L46 - Group Project

Key Links

- Paper: Fast and scalable in-memory deep multitask learning via neural weight virtualization | Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services
- Author's GitHub repo: https://github.com/learning1234embed/NeuralWeightVirtualization
- TwoLevelWeightPageVirtualization
- Project GitHub repo: https://github.com/K0rnel/NeuralWeightVirtualization
- Colab: TBD
- Overleaf (project report): https://www.overleaf.com/6513862572tscxxgqvrktt
- Report draft Google Doc: here

Remaining tasks

- ~~Check convergence of joint optimisation~~
- ~~Weight-Page Matching~~
  - ~~Inference accuracy for various weight-page size~~
  - ~~Matching time~~
- ~~Weight-Page Optimization~~
  - ~~Joint vs. Sequential Optimization~~
- ~~Matching Regularizer~~
- Benchmark 3, 5, 7, 10 networks.

Work Plan

- Project report:
  - Introduction - problem statement and evaluated paper (1 page) - K
  - Summary of the paper (1 - 1.5 pages) - S
  - Description of technical implementation (1 page) - K
  - Evaluation: (3 pages)
    - Replication of original results - K
      - Number of weight pages
      - Regularizer
      - Joint vs sequential optimisation
    - Extension to 10 DNNs - S
      - Convergence
      - 3,5,7,9 DNNs - accuracy + inference
      - 10 DNNs - failure probably due to RAM
      - Memory usage
        - 3 DNNs vs 10 DNNs
      - Optimisation time

    - Discussion and future work (0.5 pages) - K
    - Conclusion (0.5 pages) - S
- **Also mention:**
    - Discrepancies between the paper and the repo
        - Code not provided for the MCU.
        - How did they fit 4GB on an MCU. Did they have an optimised code version?
        - The code fails at 10 DNNs - probably RAM shortage.
        - Different numbers quoted for RAM in paper and README file - which one is correct?
        - Mention the increasing optimisation time.
        - The fact that OBS does not perform well - different Fisher information graph?

- Technical work:
    - ~~Replicate paper's results using authors' code~~
    - ~~Add additional DNN (us8k):~~
        - ~~Import DNN model and data~~
        - ~~Explore the model structure~~
            - ~~Restore the model graph~~
            - ~~Freeze the graph and export it to Netron App~~
            - ~~Modify the model_data.py and pintle.py files~~
        - ~~Train the model once to create model_weight.npy file~~
        - ~~Create weight pages~~
        - ~~Optimize weight pages~~
        - ~~Test DNN switching~~
    - ~~Add more DNNs (all DNNs included in the Two-Level virtualization repo?)~~ and evaluate network performance
    - Check the results from the baseline/in-memory execution scripts. It seems like the baseline starts slowly but then accelerates once it 'warms up'.
    - Extend the original repo's documentation? Add some routines/scripts to clean up the .npy files or to automate adding new networks without manually adjusting the code.

How to add a new DNN:

- Add meta, index and data files.
- Add pintle and adjust it.
- If necessary, adjust the number of weight pages in the weight_virtualisation.py script (currently set to 721).
- Add <dnn-name>_data.py file in the root directory.
- Add to the download script.
- Add to the joint optimization script.
- Add to the baseline and in-memory execution script.

Then, in the Colab notebook → call the `add` routine to add the new VNN (this will create the .vnn file). And execute joint optimisation (this will create the weight files).

Work Timeline

- 21-27.12:
    - Kornel:
        - Test DNN switching and inference accuracy of original network vs the one with US8K DNN added
        - ~~Produce US8K network graphs and relevant code~~
        - ~~Create a shared GitHub repo~~
        - Clean up the Colab file, create the “master” file to be included in the final report
    - Samuil:
        - Add remaining 4 DNNs
        - Test DNN switching and inference accuracy of original network vs the network with extra DNNs
- 28.12-03.01:
    - Write up the Problem statement, project methodology, implementation chapters
    - Continue technical experiments
- 04-10.01:
    - Write up the remaining project chapters
    - Finish all remaining experiments
- 11-17.01:
    - Proof-read, convert to latex and submit the project report
- 18-20.01:
    - Grab a pint at a pub :)