Mohamed Osman

Auto-generated and might contain hallucinations

MLST

2025-03-22

Contents

1 Show: Mohamed Osman

1.1 Executive Overview

This episode presents a comprehensive discussion of the Abstraction and Reasoning Corpus (ARC) challenge, focusing on innovative approaches to machine learning and artificial intelligence. The conversation, spanning approximately [01:03:26], primarily centers on the achievements of MindsAI team, subsequently acquired by Tufa Labs, in developing novel solutions for abstract reasoning tasks.

The discussion extensively covers the team's implementation of test-time fine-tuning, a paradigm-shifting approach that fundamentally alters how neural networks learn and adapt. Mohamed Osman, a key figure in this development, articulates how their system achieves superior performance by treating ARC primarily as a perceptual problem rather than purely computational. Their approach, achieving approximately 58% accuracy on the hidden test set, represents a significant advancement over previous methodologies.

A central technical innovation discussed is their use of a modified T5 encoder-decoder architecture, pre-trained on both language and code. This architecture enables what Osman terms "meta-modeling," where the system learns to contextualize and adapt to novel problems during the forward pass. The team deliberately avoided specialized input representations, arguing that ARC's fundamental challenge lies in maintaining flexibility in problem interpretation.

The conversation provides significant insight into the limitations of current AI architectures, particularly regarding counting and copying tasks. These limitations are attributed to representational squashing in transformer models and the inherent properties of the softmax function, suggesting fundamental architectural constraints that require addressing for further advancement.

The discussion culminates in an examination of the future of abstract reasoning challenges, including François Chollet's development of ARC version 2. Osman presents a compelling argument for maintaining access to the original ARC benchmark alongside new versions, enabling continued measurement of methodological progress.

1.2 Key Terms Glossary

1. Test-Time Fine-Tuning: Dynamic parameter adjustment during model inference phase
2. ARC (Abstraction and Reasoning Corpus): Benchmark for testing AI systems' abstract reasoning capabilities

3. Encoder-Decoder Architecture: Neural network structure processing input-output transformations
4. Representational Squashing: Compression of information in transformer networks affecting their counting ability
5. Transductive Learning: Direct prediction of outputs without explicit rule inference
6. Meta-Model: System capable of adapting its learning approach based on context
7. Forward Pass: Single processing cycle through neural network layers
8. Softmax Function: Probability distribution function in neural networks
9. T5 Model: Transformer-based language model architecture
10. Perceptual Problem: Task requiring pattern recognition and interpretation rather than explicit computation

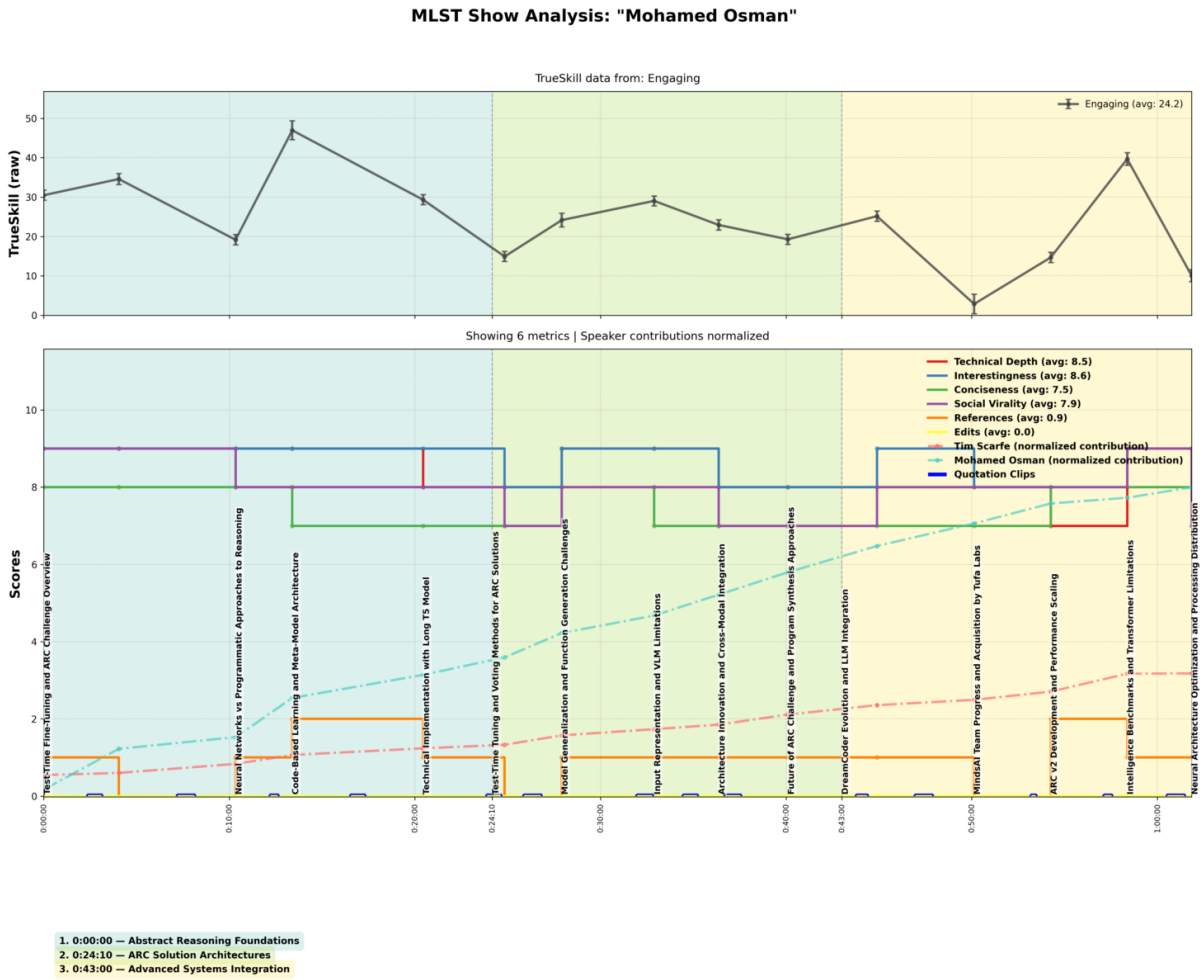

Figure 1.1: Fragment Scores Analysis

2 Technical Analysis of Mohamed + Jack's ARC paper: "Don't Throw the Baby Out with the Bathwater: How and Why Deep Learning for ARC"

2.1 Introduction

The paper "Don't Throw the Baby Out with the Bathwater: How and Why Deep Learning for ARC" by Jack Cole and Mohamed Osman from MindsAI/Tufa Labs presents a groundbreaking approach to solving the Abstraction and Reasoning Corpus (ARC) challenge using deep learning methods. This analysis breaks down their methodology, innovations, and results.

2.2 The ARC Challenge

ARC is designed to test AI systems' ability to perform abstract reasoning with minimal examples. Each task consists of: - A set of training examples (input-output grid pairs) - Test inputs to transform according to inferred patterns

What makes ARC difficult: - Requires genuine learning from few examples (2-6 typically) - Little pre-training knowledge can be leveraged - Forces systems to "figure out" transformations based solely on provided examples - Emphasizes contextual reasoning during evaluation - Conventional ML approaches and foundation models perform poorly out-of-box

2.3 Core Thesis

The authors challenge the prevailing view that neural networks cannot effectively solve ARC problems. Their central argument is that:

1. ARC problems are more "perceptual" and qualitative than quantitative
2. Deep learning excels at learning abstractions from perceptual/qualitative problems
3. The full deep learning paradigm (untrained NN + optimizer) should be incorporated in the inference process

2.4 Key Innovations

2.4.1 1. Architecture and Pre-Training Recipe

- **Base Model**: LongT5 encoder-decoder model chosen for its:
  - Extended context length (accommodates larger riddles)
  - Sequence-to-sequence capabilities
  - Non-causal attention mechanisms suitable for associative learning
- **Input Representation**: Each riddle encoded as a unified text sequence
  - Grids unrolled row-wise
  - Pixel colors represented numerically
  - Complete riddle presented together to enable cross-example reasoning
- **Direct Output Generation**: Generate output grid directly rather than producing code
  - Simpler than code generation approaches
  - Avoids additional complexity of syntactic correctness
- **Multi-Task Training**: Integrated additional tasks requiring contextualization
  - Reduces reliance on memorizing individual tasks
  - Encourages genuine contextual reasoning
- **Code Pre-Training**: Training on coding tasks provided significant performance boost
  - Code datasets demand meticulous attention to detail and context
  - Fosters better focus on hierarchical reasoning
- **Synthetic Data Generation**: Created automatic riddle generators
  - Used Johan Sokrates Wind's DSL to construct synthetic riddles
  - Carefully ensured "fairness" in generated riddles

2.4.2 2. Test-Time Fine-Tuning (TTFT)

A novel approach where the model undergoes fine-tuning during test time:

- **Methodology**:
  - Create synthetic training data from the test riddle's demonstration examples
  - Repurpose a grid pair as a new "test example"
  - Apply augmentations (color permutation, spatial transformations, shuffling)
  - Perform brief fine-tuning on these augmented riddles
  - Generate predictions for actual test grids
- **Conceptual Basis**: "Reframing" allows the model to iteratively refine its approach
  - Helps overcome initial mistakes in pattern recognition

  - Provides mechanism to adapt frames based on feedback
  - Mirrors human process of revisiting and adjusting hypotheses

- **Full Parameter Updates**: Authors chose full fine-tuning rather than lighter alternatives
  - Simplicity and reliability
  - Sufficient power to generate needed abstractions

2.4.3 3. Augment Inference Reverse-Augmentation and Vote (AIRV)

A test-time augmentation strategy:

- **Procedure**:
  - Apply spatial transformation to input riddle
  - Get predictions on transformed riddle
  - Reverse back to original orientation
  - Gather multiple predictions from different augmentations
  - Vote to select most frequent/confident output grid
- **Advantage**: Unlike beam search or temperature sampling, AIRV generates duplicate predictions
  - Enables voting to amplify strong, consistent solutions
  - Filters out noisy variants
  - Particularly effective when each riddle has only one correct answer

