Mini-Projects: 10-605/10-805, Fall 2023

Guidelines

- You must work in groups of 2 or 3 people. You will choose your own teammates so we strongly encourage you to meet people in the class.
- Groups made up of only 10-605 students will choose to complete either project A or project B. Groups with at least one 10-805 student must complete both project A AND project B.
- No grace days can be used on Mini-Project deadlines.

Deliverables

The mini-project is worth 15% of your total course grade and consists of 2 components:

1. **A project survey (due November 10th at 11:59 PM ET) [10% of the mini-project grade]**
    - The "proposal" is a Google form that asks for the names of the students in your project group, which project you intend on tackling, and some details about your intended mini-project plan. Note that the last question(s) of the form will require some thought / effort, so please allocate some time for this.
    - *Only one person per group should fill out this form*
2. **A Final Project Deliverable (due November 29th at 11:59 PM ET) [90% of the mini-project grade]**
    - Each mini-project has unique deliverables described below in their corresponding section.

Project A Details

- **Mini-Project: *Model Compression Competition***
    - We will provide you with a large model and dataset.
    - You will select and implement 3 methods for performing model pruning.
        - *You must submit your source code and you **CANNOT** use existing libraries*
    - Your primary deliverable will be a 1-2 page retrospective explaining what worked well and what did not work.
    - You will also submit your model weights for all three of your models and the code you used to generate these weights.
    - The 3 highest scoring teams will receive extra credit (5 points on their mini-project grade)

Model Compression Competition

In this mini-project, you will explore methods for neural network pruning as part of a class-wide competition: attempting to compress a simple neural network which has 592,933 parameters. This network achieves ~63% test accuracy on a 5-class classification task after training for 50 epochs with the default parameters in the notebook. To help get you started, we have provided a simple example of manipulating the weights, assigning new weights to the model, and evaluating the effect on accuracy.

Competition Logistics:

1. **Starter notebook:** We will provide a notebook for you which provides data loading helper functions, defines the model architecture, and provides some simple functions to save and load model weights. To download the notebook, go to **this link**. Make sure to use your Andrew id.
2. **Getting the data**: Download the compressed training and validation data from **this link**. Upload it to Colab using the Files tab in the sidebar and clicking the upload button. The first few cells in the notebook untar and load the training and validation sets. We will use hidden tests on Gradescope to test the final performance of your model. To avoid uploading the dataset every time the runtime disconnects, you may want to upload the dataset to Google Drive and mount it from colab.

3. **Training and pruning the network**: You need to first train the network in the notebook until convergence. Starting from this pre-trained full model, you can explore different strategies to prune the network (i.e., setting some weights to zero) without degrading the accuracy. Your output should be a pre-trained model with the '.h5' extension that has the same architecture as the neural network we define in the notebook. *Changing the architecture or designing a new model architecture yourself is not allowed and will cause the autograder to fail*. It might be helpful to store the weights of your model before pruning in order to save your progress. You can achieve this through model.get_weights() and model.set_weights(saved weights).
4. **No external data or libraries**: You must use only the provided train/validation data to train your model. You may not use any network pruning libraries; we will manually verify all submissions to make sure that's the case.
5. **Submission**: You should submit the final pruned model weights with the names:
    - my_model_weights_1.h5
    - my_model_weights_2.h5
    - my_model_weights_3.h5
      (One for each of your distinct methods)

   In the notebook, we specify how to download the model weights to your local workspace. You will then upload the model weights to Gradescope for autograding. You can submit an unlimited number of times. (But as noted below, beware of overfitting if you submit too often.) The final score will be determined according to the formula in the next bullet, using your last submission and the best score of your three model weights.

