# 20220824 ARC Challenge meeting

@asteriskCat notes:

@andreaskoepf: Now looking for a teaming approach; ARC challenge will require a lot of effort. There has been some spectucular results in large language, etc., but there is just a very small amount of training information. Working on 10 or so riddles a day, and has found a large variety of riddle types, as well as wide distributions of given riddle types. Presented PowerPoint:

- Approaches
    - Blackbox
        - No explicitly program, not search, no hypothesis, no complexity measures, multiple solutions by indeterminant sampling
        - Direct Input -> Output mapping MLP CNN LSTM transformer -> LMM, NTM
        - Zero-shot generalization of riddle training
        - Magic is expected to happen inside model, strong OOD generalization questionable.
    - Explainable
        - Internal program representation, synthesis of program candidates, testing on training examples,
        - DSL functions, neural transforms, command based per-pixel interactions
        - Programs modularized into (nested) functions
        - Search: brute force, heuristic, neural-guided (via recognition functions)
        - Human interpretable solution, steerable by DSL set and recognition
- Basic algo for explainable path
    - Analyze examples, extract features, describe boards
    - While time is left and no high-confidence prog found
        - generatate xform prog cand
- Challenges
    - Space of possible progs is large
        - Use intuition of neural network -> neural guided search
    - Riddles are very diverse, number of riddles per class very low (sometimes 1)
        - Create riddle generator for all public ARC riddles that varies colors, patterns, objecti-opositions, board-sizeds orientations, conditions, ...(when possible)
    - Human priors are not clearly defined
        - Hard, no full solution known, try as many as possible, e.g. going over each ARC riddle and teach system
- First idea: text conditioned neural transformations
    - Algorithm
        - Given input board and steps learn to generate output board
        - Learn to generate tranformation sequence from a given input/output board pair
- ARC workbench environment
    - One program for development

    - Usable by agent
    - Holds whole state of current process
    - Allows "write" scripts and application as tokenized version
    - Scripts can call other variable transform frunctions of any kind, e.g. DSL, neural-network
    - System can be used to show human solutions
    - Sub-workbenches - define ccurent inputs, pass to clean environment
- Text conditioned nested GATO model,
    - agent trained in ARC environment, conditioned on what it should do
    - solutions are always text-based (script source code)
    - Training is always conditioned on function name/description, special token to "call" sub-function
    - Sub-functions can be hard0cded or also handleed by a GATO neural-network
    - Sub-functions can be trained on their own to analyze or transform riddle specific functions.

Responses:
@Avelina: What about mixtures of experts? XLNet (not causal, allows for out of order decoding). Different from BERT in that next attention can be out of order. You can predict the pixels out of order, and get the right answer at the end.

@asteriskCat: I thought about XLNet, but only for sequence size, not for these reasons.

XLNet: Generalized Autoregressive Pretraining for Language Understanding

@Avelina: we could do that too, but to me the real strength is out of order solution.

@Mohamed Osman: - this would let you perform the easy parts first. Pick the highest candidates from beam search.

@asteriskCat: I was thinking more sequence to sequence rather than LLM

@andreaskoepf: It seems like it is really hard to recognize some of these riddles. For me it is. How can we pick the best candidates? Also, LLM will have a hard time understanding geometry - you need to build it in or train it in from the beginning. It seems unlikely that a LLM can get there by fine-tuning.

@parapraxis: Has promising results for LongT5 model. Have also been exerimenting with XLNet.

@XMaster96: Also working on LLM (multimodel). Hoping pre-contained knowledge bridges gap. LLM should generalize well given the pretraining. FLAN paper had task clusters, one task had multiple similar training data. Feels that clear generalization was apparent. The larger the model, the better generalization.

> arXiv.org
> Finetuned Language Models Are Zero-Shot Learners
> This paper explores a simple method for improving the zero-shot learning abilities of

> language models. We show that instruction tuning -- finetuning language models on a collection of tasks...

@XMaster96: The multimodel can extract useful features and put them together.

@AK: I can see how appealing this approach is, given the capabilities demonstrated by LLM and ability to recognize structure. The hidden rule in the riddles - is it in the range of what LLM can recognize? For simple rules, yes, but more complicated it will be difficult.

@Mohamed Osman: Why not do what alphazero does, take i/o pair, and have it always operate on board? Search understands what can be done. Operating on boards lets you see what's going on.

@andreaskoepf:: I see the bench as being able to show what is going on in an explainable manner. Suppose I have to manipulate a board by rotating. This could be broken down into smaller atomic steps. Teaching in terms of manipulating individual pixels, rather than objects, seems difficult (but cool, as it could be neural end-to-end). There may be some transformations which are hard to code, but easier as NNs. I feel the icecuber approach is at a deadend due to search time.

@pa: There are basically 2 teams - blackbox vs explainable

@XM: Maybe split into groups now.

@0x000FF4: Training VAE to learn rotated v not rotated. Idea is to see if the latent space can represent all of these transformations, and can be used for recognition. Using "1D bottleneck" - latent space is just a vector.

@asteriskCat: Committed to explainable team. Will write up notes and drop file in ARC challenge thread.

@pa: Working with @Avelina, training autoencoder, @pa is coding BERT.
Proposed first cut algo:
- Autoencoder (3D convolution, possibly 1D x or other restrictions) is trained,
- AE encder encodes each board
- Encoded input to LM, which solves riddle at one-shot.
- Output can be reconstructed by decoder.
- Planning end-to-end training while training BERT to refine encoder and decoder.
- Future methods may approach with memory, attention mechanisms, sampling

Also working on noise injection augmentation for @Mohomed Osman's "arcmentations" framework. Usually, denoising is a standalone task, and is not composed with other tasks. Thinking about different sized speckles and other possibilities. Are there combinations of augmentations that aren't used?

@Mohomed Osman: I just thought of composing superresolution augmentation with noise augmentation. It would be better to use a pipeline, rather than just a soup of augmentations.

@pa: We could tag riddles with augmentations that would help train riddles. These would be augmentations that help the system learn riddles, as opposed to generations

Both input and output = generation?
input only = augmentation?

Meet next week, same time.