2.5 Results and Performance Analysis

The paper reports impressive results:

- **Zero-Shot vs. Enhanced Performance**:
  - Base LongT5 model: 5% zero-shot → 13% with AIRV → 39% with TTFT+AIRV
  - 260% increase with AIRV alone
  - Additional 300% gain with TTFT
- **Model Size Impact**:
  - Larger models performed better on Zero-Shot and AIRV-only settings
  - Indicates that model size correlates with forward pass flexibility
  - Aligns with scaling laws in reasoning benchmarks
- **Competition Success**:
  - First place in 2023 ARCathon competition
  - Highest score (58%) on ARC private test set in 2024

2.6 Comparison with Other Approaches

The paper contrasts their approach with previous methods:

- **LLM-Based Approaches**: Standard LLMs like GPT-4 achieve only 10-20% accuracy
- **Neuro-Symbolic Methods**: CodeIt (14.8%), GPAR (50% on subset)
- **Brute Force Search**: The previous winning approach (Icecuber, 2020)

Limitations of previous approaches identified: - Limited generalization - Lack of contextual reasoning - Evaluation on subsets rather than full test set - Insufficient focus on perceptual reasoning

2.7 Technical Insights

Several deeper technical insights emerge:

1. **Contextual Reasoning**: Encoding all grid-pairs together enables cross-example reasoning vs. individual processing
2. **Non-Causal Attention**: Experimental comparison showed encoder-decoder structure with non-causal attention outperformed decoder-only models
3. **Pre-Training Benefits**: Extensive ARC pre-training allows “sedimentation” of basic priors into early layers, creating “room” for test-time features to emerge
4. **Beam Search Decoding**: Maintains multiple candidate solutions, particularly beneficial for ARC where each riddle has one correct solution
5. **Balance of Pre-Training vs. Test-Time Adaptation**: Success depends on both substantial pre-training and effective test-time techniques

2.8 Implications for AI Research

The paper offers significant implications:

1. **Perceptual Reasoning**: Reframes ARC as primarily a perceptual reasoning challenge rather than a pure logical one
2. **Learning Paradigm**: Emphasizes incorporating the entire learning process (model + optimizer) in the inference loop
3. **Meta-Learning**: Shows how test-time adaptation can enable solving of novel tasks without extensive pre-training on similar problems
4. **Associative Learning**: Demonstrates the importance of mechanisms that improve broad perceptual reasoning over focusing solely on logical inference

2.9 Conclusion

This work provides strong evidence that deep learning remains crucial for advancing AI's reasoning capabilities. By fully committing to deep learning's capacity to acquire novel abstractions and incorporating on-the-fly neural network training at test time, the authors achieved state-of-the-art performance on a benchmark specifically designed to challenge AI systems' reasoning abilities.

The primary takeaway is that abandoning deep learning for abstract reasoning tasks would be premature - instead, we should focus on extending the paradigm to include the complete learning process (model + optimizer) as part of inference.

3 Best Quotations

- **Tim Scarfe** [0:02:20]: Technical breakthrough in ARC challenge: Their methodology introduces two principal techniques: test time active fine tuning where you generate synthetic training data derived from each puzzle's examples and fine tune the model as you go. The second approach is augment inference reverse votes… They found that the latter improved performance by two hundred and sixty percent with test time active fine tuning providing an additional three hundred percent improvement (Interestingness: **9**)
- **Mohamed Osman** [0:07:10]: Test-time fine-tuning paradigm: 'there is a sense that test time fine tuning is a new paradigm to deep learning… you're changing parameters at test time… but there is a way to look at it in which it exactly fits the deep learning paradigm… we really see ARC as this perceptual problem' (Interestingness: **9**)
- **Mohamed Osman** [0:12:10]: **"Neural network compositionality through bias engineering" -** 'by default, they are gonna just learn statistics. And it's not elegant because they're not they're not composable, but I think […] you can get the biases deep enough, and the biases are really important such that you're able to tune the reasoning part really easy' (Interestingness: **9**)
- **Mohamed Osman** [0:16:30]: **"Meta-learning for ARC contextualization" -** we prompt everything into the forward pass all at once… it's one way to look at it as a measure of the contextual the novel contextualization ability of the transformer (Interestingness: **9**)
- **Mohamed Osman** [0:23:50]: **"Test-time reasoning optimization" -** Everything in the input, you need to kind of learn how do I compare the input and then kind of get at a very almost okay function, but probably not okay in very important ways. That's already very, very close to your solution. So now you can test time tune and kind of search over the reasoning, tune the reasoning (Interestingness: **9**)
- **Mohamed Osman** [0:25:50]: **"Voting mechanism for ARC" -** 'there are many ways that you can be wrong about an ARC riddle, but there's only one correct solution… ARC is is very special because it's only one right way to do it right. So you can imagine all of these beams and the model is is kind of uncertain about a certain next token.' (Interestingness: **9**)
- **Mohamed Osman** [0:31:55]: **"Test-time tuning mechanism explanation" -** And then the hope is that ARC has a perceptual problem and that neural networks can learn these generalized perceptual problems if you put them in domain and if you tune the reasoning that way. (Interestingness: **9**)
- **Mohamed Osman** [0:34:25]: **"VLM fixed representation limitation" -** 'This really gets to the core of ARC. Like, if you wanna understand ARC, you need to understand that you need raw representations in into the transformer rather than, you know, pre encoding some stuff.' (Interestingness: **9**)
- **Mohamed Osman** [0:36:40]: **"Flat representation architecture proposal" -** The thing that I'm advocating for is you need a flat representation and an intermixing. So you could have vision

patches intermixed with language tokens, they all condition on each other. That's that seems reasonable. (Interestingness: **9**)

- **Mohamed Osman** [0:43:45]: **"Piaget's theory of incremental development: You can play a game without describing its rules" -** 'if I ask you to write the Python program for certain articles, it could take you up to thirty minutes. But if I actually color them in, you can do that instantly. And that gets to Piaget's theory of incremental kind of development' (Interestingness: **9**)
- **Mohamed Osman** [0:46:55]: **"Academic Impact of MLST" -** There's been papers now that outline the technical content of what we have, but we wanted to put the paper out there for people to cite the original kind of source. And people right now, by the way, are citing our MLST podcast. Yeah. Like, lots of papers. The Test Time paper by MIT, by Ikin, cites the podcast. (Interestingness: **9**)
- **Mohamed Osman** [0:53:10]: **"ARC optimization insight" -** 'You always have to target the hundred percent with ARC… it's very hard to optimize for a certain competition for something like ARC, or it's a really bad idea. It's not gonna give you lots of score because of how private the dataset is, and you can't make guesses about it.' (Interestingness: **9**)
- **Mohamed Osman** [0:57:05]: ARC v1 Scaling Claim: I do think it gets solved with scale. Like, I think ARC v one, you know, give us four thirty nineties or two thirty nineties and more time on on the problem next year. And, absolutely, we get to eighty five percent. So for ARC v one. So I like, I do think it scales, and and I do have some data on that with the hidden test set that it does scale (Interestingness: **9**)
- **Tim Scarfe** [1:00:30]: **"Transformer limitations in counting" -** They've studied this in self attention transformers. They actually say it's because of representational squashing… if you scale transformers all the way up, all of the attention gets focused on the first token. And there's also another problem with the softmax function that creates this kind of directedness… transformers, like even in a trivial sense, cannot do counting or copying. (Interestingness: **9**)
- **Mohamed Osman** [1:02:15]: **"Multi-layer vs Single-layer Processing" -** You want to do the processing as you go up the layers. If you're doing all processing in one layer, that means you're overfitting. Because you're not really running an algorithm, you're just taking features and doing it in this one layer. (Interestingness: **9**)

4 Questions

- **Mohamed Osman** [0:00:35]: How does the 'weak contextualizer' approach contribute to better performance on ARC tasks?
- **Tim Scarfe** [0:02:45]: What are the specific mechanisms behind the 260% improvement from augmented inference reverse votes?
- **Mohamed Osman** [0:07:20]: How does test-time fine-tuning maintain model stability while modifying parameters during inference?
- **Mohamed Osman** [0:08:30]: What specific perception mechanisms in the model help identify the right level of abstraction for ARC problems?
- **Mohamed Osman** [0:11:55]: What specific biases and architectural choices enable compositionality in neural networks?
- **Mohamed Osman** [0:13:05]: How does the process of perceiving the right level of abstraction differ between neural and programmatic approaches?
- **Tim Scarfe** [0:14:35]: How does the diffused activation pattern in code-based reasoning differ from other types of neural network reasoning tasks?
- **Mohamed Osman** [0:17:30]: What are the specific trade-offs between meta-model approaches and direct transformation learning in the context of ARC?
- **Mohamed Osman** [0:21:15]: How does code pretraining specifically enhance the model's contextualization abilities compared to language-only pretraining?
- **Mohamed Osman** [0:24:10]: What specific metrics or criteria are used to evaluate and optimize the reasoning during test-time tuning?
- **Mohamed Osman** [0:25:50]: What metrics are used to determine the confidence scores in the voting system?
- **Mohamed Osman** [0:26:45]: How does the beam search implementation handle backtracking when incorrect paths are detected?
- **Mohamed Osman** [0:29:25]: How exactly does the pre-training process encode prior knowledge and core reasoning patterns?
- **Mohamed Osman** [0:31:55]: What is the specific mechanism by which test-time tuning updates the implicit transformation function?
- **Mohamed Osman** [0:33:20]: What specific adversarial examples could be constructed to break specialized ARC input representations?
- **Tim Scarfe** [0:35:55]: How would test-time training specifically overcome the fixed representation limitations of VLMs in ARC tasks?
- **Mohamed Osman** [0:36:45]: How does the intermixing of vision patches with language tokens affect the model's performance on cross-modal reasoning tasks?
- **Mohamed Osman** [0:39:35]: What specific algorithmic patterns in ARC solutions could contribute to information leakage, and how can this be quantified?

