**Proposal 1**

The big problem we’re trying to solve here is to verify the output of a Neural Network when we don’t know the right answer. In an ideal world, there would be some way to look at all of the computations going on in a Neural Network and prove that if those computations show up “in the wild”, the Neural Network will get the answer right

* To use an NN in an artificial pancreas system, we’d really like to have a way to fully verify it (ie. for any decision it makes, we need to be positive that it’s within the range of reasonable decisions)
  + Your conformance verification paper discussed verifying just one property of an NN’s output, but I want to check that the output as a whole is correct (ie. I want to ensure that the output is within the range of reasonable decisions according to *all* properties)
* This is difficult with current approaches because of the problem of specification. How do you specify what the right answer looks like when you don’t know all of the properties you’re looking for?
  + In the past researchers have:
    - Looked at a whole range of points to make sure that the output of the network would fall within a certain range on all points
    - Looked at a whole range of points to make sure that the output of the network had some property on all points (this is what your research did)
    - Taken the output as a first step and run through a bunch more timesteps to see if the neural network
    - Looked at specific points to try to make sure that small changes to them won’t change the overall model behavior (this is a way of verifying a system when you don’t know the right answer, definitely on the right track)
* The paper gets around this by verifying that “All inputs with a specific Neural Activation Pattern correspond to the same class”. Then we can verify that “Given an input and a NAP which it falls into, the set of points within a certain radius of that input also satisfy the NAP specification.” The magic of this is that now we can use a training data point that we got right to carve out a region of space where the network will make the exact same decision
* The assumption we have to make is that the model’s “thought process” was correct in order to generate the right example in training. So all the other inputs where the model uses the same “thought process” will also be correct (and we can mess with a specificity constraint to dial this in)
  + This approach fits so well with the artificial pancreas because it’s so hard to specify what exactly we’re looking for in a certain output (if we could specify every property that an output must have, we wouldn’t need to neural net)
  + For each output we also have no way of telling if it’s wrong or right because we’re just getting intervals so we can’t project outwards into the future
  + You mentioned in one of your lectures that the future was in combining a neural net with a traditional system. This seems like one of the best ways to do it, we use the neural net until it gets out of bounds and then we switch to a traditional system
* I wanted to see what you thought of all of this. The dream would be to get to work on paper that incorporated these ideas

**Proposal 2**

We can use the NAP approach to verify NNs of much larger sizes than the current SOTA. If we (1) do the verification at run-time and (2) figure out a way to mine NAPS without looking at the whole input space

**Proposal 3**

We can adapt the NAP approach to work with regression models and to work with time-series models so that it can actually be used in practice for the artificial pancreas

**Proposal 4**

We can combine Heuristic Estimation with the goal of expanding the space of verified inputs. Instead of going through millions of data points to mine NAPS we start from the ground-up: proposing NAPS heuristically.

**Proposal 5**

KANs are made to be much more interpretable so maybe Christiano's approach will work better on them (ie. the issue of having to assume worst-case noise may be solved by KANs)

SELECT t1.state\_name FROM highlow AS t3 JOIN border\_info AS t1 ON t3.state\_name = t1.border JOIN state AS t2 ON t2.state\_name = t1.border WHERE t3.lowest\_elevation = ( SELECT MIN ( lowest\_elevation ) FROM highlow ) ORDER BY t2.area DESC LIMIT 1

LIMIT 1 FROM WHERE t3.lowest\_elevation = PROJECT MIN(lowest\_elevation) FROM SELECT highlow ORDER BY t2.area DESC FROM PROJECT t1.state\_name FROM SELECT highlow AS t3 FROM JOIN highlow AS t3 AND border\_info AS t1 ON t3.state\_name = t1.border FROM JOIN state AS t2 AND border\_info AS t1 ON t2.state\_name = t1.border

LIMIT 1 FROM ORDER BY DESC t2.area FROM t3 FROM ORDER BY DESC t2.area FROM PROJECT t1.state\_name FROM AGGREGATE PROJECT t1.state\_name AS t3 FROM SELECT WHERE t3.lowest\_elevation = PROJECT MIN(lowest\_elevation) FROM AGGREGATE MIN(lowest\_elevation) FROM highlow FROM JOIN AS t3 FROM highlow WITH JOIN AS t1 FROM border\_info WITH AS t2 FROM state ON t2.state\_name = t1.border ON t3.state\_name = t1.border

an extra ORDER BY clause in the TSQL and a PROJECT within an AGGREGATE

**Open Questions**

**CENTRAL QUESTION: Can you verify the statement: all inputs with this property have this output**

It would be amazing to have it so I only have to verify a very small part of input space to make sure the NN is making its current decision well. But how do I verify the prtoperty that no input falls into 2 different NAPS?

Did recent ARC research solve Paul’s issue? Is there now a formalized heuristic estimator that works?

NN’s are so hard to verify because you can’t account for all the connections but in conformance verification is sriram accounting for all the connections?

Could grad descent be used to find explanation (aka minimal neural specifications)?

Heuristic estimators are sort of like an abstraction based verification where you only do the interval arithmetic on the most important parts and then you replace the rest with worst case scenario which is why it has the potential to scale better?

My idea will focus on one of the most overlooked problems which is specification of the problem to be solved. If I can actually formalize the notion that the NN is within distribution, then we could apply some techniques and stuff

**Insert Into Proposal**

1. **Core points I need to make: need to justify why it’s so important to verify the net for examples we haven’t seen**
2. **Core problems I need to solve: Make NAPS work on regression models, Make NAPS work on massive models**
3. **Connect this stuff back to what I think Sriram’s research agenda is**
4. This idea seems to have a blind-spot for regression models but a way to fix that is to
   1. Time horizons, rachability analysis…
5. Traditionally work has been done to only verify the robustness of a neural network by looking at very small regions and making sure they were all the same
6. The state of the verification field right now is verifying when we know what properties the right answer should have; there are entire competitions dedicated to this but I feel like this is not useful in many real-world scenarios (including the artificial pancreas)
7. “However, in many cases, deriving such laws from “first principles” may be quite cumbersome, if not outright impossible. Imagine a system running by a patient’s bedside in the intensive care unit of a hospital with a continuous stream of data that includes the patient’s blood pressure BP (kg m−1 s−2), lung volume V (m3 ), pulse P (s−1), and body weight W (kg). In this situation, it is unclear whether there are “precise” equations derivable from first principles, or even “approximate” empirical equations that may hold under some situations. Be they exact or approximate, these relationships are useful in numerous applications such as the run-time monitoring of safety critical systems.” -Sriram
8. “This poses a tricky chicken-egg problem: machine learning (ML) is necessary because it’s challenging to formally write down a precise definition (aka specification); but to be able to verify machine learning models, a formal specification would be needed.”