META-LEARNING

ASSIGNMENT-2: ARC (Abstraction & Reasoning Challenge)

TANMAY SINGH
GROUP-14
2021569
CSAI
CLASS OF '25

Report Backup:

Ans 1: I used the approach discussed in one of the initial papers covered in the class, Model-Agnostic Meta-Learning, wherein I used a custom Encoder-Decoder Model with Attention Layers to take the input & generate an encoded representation along with the use of Attention layers. I later generate back the original output representation & calculated the loss using a custom loss metric.
The training pipeline included the traditional inner & outer loops of MAML, wherein the meta-update takes place in the outer loop using the gradients computed in the inner loop.
The data was also processed in a manner that mimicked few-shot training & validation of the meta-updated meta-model on the unseen tasks from the validation set.
Notebook: Link

Ans 2: There are no significant novelties in my approach. The only difference was the way I prepared the data for training. I padded the input & output grids using a value not in either of the grids to a size of 30x30 in a uniform manner, which I later flattened into a size of (900, 1) for training & validation by creating separate input & output data loaders & datasets for all three sets (training, validation & evaluation).

Ans3: NA

Ans 4: *Data Processing:*

The data from the training folder (within the arc_data folder, there are two folders, training & evaluation) was split in the ratio of 80:20, where the 80% of data was used for few-shot training of the meta-model & validation one was used to test the performance of the meta-updated meta-model.
The evaluation data was prepared from the evaluation folder within the arc_data folder following the same processing steps as above.
The data was padded with a value not in either grids to max allowed size (30x30) which was then flattened to be used as an input to the custom Encoder-Decoder model having attention layers.

Model & Evaluation:

The model used which I used for the ARC challenge was a custom Encoder-Decoder model with attention layers, that took the input (900, 1), and generated an encoded representation of it, with the use of attention layers & finally generated/decoded back the output grid.
The Training & Validation was done using a model pipeline that had the tradition outer & inner loops used in MAML, with a custom loss function where each value between the predicted & the

actual output is compared (1 for wrong & 0 for correct), and then averaged out to return the average loss.
The outer loop does the meta-update using the gradients computed in the inner-loop during few-shot training.
The training loss was logged per epoch, while the validation one was logged per batch, since the meta-model was first trained on the training set & once trained enough, it was used to generate predictions on the unseen task (samples) from the validation set.

The loss plots were plotted & gave an idea that the model was indeed training, although the number of tasks solved on the publicly available evaluation set (created using the evaluation folder within the arc_data, was 0/419).
Result: LINK

Ans 5: Although I tried a lot of approaches, but none of them solved any tasks on the publicly available evaluation set (0/419).
Other approaches that I tried were:

Using CodeIt's DSL integrated with Michael Hodel's Starter notebook that generated programs exhaustively & searched for them using search algorithms (both optimized & non-optimized).
Tried Fine-tuning a LLM like LLAMA & T5 for seq2seq task.
Tried using various attention based & deep neural network models, but none of them translated into good results.

Although none of the approaches solved any tasks on the evaluation set, I used my current approach to address the ARC challenge as it is traditional in nature (in the domain of meta-learning). Thus, I found MAML to be a good base to tackle the ARC challenge even though it did not translated into desired results on the evaluation set. On the contrary, the main idea behind sticking to this approach was that the model was indeed training as the training & validation losses seemed to change every time.
Results: LINK

Ans 6: *Link to Google Drive: LINK*

Ans 7:

For Lightning:
```
!pip install torch
!pip install numpy
!pip install matplotlib
!pip install tqdm
!pip install scikit-learn
!pip install mplcyberpunk
!pip install pandas
!pip install scipy
!pip install seaborn
!pip install pyarrow
!pip install cloudpickle
!pip install gdown
```

Ans 8: Link to Google Drive Presentation: Link