- **Tim Scarfe** [0:42:45]: What specific limitations of neural networks in compositionality does Chollet identify that make him favor program space approaches?
- **Mohamed Osman** [0:44:40]: How could DreamCoder be modified to better handle perception while maintaining its program synthesis capabilities?
- **Mohamed Osman** [0:46:10]: What specific metrics and methodologies are being developed to measure transformer's innate abilities on ARC tasks?
- **Mohamed Osman** [0:47:15]: Which specific academic papers have cited the MLST podcast and how have they incorporated the test-time tuning methodology?
- **Mohamed Osman** [0:52:20]: What specific approaches is Tufa Labs considering to improve compositionality in large language models?
- **Mohamed Osman** [0:53:15]: How does the need to target 100% accuracy in ARC affect research methodology compared to other ML benchmarks?
- **Mohamed Osman** [0:56:15]: How does the creation of 'idiosyncratic riddles' in ARC v2 specifically counter the current limitations of pattern-matching approaches?
- **Mohamed Osman** [0:57:10]: What empirical evidence supports the claim that ARC v1 performance scales to 85% with additional compute?
- **Tim Scarfe** [1:00:40]: What specific mechanisms in transformer architecture cause the representational squashing phenomenon that affects counting tasks?
- **Tim Scarfe** [1:01:40]: How might transformer architectures be modified to better handle counting and copying tasks while maintaining their current strengths?
- **Mohamed Osman** [1:02:30]: What are the specific architectural patterns that best distribute processing across layers while maintaining model efficiency?
- **Mohamed Osman** [1:02:50]: How do we quantitatively measure the degree of distributed processing across layers versus single-layer processing?

5 Content

5.1 Abstract Reasoning Foundations

5.1.1 Test-Time Fine-Tuning and ARC Challenge Overview

Mohamed Osman: [0:00:00] Test time fine-tuning is a new paradigm to deep learning. It's something completely outside of the deep learning paradigm. What's the most efficient way to learn at test time? That's a very interesting question.

Tim Scarfe: [0:00:11] How did you encode the problems?

Mohamed Osman: [0:00:13] The whole point of ARC is they're going to trick you. Whatever specialization you put in the input, you can create a problem that's adversarial to that tokenization scheme or special representation scheme for problems because it's so arbitrary. The problem is so different. It's a very new problem. But the really cool thing that you get out of this is you train this model to be a very weak dumb contextualizer, maybe. But that's what you're tuning for.

Tim Scarfe: [0:00:38] Transformers, even in a trivial sense, cannot do counting or copying. They just can't do it. You count up to a hundred, and you just say, can you count these numbers up? And it just fails abysmally. What we do, which is really interesting, we

Mohamed Osman: [0:00:50] prompt everything into the forward pass all at once. We're looking for people that are interested in changing the paradigm, going into test time compute, that like working in small and nimble teams tackling really big problems.

Tim Scarfe: [0:01:10] Tufa Labs is a very, very exciting new research lab that's just started in Zurich. They are looking for amazing ML engineers to join their team. It's a very small team. If that sounds like you, go to Tufa Labs AI. You might remember when we interviewed Mohamed last year. He's part of MindsAI along with Jack Cole and Michael Hodel, a couple of legends. Of course, they were acquired by Tufa Labs. They got the highest score on the ARC challenge, about fifty-eight percent, and they just released their paper where they spill the beans on how they did it. It's called 'Don't Throw the Baby Out with the Bathwater: How and Why Deep Learning for ARC.' Now the prevailing view has been that neural networks lack the necessary capabilities for abstract reasoning tasks, and at least to some extent, they proved that wrong. Consider this particular ARC puzzle. The task requires inferring complex transformational rules from minimal examples, a challenge where vanilla LLMs like GPT get no better than around ten percent. Now, Mohamed thinks that ARC puzzles are fundamentally perceptual reasoning problems.**They incorporated the optimizer directly into the evaluation process, which allowed the overall approach to develop new abstractions during test time. And their methodology introduces two principal techniques. Number one, we've spoken about this a**

lot on the show, test time active fine tuning or what we've been referring to as Test Time Active or Transductive fine tuning, where you generate synthetic training data derived from each puzzle's examples and you fine tune the model as you go. The second approach is what they call augment inference reverse vote, where you apply transformations to input puzzles, generate predictions, reverse the transformation, and implement a voting mechanism to identify a consistent solution. They found that the latter improved performance by two hundred and sixty percent with test time active fine tuning providing an additional three hundred percent improvement, which is how they yielded the highest score on ARC, about fifty-eight percent. Now another thing they found is that the model architecture scale has a greater impact than pre-training for building new abstractions. Larger models are simply more expressive, enabling better abstractions and reasoning during inference. Now, we are counting down the days until version two of the ARC Prize. We are launching it on MLST on Monday, and I hope you're excited because we've got Francois coming over and he's going to tell us all about it. Suffice to say that all of the Frontier models are going to be going down to negligible performance on it, and I'm excited to show you. See you on Monday and enjoy the show. Mo, you've been on the show before, but basically you won the ARC challenge with Jack and Michael. Technically you didn't win because you didn't choose to share your solution, but certainly in terms of the leaderboard, you guys did the best solution. Welcome back to MLST.

\textbf{Outline: MO, JC, and MH achieved 58\% on ARC challenge using test-time fine-tuning at Tufa Labs Zurich, with TS discussing their approach and upcoming ARC Prize v2 announcement (audio: clear quality, neutral tone with increasing enthusiasm towards end)} **Points:**

- Test-time fine-tuning introduced as new paradigm outside traditional deep learning
- ARC problems designed to be adversarial to any specific tokenization scheme
- MindsAI team (MO, Jack Cole, Michael Hoddle) acquired by Tufa Labs achieved highest ARC score of 58\%
- Two key techniques: test-time active fine-tuning and augmented inference reverse votes

References Mentioned:

- Original ARC challenge paper by François Chollet introducing the Abstraction and Reasoning Corpus as a benchmark for measuring intelligence by **François Chollet** (Link)

Mohamed Osman: [0:04:02] Thank you so much for having me. It's always a pleasure, Tim. It's really nice seeing you here in beautiful Vancouver. I feel like you're kind of a guest to me in Canada here. It's really nice seeing you. Thanks.

Tim Scarfe: [0:04:15] Amazing. So congratulations on the incredible result. Tell me more.

Mohamed Osman: [0:04:20] Thank you so much. We've been obsessed with the ARC challenge for a very long time. Me and Jack have been working on it for two years as part of the same team. Michael has been working on it, and we're now joined for also two years, by the way. We've always thought that this benchmark was going to get more important and more important. And you know, this is the case now. At NeurIPS, lots more people know about the ARC challenge. There's been lots of great popularization of it happening. Really happy to see that. We continued developing our methods based

on similar philosophies that we had, and I'm really excited to dive deeper into those. And also, give you a little bit of some of our results that we're planning to share very soon in a paper.

Tim Scarfe: [0:05:23] Very cool. So by the time this goes out, you probably would have released that paper, so you can probably tell us some of the headlines.

Mohamed Osman: [0:05:30] The paper is not going to be super interesting technically because we've shared our ideas before. We've shared them on MLST, and we're always like sharing. And if you look at the top ten current leaderboard on ARC, maybe close to eighty percent were using similar ideas, very similar ideas in certain cases. So we were always happy to share. And we've been pretty open about that. We're happy to see everyone converging on test time fine tuning, and the voting mechanism that we found to be super useful.

Tim Scarfe: [0:06:20] Very cool. Well, why don't we break down what are the key innovations? And maybe you can't talk about all of them, but what are some of the key innovations that led to your win?

Mohamed Osman: [0:06:27] I'll preface this by saying, lots of the current methods are using very similar things. So it's like - I'm very happy to see that everyone has kind of leveled up. I'd be remiss to say it's not because of the amount of sharing we were doing. So I'm very happy to see that, and I'm happy to see what new innovations are going to come up now that we've all leveled up here. Let's break it down. The innovation - I think we should start with test time fine tuning because it's really the big one. I like framing it in this way.**So just to get the general idea - there is a sense that test time fine tuning is a new paradigm to deep learning. It's something that's completely outside of the deep learning paradigm. You're changing parameters at test time. That's a fair thing to say. But there is a way to look at it in which it exactly fits the deep learning paradigm. That's how we arrived at it. The idea is we really see ARC as this perceptual problem. You have an interpretation problem. It's kind of subjective because you have these biases. You're looking at the problem, and you have an almost infinite set of possible transformations from inputs to outputs. It's really hard to narrow down what you should look at. It's really hard - imagine a riddle where there is a box, and inside it there is another object, and there are different types of objects.** It's really hard to find the right level of representation to even start searching around the solutions. So you have to take it all at once and hope that something pops up. It's very similar to looking at an image. There is an almost infinite set of different colors and different lightings, and they're all the same thing, but you have to contend with this infinity before you can proceed to abstract over the image itself to say, 'okay, now there are four apples' and start counting. You first have to identify the apple in infinite lighting, infinite coloring. And that perception part is really important in images, but we think it's really important in ARC also. Once you have the right level of abstraction, search becomes much easier. Searching for the right function is very easy if I know this is the relevant object I need to count or do something around this specific object or this object-to-object mapping. So it's very important to get the right level of perception. So what's the best way that we know of to tackle a novel perceptual problem? It's the deep learning paradigm. If you want to gain a new skill, a perceptual skill, want to classify mugs, you start with an untrained network and train it on a bunch of images of these mugs. And so we take that idea but apply it on ARC. The claim with ARC is completely novel at test time examples. So the logical thing to do - and you have to really take seriously this idea that ARC is one of the most important or very difficult parts, if not the most difficult, is this perception problem.

So then you just apply that paradigm but in test time. And it's known to acquire skills really well. So you apply the whole paradigm to acquire skills at test time.