   You must also complete a 1-2 page project report, detailed below, and you must submit all of your code used to generate your final weights.
6. **Evaluation**: Each submission will receive a score, which is a function of the accuracy of the test data and the sparsity level of the model. The autograder calculates the model score as follows:

```
if accuracy > 0.6 and sparsity > 0:
      score = (accuracy + num zero weights / total parameters) / 2
else:
      score = 0
```

   This metric encourages sparsity (the ratio between the number of zero weights and the number of total parameters) while ensuring a reasonable accuracy (accuracy > 0.6). You must obtain an accuracy higher than 0.6 and a total score higher than 0.36 to receive credit, though we encourage you to search for higher scores as part of the competition!
7. **Competition**: We will give up to 15 bonus points on this mini-project to the top teams with the best scores: 15 points for first place, 10 for second, and 5 for third. We will maintain a leaderboard on Gradescope during the competition so that you can see how you are doing, but ***the leaderboard does not determine your final competition rank.*** Instead we will hold out

some additional test data, and rank teams based on their performance on this test data using the same evaluation formula as above. This test data for the competition only affects your competition rank, not any other component of your grade. We will review the top submissions and may disqualify teams that don't follow all the above rules; in this case the next-best team will move up a place. ***Warning:*** beware of overfitting. If you submit too many times, you will be implicitly fitting your model to the leaderboard's hidden data. In this case the leaderboard score may become overly optimistic, and your final test score may be lower than expected.

Grading:

- **Project proposal (10%):** Provide information about your group and the methods you will use for pruning.
    a. ***Group*****:** list the members of your group
    b. ***Methods***: You must implement *at least 3 different methods* for neural network pruning from scratch. See below for examples (note: you are *not* limited to the methods below---these are just examples).
        - Magnitude-based pruning: https://arxiv.org/abs/1506.02626
        - L1-norm based filter pruning: https://arxiv.org/abs/1608.08710
        - ThiNet: https://arxiv.org/abs/1707.06342
        - Regression based Feature Reconstruction: https://arxiv.org/abs/1707.06168
        - Network Slimming: https://arxiv.org/abs/1708.06519
        - Sparse Structure Selection: https://arxiv.org/abs/1707.01213
- **Project report (90%)**: Answer the following questions in a 1-2 page report, which you will submit as a PDF on Gradescope:
    a. ***Methodology*** [20 pts]: Which 3+ pruning methods did you use? Describe each briefly.
    b. ***Comparison*** [60 pts]: Compare the performance of the pruning methods: For example, which method led to the best overall model score? Which was easiest to implement? Provide a plot exploring the methods in terms of the Pareto frontier of sparsity vs accuracy.
    c. ***Reflection*** [10 pts]: Did the results match what you expected? What did you learn about model pruning as part of this project?
- **Code:** *You must submit the code/notebooks you used to implement your 3+ pruning methods. Note that you must implement these methods from scratch---you can't use pre-existing pruning libraries. You will receive a '0' if you do not submit your code or you do not implement your methods from scratch.*
- **Extra credit**: The 3 top teams on the leaderboard in Gradescope will receive up to 15 extra credit points on their mini-project grade.

Resources:

- Rethinking the value of network pruning: https://arxiv.org/pdf/1810.05270.pdf
- Modeling of Pruning Techniques for Deep Neural Networks Simplification: https://arxiv.org/pdf/2001.04062.pdf

Project B Details

- **Mini-Project: QA Chatbot Evaluation Report**
    - You will use the Zeno platform to write a report answering the questions posed below. This is the primary deliverable.
    - You will also submit your code used to help you write this report.

Overview

- It is easier than ever to prototype complex machine learning systems, such as chatbots, using large language models. However, understanding a model's limitations and how they need to be improved or fixed remains a significant challenge.
- For this mini-project, you will be tasked with conducting a thorough evaluation of multiple question-answering chatbot models with the goal of:
    - Picking the best model for a specific task
    - Identifying significant issues that could cause problems in production
- You will have access to a dataset of insurance chatbot conversations and the outputs of 6 chatbot models in a Jupyter notebook. The data will be set up to use with Zeno (zenoml.com), a system for data analysis and evaluation.
- You will be tasked with exploring the data, adding features, and running analyses.
- The final deliverable will be a report created with Zeno discussing your final recommendations with supporting evidence and charts.

