I. Additionally, a thorough evaluation of your model is essential. The specifics are as follows.

Deep Learning Model Development. Develop a CNN architecture tailored for facial image analysis for "A.I.ducation Analytics":

- This architecture must be implemented in PyTorch and trained using the dataset you've gathered in Part I (you can make changes to your dataset, but make sure you document them in your report).
- Your workflow must be self-contained – meaning, you cannot use external tools or libraries beyond the ones specified below.
- Ensure that your model undergoes a minimum of 10 epochs during the training process. However, be vigilant of overfitting – monitor your model's performance on the validation set and consider employing early stopping techniques if necessary. The optimal number of epochs can vary, so experimentation and close monitoring of both training and validation performance metrics are key.
- Post-training, your code must have the ability to save the trained model. Furthermore, you must have a separate Python program that can load and run the saved model on a complete dataset and an individual image (application mode).
- As part of your exploration, delve into understanding the performance dynamics of your architecture. Specifically, compare the performance of your final architecture against two distinct architectural variants to gain deeper insights:

  Vary the Number of Convolutional Layers: 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.

  Experiment with Different Kernel Sizes: 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.

  Ensure that, for each variant, you document the changes made, the reasoning behind those changes, and the observed performance implications. This will not only allow you to refine your model but will also provide valuable insights into the intricate dynamics of Convolutional Neural Network design.

Evaluation. Evaluate the performance of your main model and its two variants, using the same testing data for all models:

- 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 F_1-*measure*. The table must have separate rows for the Main Model, Variant 1, and Variant 2. For each model, the columns capture the macro-averaged precision, recall, and F1-score; micro-averaged precision, recall, and F1-score; and overall accuracy (refer to Table 1 for the layout).
- Ensure that your dataset is automatically split into training, validation, and test sets. You can use tools like `train_test_split` from scikit-learn or native methods in *PyTorch*. Recommended split ratios are 70% for training, 15% for validation, and 15% for testing. Do not manually split the datasets.
- Although scikit-learn (including *skorch*[1]) is permitted for the evaluation phase, confirm that your model is primarily developed using standard PyTorch. Utilizing common Python modules such as pandas and numPy is acceptable. If you are unsure about the use of a specific module, please ask using the Moodle Discussion forum.

Model	Macro			Micro			Accuracy
	P	R	F	P	R	F	
Main Model	-	-	-	-	-	-	-
Variant 1	-	-	-	-	-	-	-
Variant 2	-	-	-	-	-	-	-

Table 1: How to present the performance metrics of the main model and its two variants

Report. You have to update your report from Part 1 with the following information, added after the previous "Dataset Visualization" section:

CNN Architecture. Describe the architecture of your CNN, as well as the two variants you developed, and provide details on the training process as follows:

1. **Model Overview and Architecture Details:**
   - Provide an overview of the main model, including its layers, activation functions, and other pertinent architectural components.
   - Discuss any specific design nuances, such as regularization techniques, dropout layers, or other unique features.
   - Highlight the changes made for Variant 1 and Variant 2, specifically noting how each deviates from the main model.
2. **Training Process:**
   - Detail the training methodology: number of epochs, learning rate, loss function used, and other relevant training hyperparameters.

[1] See https://github.com/skorch-dev/skorch

- Mention any optimization algorithms or techniques used, like mini-batch gradient descent, Adam optimizer, etc.

Length: ca. 2–3 pages (excluding images/diagrams)

Evaluation. Elaborate on the evaluation of your system:

1. **Performance Metrics:**
   - Present the metrics (accuracy, precision, recall, and F_1-measure) of the Main Model and the two variants using the table defined above.
   - 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.
2. **Confusion Matrix Analysis:**
   - Display the confusion matrices for each model.
   - Identify which classes were most frequently confused and discuss any dataset or model-specific reasons that might be behind these misclassifications.
   - Highlight well-recognized classes and speculate on reasons behind their success.
3. **Impact of Architectural Variations:**
   - Reflect on how depth (number of convolutional layers) influenced performance. Did it seem to make the model capture more detailed features or overfit?
   - Discuss how kernel size variations affected the model's recognition abilities, especially concerning finer versus broader facial features.
4. **Conclusions and Forward Look:**
   - Summarize the primary findings: which model performed best and why?
   - Offer suggestions for future refinements in model architecture or training strategies.

Length: ca. 2 pages (excluding tables)

Note: if you made any changes to your dataset from Part I, make sure you update the previous sections in the report. Additionally, clearly explain what was changed in your dataset and why.

Deliverables. You are expected to submit your complete project, not just the additions from Part II. Like for Part I before, 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 cleaning, data visualization, and dataset processing.
- **New:** Your PyTorch code for the CNNs, including the variants, code for evaluation as well as saving, loading, and testing the models.
- Note: Only pure Python code (`.py` files) will be accepted. Jupyter notebooks or other formats will not be considered.
- 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.

Dataset. A file or document detailing the provenance of each dataset/image:

- For publicly available datasets, incorporate only a reference with the necessary details such as dataset name, source, and licensing type (i.e., do not try to submit your whole dataset content here).
- For custom or modified images, ensure you include them alongside any manually crafted metadata.
- Supply 10 representative images from each class within your archive and incorporate a direct link to the full dataset in your repository (e.g., on Github).

README. A comprehensive `readme.txt` or `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 (a) data cleaning and (b) data visualization.
- **New:** 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.
- **New:** incorporate the new sections defined above with the sections from Part I for a complete report.
- Submit your report as a PDF file.

Originality Form. Include a **single** *Expectation of Originality* form:

- Available at https://www.concordia.ca/ginacody/students/academic-services/expectation-of-originality.html
- This form, attesting to the originality of your work, must be electronically signed by **all** team members.
- If the form is missing, your project will not be marked!
- **New:** If your group has not changed, you can submit the same form as for Part I.

Submission Procedure: You must submit your code electronically on Moodle by the due date (late submission will incur a penalty, see Moodle for details).

Project Demo. We will schedule demo sessions for your project using the Moodle scheduler. The demos will be on campus and all team members must be present for the demo. Guidelines for preparing for the demo will also be posted on Moodle.

Other Guidelines. Please refer to the Project Part #1 document for the *Academic Integrity Guidelines for the A.I.ducation Analytics Project* as well as the *Project Contribution and Grading Policy.*