Outline: MO discusses with TS their team's success on ARC challenge, explaining their test-time fine-tuning approach and perceptual learning paradigm; audio shows high engagement and clarity with minimal background noise. Points:

- Team has been working on ARC challenge for two years with Jack and Michael
- ~{}80\% of top ARC leaderboard competitors using similar ideas including test-time fine-tuning and voting mechanism
- Test-time fine-tuning represents new paradigm outside traditional deep learning by changing parameters at test time
- ARC viewed as perceptual problem with infinite possible transformations requiring right level of abstraction

5.1.2 Neural Networks vs Programmatic Approaches to Reasoning

Tim Scarfe: [0:10:20] Right. So that's certainly true. There are things that neural networks can do that we cannot write programs to do. Just because they are - Chollet kind of distinguishes between perceptual problems and like type problems. Many of those problems are perceptual problems. One really interesting thing though is that you guys, counter to people like Kevin Ellis, you are fans of solution space prediction. So you don't create intermediate Python functions. Is that the case?

Mohamed Osman: [0:10:53] Yes, that's true.

Tim Scarfe: [0:10:54] So that's a pretty huge strategy right there. I have this intuition, and Chollet does, that there's something special about Python programs. And the special thing is mostly that they have this kind of compositionality, which means they can be composed together. They can be decomposed into small parts. You can construct a library, and you can take bits together and so on. And we intuitively feel that neural networks, for whatever reason, they don't have this compositionality. Maybe we can make neural networks that do in the future, but right now they don't. So people who are really bullish about neural networks, and certainly I think you guys are and Tufa is as well, think that no, actually they can do this kind of compositionality, they just need to be coaxed in the right way.

Mohamed Osman: [0:11:41] That's a very poignant point. It's absolutely true that neural networks by default are not compositional, and you need to do a lot of work to get them to be compositional. That's not easy to do. We did do that work, and it took us a lot of time to really understand that point - that by default, they're going to just learn statistics. **And it's not elegant because they're not composable, but I think - and I will speak to this later - that for a certain domain, you can get the biases deep enough, and the biases are really important such that you're able to tune the reasoning part really easily. And in that case, in that sense, you do get compositionality, but it's not an elegant solution. Without the test time aspect, without putting the biases deep enough, you don't get that.** So now what do you do and how do you do that more efficiently? These are all things that we want to explore in the future, and we're going to do that at Tufa Labs. With the Python programs,

there's something again that's really important. You can perceive the problem - if you perceive it right, you can take action in Python or in the neural space. I think that's fine as long as you see the right level of abstraction. You have that dynamism inside the model to be able to do some correlations from input to output, then match those correlations and enforce them in the other input to output, and then make sure it generalizes. After that, you can either output in Python or in a direct output manner.

Outline: TS and MO discuss neural networks' capabilities vs Python programs, focusing on compositionality, with MO explaining how neural networks can achieve compositionality through careful bias engineering at Tufa Labs; audio quality good with neutral tone and consistent prosody. Points:

- Discusses Chollet's distinction between perceptual and type-2 problems in neural networks
- Highlights special compositional nature of Python programs
- Explains how neural networks can achieve compositionality through careful bias engineering

References Mentioned:

- Kevin Ellis's work on DreamCoder, focusing on program synthesis and neural guidance for program search by **Kevin Ellis et al.** (Link)

5.1.3 Code-Based Learning and Meta-Model Architecture

Tim Scarfe: [0:13:23] Well, let me press on this a little bit because I think this is potentially the most exciting part. So we have an intuition that Python programs are compositional. And I interviewed Laura Ruess the other day. She's working at Cohere, and she did this paper showing that basically you train influence functions on neural networks, and then you get them to do fact retrieval, or you get them to do reasoning tasks, and you can see how much of the source documents light up essentially based on the task. So when doing reasoning tasks, it had a very diffused activation in terms of the source documents. And what she noticed is that it was looking at code. So when calculating the slopes of lines and so on, it was looking at things on Stack Overflow, and it was looking at procedures of how to perform some reasoning, and it was applying that in a very diffused way to a specific problem. The fascinating thing is that even though it's not explicitly generating code, you can fine-tune it on code, and code has some form of compositionality. You can coax the neural network to do an approximate form of compositionality even though you're doing solution space prediction. That is fascinating.

Mohamed Osman: [0:14:41] One of the things that we do for pre- training - with code, I think there's something really interesting about what you're saying about code. And by the way, I know about that work. It's really amazing work. With code, it's really hard to predict the next token. You have to contextualize a lot more. You have to know what we're doing, what is the exact name of the variable, what is the exact relevant variable here. With language, that's not the case. With language, you can shortcut. You can cheat easily. You can use an inaccurate word, and that'll probably be fine. With

code, you have to be very precise, so you have to be very contextual. And I think that's another way of looking at the influence of code. Code pre-training has been shown to improve reasoning across many domains in a few new papers now. So this has been reinforced. Training on code is really interesting. I also want to kind of go to this idea of tuning the reasoning. You mentioned that the neural networks, while they're solving this reasoning task, they're looking at code. I think framing the problem also matters in a very important way here. So first of all, we established that test-time training gives you the generalization ability. So what's the best way to, what's the most efficient way to learn at test time?**That's a very interesting question. And so what we do is we prompt everything in context. We're just speaking about contextualization ability, and that's a really important thing for us to maximize for ARC. Because if you're prompting things into ARC, like in the ARC format, you need the model to be as dynamic as possible. So you have your inputs and outputs and inputs and outputs, and now I have this test input. And the problem is so different. It's a very new problem. So contextualization and the ability of the model to be steered is very important. What we do, which is really interesting, we prompt everything into the forward pass all at once. So inputs and outputs, input and outputs, input, and then test input, like new input, is all in the forward pass at once. It's one way to look at it as a measure of the novel contextualization ability of the transformer.** The really cool thing that you get out of this is you train this model to be a very weak, dumb contextualizer maybe. It doesn't - the modeling ability in the forward pass is not that good, but that's what you're tuning for. It's kind of like a meta model in a sense. That's what you're pretraining for. And so the model is going to learn reasoning patterns. It's not going to learn exact transformations like Clem's work. So Clem's work is where the function learns just the transformation. But our model, by putting everything in the forward pass and training it over many different ARC riddles, you're kind of saying, okay, the thing to learn here is this meta task of looking at the context and then generalizing from it or doing your best at modeling from it. And so you have this weak meta model, and now that's much easier to tune than a Clem-type model where for Clem's model, the encoder only receives one pair of input to output. You get stronger induction or regularization - I don't know what you want to call that - but with that model, there are many possible - you don't get that meta ability. There are many possible sets of transformations given only one instance. You need two or three instances to kind of narrow that infinity down. So you're training this meta model, and now tuning that meta model is a much easier task. That's really interesting. You kind of have a predictive model that's going to be kind of wrong. You can just tune tiny pieces of it to get the reasoning to click. That's really key. And it's work that is not super present in the literature. I didn't survey too well, but there was one interesting paper, the MLC paper by Brandon Lake in Nature, which was looking at reasoning. They don't do test-time training or anything like that, but implicitly, there's a way to look at it where it's exactly this, actually. It's exactly doing what we're doing. So for that paper, what they do is they have this input to output kind of test that they want to learn. They retrieve similar things, put them in context, and then they put the new thing, and that's their forward pass. And then they train over that, and that helps them do much better. They're kind of training over that. So that's kind of test-time tuning if you take that whole paradigm and put it at test time. So we train over that, but at test time, and it's the framing of the model as a meta model. And now the tunings that you have to do are much smaller. So I think that's maybe one of the biggest differences between our work and Clem's.

Outline: TS and MO discuss the role of code in neural networks' reasoning capabilities, with MO explaining their test-time training approach for ARC using context-aware meta-models, contrasting with Clement Bonnet's work on latent program networks - audio quality excellent with consistent engagement. Points:

- Laura Reuss's research shows neural networks use diffused activation patterns when performing reasoning tasks, often leveraging code examples from sources like Stack Overflow
- Code requires more precise contextual understanding compared to natural language, making it valuable for training reasoning capabilities
- Their ARC approach uses comprehensive context in the forward pass, treating the model as a meta-learner for novel contextualization
- Contrasts their meta-model approach with Clement Bonnet's work, which focuses on single input-output pair transformations

References Mentioned:

- Laura Ruis et al. investigate how LLMs use pretraining data for reasoning tasks, finding that procedural knowledge in pretraining drives reasoning, unlike retrieval, synthesizing strategies from documents demonstrating similar reasoning. by **Laura Ruis** (Link)
- Clement Bonnet's work on Latent Program Networks (LPN) for program induction, published in 'Searching Latent Program Spaces' (2024) by **Clement Bonnet** (Link)

5.1.4 Technical Implementation with Long T5 Model

Tim Scarfe: [0:20:26] Can you just summarize that again, just so the audience understands it? So you said that you're doing more kind of training of the model before you do anything at test time. Can you explain that?

Mohamed Osman: [0:20:37] Yeah, so we do pre-train the models on ARC.

Tim Scarfe: [0:20:40] So you start with a language model, you know, like - I don't know what you use, maybe LLaMA or something like that.

Mohamed Osman: [0:20:46] T5 variant, long T5. So you start with a T5 language model.

Tim Scarfe: [0:20:50] Yeah. And what kind of language model? It's not a normal, like, autoregressive language model?

Mohamed Osman: [0:20:54] It's an encoder-decoder. So that's yeah - it's an encoder-decoder, and it's pre-trained. So we don't train it from scratch. Pre-trained on language, which we think has that contextualization aspect a little more. And we also train it on code further, so then it also gets stronger emphasis on this contextualization ability. You want to train the best kind of forward pass steerable model - basically a dynamic model that can change a lot based on its inputs. And that is possible with Transformers, but only to a certain extent. So that's where we start with.

Tim Scarfe: [0:21:34] Even that is very interesting though, because no one else has done that. So you're starting with an antique model from 2020, which is an encoder-decoder, which is something that doesn't really exist anymore. I mean, those were originally used for like machine translation tasks years ago.

Mohamed Osman: [0:21:51] Yeah.

Tim Scarfe: [0:21:51] And it's presumably trained on a tiny, tiny corpus compared to modern models. It might be trained on like, I don't know, like less than a billion tokens probably.

