Graduating Member Notes

Check individually mentioned member notebooks for clarification; but here are general updates and potential future tasking.

- NNLearner

- Made switch to layer/block level primitives to expand search space from Fall 2022 Semester.
- Modeled off of the architecture of the AASCE Challenge Paper:
  - ResNet - hope to refine feature extraction using EMADE
  - DecNet - decodes the output of the ResNet portion and returns 3 outputs: hm, reg, and wh. Each has a distinct loss used for updating the neural net weights.
  - Decoder - takes the 3 outputs of DecNet and returns the landmark points.

  Generally, primitives can be found in `neural_network_methods.py`, with notable exceptions being the DecNet and Decoder layers found in `neural_network_helper.py` (in `/emade-athi/emade/src/GPFramework`)

  See `genADFPrimitives()` in `gp_framework_helper.py` for where primitives are added to the pset.
- To achieve results for 2023 Spring final presentation, we moved away from layer level granularity, but future plans are to reintroduce layer granularity. Theoretically a matter of implementation; existing model can call .fit(). Nathan and Lucas noted that calling .fit() involves some intermediate steps where additional neural net architecture is necessary.
- Unclear notes from Spring 2023 tasks can be found here (green appears to be completed)
  A quick summary:
  1. Decoder may need cobb angle calculation to be implemented inside for evaluation
  2. Unclear how implicit padding impacts EMADE
  3. MaxPool2D parameters: unclear how to implement kernel size and torch dilation in keras
  4. BatchNormalization and Relu are currently an enum as opposed to a layer primitive; would ideally like for it to be a primitive
  5. Implement conversions from Testing team to the keras implementation in EMADE
- NNLearner testing:

    - How to pack x-ray data to npz for EmadeDataPair is found in Lucas Yim's notebook
    - In GPFramework/UnitTests, you will find a `image-nn_unit_test.py` file that contains the exact test for the construction and testing of an NNLearner for images
        - You will notice that the current test in there reconstructs the AASCE model and in training, training accuracy is equal to validation accuracy in all epochs. There is clearly some issue somewhere and we suspect it is from either npz or how the data is split before fit
    - You will find code that unpacks the npz correctly and then you will find how the layers are added to the layer list and ultimately placed into the NNLearner after that under the `test_nnlearner` function

TASKS

- Want to implement formal runs via standalone.
- EMADE runs are next from fixing individual generation.

- Testing

- Much of the testing team should still be in VIP next semester so notes here will be more sparse
- Collab notebook here; a good baseline DecNet (or any neural net) testing template: `nzhong31/Vertebra-Landmark-Detection-changed/custom_dec_net_testing.py`
- At Spring 2023 conclusion, PyTorch and Keras versions were trainable and had outputs compared. The model uses three loss metrics derived from the DecNet output to update its weights.

TASKS

- It appears the hm output had the biggest discrepancy between the two; would like to reduce these difference as much as possible.
- Want to begin communicating changes to rest of the sub-subteams and implement any confirmed keras conversion in the individual.

- Preprocessing/Datasets

- Nathan implemented cropping and contrast methods in `cropping-methods.py` as well as `contrast_preprocessing.py` found in his directory `nzhong31/Vertebra-Landmark-Detection-changed`
- Landmark visualization can be found here:
- Want to begin communicating changes to rest of the sub-subteams and implement any confirmed keras conversion in the individual.

TASKS

- Nothing major; depends on Shriners help. Would like to incorporate the 700+ unused x-rays in the datastores. See RohlingML→Data→Shriners_Full_image_set→Consume. There is sample code to download the additional images. I won't include it explicitly here in case there is sensitive information at risk, but you should see a snippet that includes an import that looks like this: `from azureml.core import Workspace, Dataset`