Details

Evaluation is an integral part of any successful ML project. Rigorous evaluations provide us with the information for important development decisions such as which model to use, where to collect more training data, whether or not a model is ready for production, etc. In this mini-project, you will use the Zeno evaluation platform to conduct such an analysis, taking the role of a specific persona and writing a Zeno report about your findings. We are using Zeno for evaluation as it is tied to the data to be analyzed, provides different exploration tools, supports interactive charts, has a reporting interface, and is developed as an open-source project here at CMU. Here are two example reports similar to your final deliverable: Audio Transcription Report and GPT Translation Report.

For this mini-project, you will take the persona of an AI engineer who has been recently hired at Scottie Insurance Inc. Your job is to create a question-answering chatbot that customers can interact with to get general information about their policy and take actions like filing a claim. You have created multiple candidate models by prompting leading models such as OpenAI's GPT3.5 and Llama 2. You then ran the models on a dataset of existing customer conversations with customer service representatives. Your manager has asked you the following questions:

Initial Questions:

1. How do the models handle sensitive data, and in particular social security numbers?
2. How verbose are the models? Do some models produce longer answers?

High-Level Questions:

1. Which model should we be using for our use case? Can we get away with using the cheaper open-source models instead of OpenAI? Why or why not?
2. If we pick the best model, what issues should we fix or look into before we deploy it in the wild? Does the model hallucinate? Does it make up facts?

Your assignment is to use the Zeno platform to write a report answering both the initial and high-level questions for your manager, supporting your answers with evidence. **We are looking for answers that have both text and supporting charts / instance views.**

Project Setup:
Here are the steps to get started:

- **Get starter code:** Copy this Colaboratory notebook to get started
- **Create a Zeno account:** Go to hub.zenoml.com and create an account. Get an API key at https://hub.zenoml.com/account to use in the starter notebook.
- **Run** the 10-605/805 mini project chatbot evaluation.ipynb notebook with your API key to set up an initial Zeno project.
- **Using Zeno:** Watch this video to learn more about how to use Zeno to evaluate your models and create charts and reports.
- **Running analyses:** You will likely have to add new features to your DataFrame (e.g. output length) to generate the slices and analyses necessary for your report.
- **Write a report:** Your final deliverable will be a Zeno report (see this example). To create a report, go back to the main page of hub.zenoml.com and click on create report. You can include text, charts, or instance views in your report to quantitatively back up your recommendations.
- What makes a good report? (adapted from the CMU ML blog)
    - **Length:** Every report should consist of a title and a main text. In general, reports should have 1000 - 3000 words, which takes about 8-25 minutes to read.
    - **Content:** Start with a high-level description of the problem that your persona is solving, followed by the evaluation results. Throughout, motivate the problem by providing adequate context. Feel free to use hyperlinks and other markdown throughout the report to link to useful sources.
    - **Visual aids:** Aside from text, visual aids, when used appropriately, are highly effective in conveying information. Include thoughtful data visualizations that you can create in Zeno. Authors are free to express their creativity.
    - **Style:** Since this is more like a blog post rather than an academic paper, authors do not need to use a rigorous style of writing. In contrast to academic papers, we encourage authors to include more high-level intuitions of problems and concepts. Visualizations are highly encouraged.
    - **Examples:** You can find some public reports in Zeno that serve as examples.

Grading
Your deliverable for the proposal is the google form described at the beginning of the document. Your deliverable will be your Zeno report (details on how to submit the report will be posted to Piazza prior to the submission deadline).

- **Project proposal (10%)**: A bar chart showing comparing the overall ChrF on all instances.
- **Zeno Report (90%):**
    - Structure [20 pts]**:** Did you specifically answer the questions asked? Did you recommend the best model and identify potential issues?
    - Originality [15 pts]: Your report should not be a verbatim copy of existing findings. You should highlight the main ideas and take-aways in your own words. Good reports will provide their own high-level commentaries, recommendations, technical commentaries, philosophical questioning, and takeaways.
    - Quality [20 pts]: Does the report answer the main question? Is the evaluation well-supported and thoroughly explained?
    - Clarity [20 pts]: Is the report clearly written? Is it well organized? Does it adequately inform the reader? Do the authors effectively use the report medium, including visual aids and a more informal tone?
    - Significance [15 pts]: Is the report likely to be useful for further decision?
    - Extra credit [15 pts]: The course staff may select a few reports that go above and beyond to receive additional points.