Mohamed Osman: [0:22:02] Yeah.

Tim Scarfe: [0:22:03] So you start with that. But the benefit of it is, I'm guessing it's a tiny model. It's like what? In the millions of parameters, like three hundred and forty million parameters?

Mohamed Osman: [0:22:10] Exactly, yeah.

Tim Scarfe: [0:22:11] So it's a tiny model.

Mohamed Osman: [0:22:12] Yeah.

Tim Scarfe: [0:22:13] You start with that, then what?

Mohamed Osman: [0:22:14] So then we have this pre-training recipe that involves code. It involves auto-generated ARC tasks in various ways. And Jack did lots of really good work there, Jack Cole. So there's this magic recipe, but the bottom line is it has lots of code. It has lots of synthetic tasks. By the way, the amount - like the total number of new tasks is not big. It's not like we are able to sit down and generate so many synthetic riddle generators. The total number of new concepts in pre-training is low, but there's an important thing that we think is happening during pre-training even with the few number of concepts trained. But we have a recipe that has some code, and it has some synthetic tasks. And then we get our pre-trained ARC model.

Tim Scarfe: [0:23:20] Okay.

Mohamed Osman: [0:23:21] Now we go into the test time where we feed everything into the forward pass, like we were doing in pre-training. And then all of the instances are in the forward pass, and then we get the test input - the new input that we need to predict the output for. That's also in the forward pass, and we need to predict the output. And what I was saying about tuning - the reasoning is the framing of the problem is really important. **Everything in the input, you need to learn how to compare the input and then get at a very almost-okay function, but probably not okay in very important ways. That's already very, very close to your solution. So now you can test-time tune and search over the reasoning, tune the reasoning, rather than a different way of looking at it. And then okay - so first of all, that compares to the MLC work. It compares to lots of ideas where when you frame the problem in a meta way, you give the model more to learn. It kind of is always the best thing to do in the forward pass, and then you have to scaffold and tune and do everything else.** But you're halfway there with that step. And so the tuning is much less work at that point.

Outline: MO describes their ARC solution approach using Long T5 encoder-decoder model (340M params) with code pretraining and synthetic tasks, followed by test-time tuning - audio quality good with consistent neutral tone and clear articulation. Points:

- Team uses Long T5 variant as base encoder-decoder model
- Model pretrained on language and code to enhance contextualization capabilities
- Pretraining recipe includes code and synthetic tasks with few total concepts
- Test-time tuning focuses on reasoning optimization with all instances in forward pass

References Mentioned:

- T5 (Text-to-Text Transfer Transformer) - Original paper introducing T5 architecture and framework for transfer learning in NLP. The model converts all NLP tasks into a text-to-text format. by **Colin Raffel et al.** (Link)

5.2 ARC Solution Architectures

5.2.1 Test-Time Tuning and Voting Methods for ARC Solutions

Tim Scarfe: [0:24:50] Okay. At test time, so you call it the forward pass, so you put the test instances in there. You represent it in a really clever way that helps the language model know, the seek-to-seek model do its thing, and then you do some green blatting for want of a better term. You do lots of sampling until you get ones that fit all of the specifications. Do you do any augmentation in the forward pass?

Mohamed Osman: [0:25:13] Yes, so the forward pass is just like prompting the model with the representation of the board. And the representation isn't super special. It's just a plain representation. So just a plain representation into the model. We don't do any filtering, if that's what you mean by green blatting. It's a very transductive process - you just produce the output directly. So we can't do any filtering actually.

Tim Scarfe: [0:25:37] Right, but do you do loads and loads of sampling until you get ones that you think are good?

Mohamed Osman: [0:25:43] Yes. So we have a method of voting that we also introduce in the paper.**And voting, I think, is especially suited to ARC. So it follows this idea of there are many ways that you can be wrong about an ARC riddle, but there's only one correct solution. So you try all of these ways and then hope that the majority vote is going to go to the only right way to do it.** So, yes, we do that.

Tim Scarfe: [0:26:15] **And tell me more about the voting.**

Mohamed Osman: [0:26:16] I think that summarizes it pretty well.**We do augmentation. So there are many ways to sample things out of a model. One way to sample something out of a model is beam search, right? Or you could do temperature-based sampling, which I don't think is a**

good idea for ARC because if you just get bad RNG, you're kind of done for. But beam search is really interesting because ARC is very special because there's only one right way to do it. So you can imagine all of these beams and the model is kind of uncertain about a certain next token. That's totally fair. So we're going to take a beam, we're going to try all of them. And then iteratively, if you've made the wrong decision, you're going to be kind of lost. Like in the pixel space, if you're outputting and you've made the wrong decision, you're going to be very lost because you don't know - the mistake is already done. Obviously, models can't backtrack, so how do you really continue after a mistake? It's unclear. Things are ambiguous so the probabilities are going to disperse, and then these beams are going to fall. But if you've done the right thing, you're going to be more and more certain of the next token. So that's one aspect of how we get the samples, but we have a variety of ways that we get samples for voting. We go into it more in the paper that will hopefully be out. And that's the voting.

Outline: MO explains to TS the test-time voting method for ARC solutions using beam search, emphasizing single correct solutions versus multiple wrong possibilities, with high audio clarity (8/10) and neutral, engaged technical discussion. Points:

- Forward pass involves simple board representation input to model
- Voting method exploits ARC's single-correct-solution nature
- Beam search preferred over temperature sampling for ARC

5.2.2 Model Generalization and Function Generation Challenges

Tim Scarfe: [0:27:54] That was the sampling. How do you do the voting?

Mohamed Osman: [0:27:56] So it's just majority vote.

Tim Scarfe: [0:27:58] Okay. And you said that there's only one way of getting it right, but aren't there uncountably many ways of getting it right?

Mohamed Osman: [0:28:06] Well, there's only one way with the human priors and with the ARC core knowledge of getting it right, hopefully. Otherwise, it's an ill-designed riddle. Like, for an ARC riddle, there's only one correct answer officially. But of course, you can have different systems that have different biases that think this is the right way. But hopefully it's very clear for the human biases and the core knowledge priors that there's only one way of getting it right.

Tim Scarfe: [0:28:39] We should explore that. The ridiculous example is I could have a program which says, if you see this specification, then give this answer. If you see this specification, then give this answer, and if you see the third one, then give this answer. So it's just explicitly encoding the answers without any generalization. And you're making the argument that because we have a certain structure of priors that can be composed in a certain way, you think that it's very unlikely that it will find a solution other than the ones which humans would agree on in that compositional space?

Mohamed Osman: [0:29:18] That's what we hope to do with the training. So with the pre-training, that's what we hope to encode. We hope to encode the priors really well. And I really want to get into prior encoding also, and what that means because there's lots of theorizing about that we get, and

there's lots of people saying, okay, you're memorizing certain reasoning patterns. So let's pin that for later. But that's what we're hoping to do with the pre-training. We're hoping to get the right priors and the right kind of core knowledge into the model in such a way where it only outputs the resolution. But the second thing we want to do is make the model steerable for a new riddle to not rely on the priors too much and to be able to produce the right solution that generalizes. So it's this very difficult balance. You want the model to be contextual but also have the priors very deeply there to be able to use them to search.

Tim Scarfe: [0:30:22] So there's a trade-off between flexibility and correctness. The reason I asked this is in Wending and Kevin's paper, they said that because they did an ensemble with induction and transduction, but they favored the induction - and I say transduction, they're both transduction - but they had a nine percent false positive rate on the explicit function generation, which meant that it was creating functions which gave the correct answer but was actually wrong. So it worked on the test specification but was actually wrong.

Mohamed Osman: [0:30:50] Yeah.

Tim Scarfe: [0:30:51] Have you seen that?

Mohamed Osman: [0:30:52] We don't output programs, so we haven't seen that, but you can have transductive - in my definition of transduction - like, you can have intra-riddle transductive Python programs. Meaning, just like you said earlier, if this is the question, give me this answer. If this is the input board, give me this answer. You can have that with Python programs. You can have overfit Python programs which just means that they didn't get the right prior from this one and this one and this one to be able to generalize to new ones.

Tim Scarfe: [0:31:25] How much of an issue was that for you guys?

Mohamed Osman: [0:31:30] We kind of just sidestepped this whole thing and found different ways of giving feedback to the model - i.e., test-time fine-tuning to hope that the built-in model has the best chance to generalize to the new test riddle.**So basically, the answer is everything we do at test time. We test-time tune, which means you start with the initial guess from the model based on the ICL, the in-context learning or contextualization ability of the model, and then you tune that. What's happening is there is an implicit model inside - there's an implicit kind of transformation function inside the model that is getting tuned. That implicit model is giving a guess, and then the guess is incorrect in maybe major, maybe minor ways. It's getting the feedback through gradient descent, updating that implicit model through the weights, and then giving another guess and another guess. So that's test-time tuning. And then the hope is that ARC has a perceptual problem and that neural networks can learn these generalized perceptual problems if you put them in domain and if you tune the reasoning that way.** So that's how we deal with it.

\textbf{Outline: TS and MO discuss ARC model generalization, test-time tuning, and implicit model transformation, referencing Ellis et al.'s work on 9\% false positive rate in function generation - audio quality good with minimal background hum, speakers maintain consistent engagement.} **Points:**

- Questions whether there are uncountably many ways of getting correct answers in the ARC challenge
- Illustrates memorization approach: explicitly encoding answers without generalization
- Explains goal of encoding priors through pre-training while maintaining model steerability
- References Ellis paper's finding of 9\% false positive rate in function generation
- Describes test-time tuning approach to improve model generalization

References Mentioned:

- Ellis et al. (2024) 'Combining Induction and Transduction for Abstract Reasoning' - Paper discusses ensemble approach combining inductive and transductive methods for ARC, noting 9\% false positive rate in function generation by **Wen-Ding Li, Kevin Ellis et al.** (Link)

5.2.3 Input Representation and VLM Limitations

Tim Scarfe: [0:32:53] How did you encode the problems?

Mohamed Osman: [0:32:55] It's a very plain, like numbers as text. So there's nothing special there.

Tim Scarfe: [0:33:00] Okay.

Mohamed Osman: [0:33:01] Absolutely nothing special. This is a very interesting question. You shouldn't, in my opinion, you shouldn't generate any special representation for ARC. The whole point of ARC is they're going to trick you. Whatever specialization you put in the input, you can create a problem that's adversarial to that tokenization scheme, or representation scheme for ARC problems because it's so arbitrary. The main point of ARC is raw. You get the problem raw, and then be flexible about combining it because it's going to be novel, and it's going to be raw. So this is another thing - VLMs are really bad for ARC. Maybe this is giving a lot of alpha to people, but I think for a big majority of people, the first thing they try is 'oh yeah, we need this - this is a visual problem. Let's use a VLM.' But that's a really bad idea. And that gets to some of the stuff I was seeing at NeurIPS, which is people are coming up with lots of datasets where VLMs do really bad, like the six-finger problem. If you have a hand with six fingers, you give it to a VLM, ask it how many fingers, it'll tell you five. Why? Because the VLM is a fixed representation machine. This really gets to the core of ARC. If you want to understand ARC, you need to understand that you need raw representations into the transformer rather than pre-encoding some stuff.**So the fixed frozen encoder in most VLMs, it will take some kind of perspective, preexisting perspective on the ARC problem, but you haven't seen the rest of the input-output pairs yet. You don't know what the right framing is. You're doing that, you're guessing, and there are many ways that you could be wrong. It's the same thing with the six fingers. With it, the VLM only knows certain things, and it's going to give you in its output tokens**

a representation of only that framing on certain things, like whether it's a hand - it's not flexible enough to be able to recombine. It doesn't know what the question is also. With a VLM, you already encoded the image before asking the question 'I need to count the fingers in the image' so that the transformer model and the VLM encoder can condition on that.

Tim Scarfe: [0:35:30] Couple of things on that. There's the notion of - I mean, Greenblatt did GPT-4 vision model, but I'm not entirely sure how that works. But the best way to do a visual model is to have it multimodal. So it's a language model and it's a vision model all in one, and it can do some kind of crossover between them. So there's the notion of whether having a vision model actually helps with ARC, or whether it's better to think of it as a reasoning problem, and whether to skip the interpretation part and give it a structured representation. Maybe we should start with that. Do you think in principle whether we had a strong multimodal model, which could transfer between the reasoning domain and the visual perception domain, it would work? Or the other point of your statement was that it's not flexible enough, but could we not do test-time training and all of this transduction stuff with a VLM as well?

Outline: MO discusses ARC problem encoding with TS, emphasizing raw representation over specialized encodings, and critiques VLMs' limitations for ARC due to fixed representations, followed by discussion about multimodal models and test-time training - Audio quality good with neutral tone and consistent prosody. Points:

- ARC problems are encoded as plain numbers in text format without special representation
- Specialized input representations for ARC are vulnerable to adversarial problems due to the arbitrary nature of tasks
- VLMs perform poorly on ARC due to fixed representations that prevent flexible recombination of concepts
- Suggests multimodal models might better handle vision-reasoning crossover in ARC

References Mentioned:

- The 'six finger problem' where VLMs consistently miscount six fingers as five, demonstrating their fixed representation limitations by **Chen et al.** (Link)

5.2.4 Architecture Innovation and Cross-Modal Integration

Mohamed Osman: [0:36:21] Yeah, you could. I just don't think it's elegant. I think it could get you to like 60% or whatever, but it's not going to get you to 100% of the way there unless you change the architecture. So I'm making a statement about a specific architecture just to be clear. Multimodality might help.**The thing that I'm advocating for is you need a flat representation and an intermixing. So you could have vision patches intermixed with language tokens, they all condition on each other. That seems reasonable. But if you have a pre encoding that is frozen and you don't test-tune, that's just a bad idea. If you do test-time tune, maybe that works. But it's just a crutch. It's not reliable. It's better to just really focus on contextualization and on being flexible because that's the point of ARC. And it's a very, very good benchmark because of that, because it found a**

way to test that. And I think one thing that's really good about our approach is there's a very different way of seeing it. Like, if you want to measure the contextualization ability, the true contextualization ability of these models, of these architectures, then one thing you could do is just prompt it in the forward pass and see if it's able to solve it. So that's a really cool way of attempting to solve ARC. Obviously, that's what we're doing. And that allows you to ask certain research questions, and I'm really excited to answer at Tufa Labs. So it's not just about solving ARC. You could start framing things about measuring what transformers are doing or the contextualization ability. I hope more people start doing that as well.

Tim Scarfe: [0:38:04] One of the really cool things about test-time compute - I don't know if you saw Hugging Face have just released this kind of like O type thing, and they're showing that they can make a one billion parameter LLaMA solve the same types of math problems as a vanilla eight billion LLaMA model. So what it seems to do is it gives you this ability to make a small model behave as if it's a big model. And that seems to be a superpower.

Mohamed Osman: [0:38:31] I am not familiar with that release that Hugging Face did, so I can't comment on it. I can give you some more details about how our models scale on the hidden test set. I think that's really interesting, something people really want to know. So what do the scaling laws look like for the hidden test set that is just completely uncontaminated, right? By the way, the public- Not completely. The hidden test set?

Tim Scarfe: [0:39:07] Well, yeah, I think there is a potential objection that because we have now hit the hidden test set a lot of times, that there's some unintentional information leakage. Could you comment on that?

Mohamed Osman: [0:39:21] I think the information leakage, from an information theory perspective, is very, very low. But I'll tell you what we are trying to do with the information leakage. We're trying to make it algorithmic. Right? Like, there's information leakage about the type of things that work on ARC. And that was kind of the hope, but I don't see it as significant at all, honestly. The ARC dataset was actually available for a hundred submissions a day for like a couple of years, and people didn't seem to gain too much from that. We didn't gain too much from that. We weren't honestly using it that much. I think it's very minimal.

Outline: MO discusses ARC model architecture challenges with TS, emphasizing the importance of flat representation and intermixing of vision patches with language tokens, while debating test-time tuning efficacy and addressing concerns about hidden test set information leakage. Points:

- Argues that multimodality alone can achieve ~{}60\% performance but requires architectural changes for full effectiveness
- Advocates for flat representation with intermixed vision patches and language tokens
- References Hugging Face's O(1) development showing 1B parameter model matching 8B model performance
- Addresses information leakage in ARC hidden test set, arguing it's algorithmically minimal

References Mentioned:

- DeepSeek-R1-Distill-Llama research showing performance comparable to OpenAI-o1 across math, code, and reasoning tasks with smaller parameter models by **DeepSeek AI** (Link)

5.2.5 Future of ARC Challenge and Program Synthesis Approaches

Tim Scarfe: [0:40:05] Yeah. Because, Cholet thinks that the reason he needs to make the next version of the ARC challenge is that if it's not overfit now, it it kind of is, but just in a latent way. So he says that when you look at the ensemble of all of the various different approaches to work, even back in twenty twenty, you know, the ensemble was getting about forty nine percent. So he said that, you know, if you did a targeted, you know, imagine we're like security research. Is that that's what they do? They they like, they they look for sources of entropy, and they mix things together in a targeted way, and and so he said even though it hadn't really been attacked in this way yet, it's only a matter of time until it was.

Mohamed Osman: [0:40:43] Yeah. Yeah. I mean, I think that's fine. I think it's it's good to refresh the dataset and also maybe to remove the the brute forceable one. So yeah. I mean, when I I spoke to Francois a little bit, and he mentioned that even in in his talk, He also mentioned this the aspect of, hey. I'm gonna find out the brute forceable ones. Right? Because that was the main approach in twenty twenty, and and I'm gonna remove those. I think that could be a good thing that you do. I do think the the domain is so huge and the the possible variation in the are so big that it's totally fine to to kind of use it. But maybe yeah. Like, again, from an information tier perspective, the bits are not that much. But, I mean, yeah, who knows? I'm totally fine with a new ARC dataset that is harder. I think we're up for the challenge. The refreshment is good. The the the guys, I spoke to them a little bit, they're doing really good calibration with human testing to make sure that, you know, the datasets are well calibrated. So, it's it's gonna be fun.

Tim Scarfe: [0:41:56] Chole is quite bearish about some of the the sort of the test time compute strategies that that are being used. And maybe could you just reflect on why that is and what you think he would rather people did?

Mohamed Osman: [0:42:11] Yeah. He spoke about that a little in his talk. So he's bearish on test time compute or the specific strategies that people are using for test time compute?

Tim Scarfe: [0:42:20] I think both. And I asked him, like, what you do? Because he's doing his own startup now. And he has advocated almost for something that resembles the original DreamCoder. So he thinks that we should have, so first of all, he's a big program space guy. So he doesn't, he thinks that there are inherent limits in compositionality and whatnot using neural networks. We should do the program space thing, and whether it's a DSL or an actual program language, I can't remember exactly what it said, let's say programming language. And he says that we should have a neurally guided search, which is what the original Dreamcoder thing Yeah.

Mohamed Osman: [0:42:57] Reason I don't like Dreamcoder type approaches is not because of DreamCoder specifically. And I spoke to, like, even Kevin Ellis about this, and they like, I it's it's interesting their current approach, right, when when you consider that that he was the first author there. But so I think there are two kind of things that you have to focus on, especially when talking about DreamCoder. The the output space, that's what Kevin Ellis mentioned. It's it's too restrictive. Right? This lambda calculus, it's not it's it's very inflexible. Right? And and that's something also that I think, like, in my opinion, generating Python programs is hard.**If if I ask you to write the Python program for certain articles, it could take you up to, like, thirty minutes. Even if you are a good programmer, maybe ten minutes to thirty minutes. But if I actually color them in, you can do that instantly. And that gets to, like, Piaget's theory of, like, incremental kind of development. So you can be in a game without being able to perfectly describe the rules of the game, but you can act in the game. So you can color the things. You can represent the thing and and act in it, but the level of being able to fully specify the rules of the game and write them down, you don't have to have that. That's a higher level according to PSG. So so that that exactly, like, hits it home to me.** So that's the first problem with DreamCode. The second problem is it just doesn't focus on the perception. Right? It could, and if it did, maybe it would do well, but it doesn't. And I think the main thing should be, okay, how are you tackling being flexible in the perception space, and then you can go on to do whatever you like to learn programs in the sleep phase and and compose them, and all of that stuff is fine. But first the first question is that.

Outline: TS and MO discuss Chollet's plans for new ARC challenge, DreamCoder's limitations, and disagreements on program synthesis approaches with neutral tone and consistent audio quality. Points:

- Chollet believes ARC challenge needs updating due to latent overfitting, with ensemble approaches already reaching 49\% in 2020
- Confirms discussion with Chollet about removing brute-forceable problems from ARC dataset
- Critiques DreamCoder's restrictive lambda calculus output space and lack of focus on perception

References Mentioned:

- François Chollet's ARC Prize 2024 Technical Report discussing the limitations of ARC-AGI-1 dataset and the need for updates by **François Chollet et al.** (Link)

5.3 Advanced Systems Integration

5.3.1 DreamCoder Evolution and LLM Integration

Tim Scarfe: [0:44:53] So that's exactly right, and what's also interesting is that Kevin has abandoned using DSLs. So he has, with open arms, embraced language models, and why is that? Well it's because, as you were just saying, Python code is Turing complete. You can represent any concept. It's incredibly flexible. The reason why he didn't generate Python code in the first place was because there was this intractable search problem, you know, it wouldn't have been possible. Now with language models, we can, and it's because language models encode our knowledge, our notions of what's interesting, even just basic priors that we couldn't even put into words, like the complexity of a program, and the intuition and creativity and all of this kind of stuff. So now we can generate these programs, and we can mix them and remix them and do all of this kind of stuff. Everyone at NeurIPS is doing the same thing. I guess the question to you is, this was a restricted compute benchmark, so, of course, you made certain trade-offs that you wouldn't make otherwise. Like now at Tufa Labs, you've got all of the compute in the world, all the time in the world. What would you do differently?

Mohamed Osman: [0:45:56] That's a great question. I think the most interesting thing to me right now is the thing I mentioned about prompting everything in the forward pass at once and measuring the transformer ability, innate ability, and then tuning the reasoning, these types of lines. So there are so many angles to tackle ARC. It's not about the compute to do many experiments and to tackle the different angles and to get people on that will help and that are really interested in tackling these things and interested in thinking along these same lines. Right? Like, transformers are really bad at ARC. That's the statement that language models are really bad at ARC, and they are. But let's measure it, and let's try crazy ideas to get them to improve. That's what I'm after.

Tim Scarfe: [0:46:46] Tell me about this paper that you're writing with Jack and Michael.

Mohamed Osman: [0:46:49] **Me and Jack are working on a paper to outline the test time tuning. There's been papers now that outline the technical content of what we have, but we wanted to put the paper out there for people to cite the original kind of source. And people right now, by the way, are citing our MLST podcast.**

Tim Scarfe: [0:47:11] **No way.**

Mohamed Osman: [0:47:11] Yeah, lots of papers.**The Test Time paper by MIT, by Ekin, cites the podcast. Like, lots of people cite that MLST video podcast, which is not great, right?**

Tim Scarfe: [0:47:23] Good for me. **Yeah, it's probably because you didn't have it in writing anywhere. Yeah. Folks, you can continue to cite the MLST podcast as much as you want.**

Mohamed Osman: [0:47:33] That's actually amazing. **You should have like a counter. I hope people do that more, but I also want to put a paper out there so people can cite the written work, and we can track it better. I mean, this is a great format. We shared a lot about our method, and people got inspired by it and implemented it and put a lot of papers out.** So we want to put our paper there. Technically, it's very similar. With the stuff we're doing now, it's not far away from the stuff everyone else is doing. Some people are complaining about us not open sourcing. But again, the stuff is already out there. A professor told me this - I'm more worried about OpenAI not open sourcing rather than MindsAI. So I think that's the key thing. And another key thing is we're working on more papers, not just that one, that explore different angles, and we're going to be putting lots of stuff out there in the very near future. I'm super excited about that.

Tim Scarfe: [0:48:39] Quick point on that. Why did you not open source the paper, you know, the approach and already put more out there?

Mohamed Osman: [0:48:46] We were in a very tough position from many different angles. The requirement for open sourcing was a little too extensive. They required weights, but also training code, training examples and the entire test time code. There was a lot of things. We felt like with Michael's DSL that so many people are now using and have cited for everything ARC-related for test time tuning and voting that everyone is using to get the top scores, we've contributed a lot to the community. The requirements were a little too much, especially for us targeting that 85%. The incentive structure really didn't make sense if you look at the prize money and the potential gain. I think lots of people were making the comment - the incentive structure, if you don't get 85% from the first competition, just wasn't great. But we're super excited to do that moving forward. The competition team have told us that they're working on that absolutely, and they're taking that feedback.

Outline: MO and TS discuss test-time tuning for ARC, Kevin Ellis's shift from DSLs to LLMs, and MO's upcoming paper with Jack documenting their method, noting MLST podcast citations in academic papers - clear audio quality with sustained engagement and minimal background noise. Points:

- Kevin Ellis has shifted from DSLs to embracing language models due to their ability to encode implicit knowledge and handle complex program generation
- Discussion of transformer capabilities in ARC and the need to measure and improve their performance through experimentation
- Academic papers are citing MLST podcast for test-time tuning methodology
- Explanation for not open-sourcing their approach due to extensive requirements and incentive structure issues

References Mentioned:

- Kevin Ellis's recent work on LLM-Guided Compositional Program Synthesis, marking shift from DSL-based approaches to language models by **Wen-Ding Li and Kevin Ellis** (Link)

5.3.2 MindsAI Team Progress and Acquisition by Tufa Labs

Tim Scarfe: [0:50:07] Just to be clear to the audience, what was the incentive given that you got around fifty-six, fifty-seven percent? What was the incentive? If you did open it up, what would you have got?

Mohamed Osman: [0:50:18] That's a very good question. Just for reference, we started out at thirty-three percent at the beginning of the competition. We did all the work to get it to fifty-five point five. We even got fifty-eight score on the hidden set, but it wasn't on the leaderboard because the time had passed. We moved up a lot, and what we would have gotten was twenty-five thousand US dollars. After taxes, not a huge incentive. But I want to say it was a great competition, and they did a lot to put the word of ARC out there into the world. They've taken that feedback, and we have also up-leveled the community and shared a lot. So I feel like we're both happy, and we don't need to dwell on it too much - we can just look forward to next year.

Tim Scarfe: [0:51:17] You guys are now working with Tufa AI Labs in Zurich. As I understand, Tufa have acquired MindsAI. Tell me the story.

Mohamed Osman: [0:51:27] The whole team is now working out of Tufa Labs. We have a lot of funding, we have a lot of compute coming in. We're going to be purely focused on ARC for around a year and different angles to ARC. We're working on not just this paper, but another one and a couple of other angles that we hope turn into papers. We are going to be putting stuff out there, and we're hiring, and we have compute coming. So apply if any of that sounds interesting to you. We have also plans for things to do after ARC. We spoke a little bit about this with the compositionality of these large language models - it's not there. How can we get them there in a more general format? Lots of system two goals, and test time obviously is a very ripe area more generally for research, so we're going to be exploring all of that beyond ARC.

Tim Scarfe: [0:52:35] All of that is very exciting. The compositionality problem - Clem was saying the same thing. That is the golden ticket. If we can solve the compositionality problem, I think reasoning opens up. But presumably, the approaches that you're going to be working on now are going to be slightly broader than what was appropriate for this particular benchmark. So you're going to be doing a few things differently. What particular kind of strategies and approaches are you going to be looking at now?

Mohamed Osman: [0:53:03] Different creative ways of doing test time is really interesting. **First, I'll say you always have to target the hundred percent with ARC. It's very hard to optimize for a certain competition for something like ARC - it's a really bad idea. It's not going to give you lots of score because of how private the dataset is, and you can't make guesses about it. It's just formulated in a really nice way where you have to go for broke.** Whether you are working in a small team with very little funding or with a lot more funding, I think it kind of is the same, but what we get now is different angles. Research angle and the competition angle, and these are super well aligned, by the way - they mesh together really well. Just more angles of that. I'm also thinking about different benchmarks around ARC. I have some really good ideas there. ARC is a special format that allows you to benchmark certain things. Maybe we can go into that at another time.

\textbf{Outline: MO discusses MindsAI team's progress in ARC competition (33\% to 55.5\%), subsequent acquisition by Tufa AI Labs, and future research directions focusing on compositionality and test-time learning, with TS exploring implications; audio quality is good with neutral, consistent prosody and clear articulation.} **Points:**

- MindsAI team improved ARC performance from 33\% to 55.5\%, with 58\% on hint set post-deadline
- Tufa Labs acquired MindsAI team, providing substantial funding and compute resources
- Research focus on improving compositionality in large language models and exploring test-time optimization

5.3.3 ARC v2 Development and Performance Scaling

Tim Scarfe: [0:54:15] What do you think will happen when Chollet releases the new version of ARC? And by the way, one thing that Kevin and Zenna said, which I didn't predict, was that has ARC become a lame duck benchmark now that Chollet has said he's going to invalidate it next year?

Mohamed Osman: [0:54:30] Invalidated meaning?

Tim Scarfe: [0:54:31] He's making a new version of it. So it's almost like, are you wasting your time working on ARC one when everything changes with ARC two?

Mohamed Osman: [0:54:38] So I think ARC two is going to be the same format as ARC one. Right? So I think what what's going to happen is it's just going to get harder.

Tim Scarfe: [0:54:47] So what he said to me was that he's employing loads of humans to design and select tasks that are at the appropriate level of difficulty. Some tasks are insanely difficult and most humans don't get them right, and some are too easy. And there's an interesting overlap between

what's easy for a human and what's easy for a computer. Chollet, of course, is a big believer that they are a proxy for intelligence, that there should be some kind of g factor or generalization between them. The other thing I wasn't entirely clear on is, is he protecting ARC two, you know, the second version of ARC through diversity? So just reducing information leakage through sheer diversity, or is he talking about creating a dynamic benchmark which is almost entirely impervious to overfitting?

Mohamed Osman: [0:55:37] I will say this. I was talking to Chollet, and I asked him, are you going to have a dynamic benchmark where you iteratively have some kind of framework where you can ask for more? And he did say no. It's going to be the same format, just different data and very likely more difficult in the right way they are calibrating for. So I think one thing that Chollet speaks about related to your first question is getting the right level and this idea of idiosyncratic results. Like one-off riddles that are so creative. I think it requires real creativity to come up with those riddles. They're so creative, they're just kind of very novel. Like, you can't label them. If you were working on a labeling approach for ARC and trying to find the right label, it would be hard to categorize. They're one-offs. And I think what he's going to do is have a lot more of these idiosyncratic riddles. And I think that's interesting. I think that's a good thing. But I also think that the original ARC formulation was already good enough, and it's already enough for us to come up with new methods. And even if it remained the same, there's still lots of angles where you can measure generalization, improve on it, even with this fixed thing. **And I do think it gets solved with scale. Like, I think ARC v1, give us four 3090s or two 3090s and more time on the problem next year, and absolutely, we get to 85 percent. For ARC v1. I do think it scales, and I do have some data on that with the hidden test set that it does scale, actually, in a very interesting way that you wouldn't expect.** But I think it does scale, and I'm okay with more idiosyncratic results. I think maybe that would tune our signal for what is generalization a little bit better. So they're putting a lot of hard work into that, and I'm excited for it. Just maybe if they do listen to this, one request or hope that they also open up the v1 again so that we can have a benchmark for progress on the methods. Like, we know the scores really well on v1. If we can still submit to v1, even though now there's a v2 and it's very different, if we can still submit to v1, that would be really good to have maybe as a validation set or another benchmark. You know what I mean?

\textbf{Outline: TS and MO discuss upcoming ARC v2 benchmark by Chollet, with MO revealing v2 maintains format but adds more idiosyncratic riddles, while expressing confidence that v1 will scale to 85\% with sufficient compute; audio maintains neutral tone with high clarity (7/10) and consistent engagement (8/10).} **Points:**

- Questions if ARC v1 has become obsolete given Chollet's announcement of v2
- Confirms ARC v2 will maintain same format but increase difficulty
- ARC v2 will focus on idiosyncratic riddles that require genuine creativity
- Claims ARC v1 can reach 85\% performance with sufficient compute (4x 3090s)

References Mentioned:

- François Chollet's Abstraction and Reasoning Corpus (ARC) benchmark, a test designed to evaluate general intelligence through abstract reasoning tasks. Currently hosted on GitHub with ongoing development of version 2. by **François Chollet** (Link)
- December 2024 O3 breakthrough on ARC-AGI showing significant performance improvements with increased compute, suggesting scaling properties align with Osman's predictions by **OpenAI** (Link)

5.3.4 Intelligence Benchmarks and Transformer Limitations

Tim Scarfe: [0:58:22] Yeah, I mean, Chollet's measure of intelligence was so fascinating because he was talking about this adaptation and being able to create a new skill program in response to novelty. And if you think about it, it's not a foregone conclusion. This is actually a very difficult thing to do from a psychological point of view, because you're trying to create a general benchmark that given the base knowledge of the average person, they would be able to do this generalization. You gave the example of the one-offs. That would be why are manhole covers round? Like these ridiculous riddles that they used to use at Microsoft to hire people. Only one in a thousand people would say, the information gain on that is basically zero. So you need to create a set of programs that the average people with their base priors would be able to do. And ARC solved that really well, and I think Chollet doesn't get enough credit that he did so cleverly select challenges that would work well for. And of course now he needs to diversify across that. One other quick question is, in your experimentation, did you notice patterns? Like which types of tasks are you failing on and which ones are you doing well on?

Mohamed Osman: [0:59:33] That's a good question. Professor Melanie Mitchell has a great benchmark concept, Concept ARC. She's been thinking about this for a long time. She has lots of great ideas there. And one thing I'm noting is, I've spoken to lots of people that are in the benchmark and are using neural methods. And they all say concept ARC counting is the lowest, right? Like, it's just a thing. And if you try to add more counting-based trials, or I don't think you should add priors. We haven't tried that - priors in terms of feature engineering and that kind of thing is not good. But just counting for some reason is really abysmal for neural networks.

Tim Scarfe: [1:00:24] Yeah, and I can tell you why.**I interviewed two guys at DeepMind on Monday, and they've studied this in self-attention transformers. They actually say it's because of repre-**

sentational squashing. I won't spoil the surprise - you can watch the interview. But there's a couple of problems with transformers that because of the way they're set up, they almost all of the attention gets kind of focused on in the limit. Actually, if you scale transformers all the way up, all of the attention gets focused on the first token. And there's also another problem with the softmax function that creates this kind of directedness. And for certain types of reasoning tasks, you want it to be directed, and for certain types of creative reasoning, you want it to be diffused. They were basically saying that transformers, even in a trivial sense, cannot do counting or copying. They just can't do it. Like you can give it a trivial example where you say, you know, count up to a hundred, and you just say, can you count these numbers up? And it just fails abysmally straight away. Copying's an interesting example, because even if you use tools - people say, well, it's okay, you can just use tools, you can just stick it into a Python tool. If you can't even copy the tokens into the tool, then you can't do that. But the upside of this is all we need to do is fix this problem. All we need to do is whatever the problem is in these architectures, if we can make them copy and count, then maybe all of these problems will just disappear.

Outline: TS and MO discuss intelligence benchmarks, focusing on ARC and Prof. Mitchell's ConceptARC, highlighting transformer limitations in counting tasks and MO's observation that neural networks consistently struggle with counting problems - audio quality good with minimal background noise. Points:

- Discussion of Chollet's intelligence measure focusing on adaptation and skill program creation in response to novelty
- Reference to Melanie Mitchell's ConceptARC benchmark and observation about universal poor performance on counting tasks
- DeepMind research reveals transformer limitations in counting due to representational squashing and attention focus issues

References Mentioned:

- ConceptARC Benchmark paper by Melanie Mitchell et al., published on arXiv in May 2023. The benchmark systematically assesses abstraction and generalization abilities on spatial and semantic concepts within the ARC domain. by **Arseny Moskvichev, Melanie Mitchell** (Link)

5.3.5 Neural Architecture Optimization and Processing Distribution

Mohamed Osman: [1:01:50] Yeah, it's exactly that. You need to dive really deep into the architecture and see where the problem is. And in this case, it can be very clear what the problem is, right? The softmax is a max - kind of a cheat code way to achieve something similar to a max. And you can see how that would bottleneck things. You can also see how more layers matter. There is something a really brilliant postdoc told me this. He told me, you want to do the processing in a single layer.**You don't want to do the processing of these things in one layer. You want to do them as you go up the layers. And what he means is if you're counting all in one layer, that means you're overfitting,**

right? Why? Because you're not really running an algorithm. You're just taking this feature, this feature, and you're doing it in this one layer. But if you just have one layer that goes and attends to this thing and updates itself and then goes and updates this thing - that's the general algorithm that you want. It's not this heuristic kind of MLP at one single layer that will just get this feature and this feature and this feature and do everything in one layer. That's not what you want. And so there are lots of ways of doing that that I'm super excited to explore. And I think you hit the nail on the head there with what you just described.

Tim Scarfe: [1:03:22] Mo, it's been amazing to have you here. Thanks so much for coming on.

Mohamed Osman: [1:03:25] Thank you so much.

Outline: MO discusses neural network architecture optimization, emphasizing distributing processing across layers rather than single-layer processing to avoid overfitting, with TS concluding the interview - audio shows high clarity (8/10) and engaged discussion with minimal background noise. Points:

- Identifies softmax as a bottleneck in neural architectures, acting as an approximation of max function
- Emphasizes importance of distributing processing across multiple layers rather than concentrating in single layer
- Single-layer processing indicates overfitting as it's not implementing true algorithmic learning

References Mentioned:

- Discussion of softmax bottleneck in neural networks and its limitations as approximation of max function. See: 'Mixtape: Breaking the Softmax Bottleneck Efficiently' (NeurIPS) by **Yang, Zhilin and Dai, Zihang and Salakhutdinov, Ruslan and Cohen, William W.** (Link)

6 References

Bonnet, C. (2024). Searching Latent Program Spaces. arXiv:2411.08706.

Chen, X., et al. (2024). SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities. CVPR 2024.

Chollet, F. (2019). The Measure of Intelligence. arXiv:1911.01547.

Chollet, F. (2024). ARC Prize 2024 Technical Report. arXiv:2412.04604.

Chollet, F. (2019-2024). Abstraction and Reasoning Corpus (ARC). GitHub repository.

DeepSeek AI (2024). DeepSeek-R1-Distill-Llama Performance Report. Hugging Face.

Ellis, K., et al. (2020). DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. arXiv:2006.08381.

Li, W.D., Ellis, K., et al. (2024). Combining Induction and Transduction for Abstract Reasoning. arXiv:2411.02272.

Li, W.D., Ellis, K. (2025). LLM-Guided Compositional Program Synthesis. arXiv:2503.15540.

Moskvichev, A., Mitchell, M. (2023). ConceptARC: A Benchmark for Compositional Reasoning and Generalization. arXiv:2305.07141.

OpenAI (2024). O3 Performance Breakthrough on ARC-AGI. arcprize.org.

Raffel, C., et al. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683.

Ruis, L., et al. (2024). How Language Models Use Knowledge from Pretraining Data for Reasoning. arXiv:2411.12580.

Yang, Z., Dai, Z., Salakhutdinov, R., Cohen, W.W. (2019). Mixtape: Breaking the Softmax Bottleneck Efficiently. NeurIPS.