OMNeT++ Community Summit 2020, Virtual Summit, October 5-6.

# Proposed Research Topic: A Neural-Network-based WiFi Error Model in INET

Attila Török

# “Proposed Research Topic”

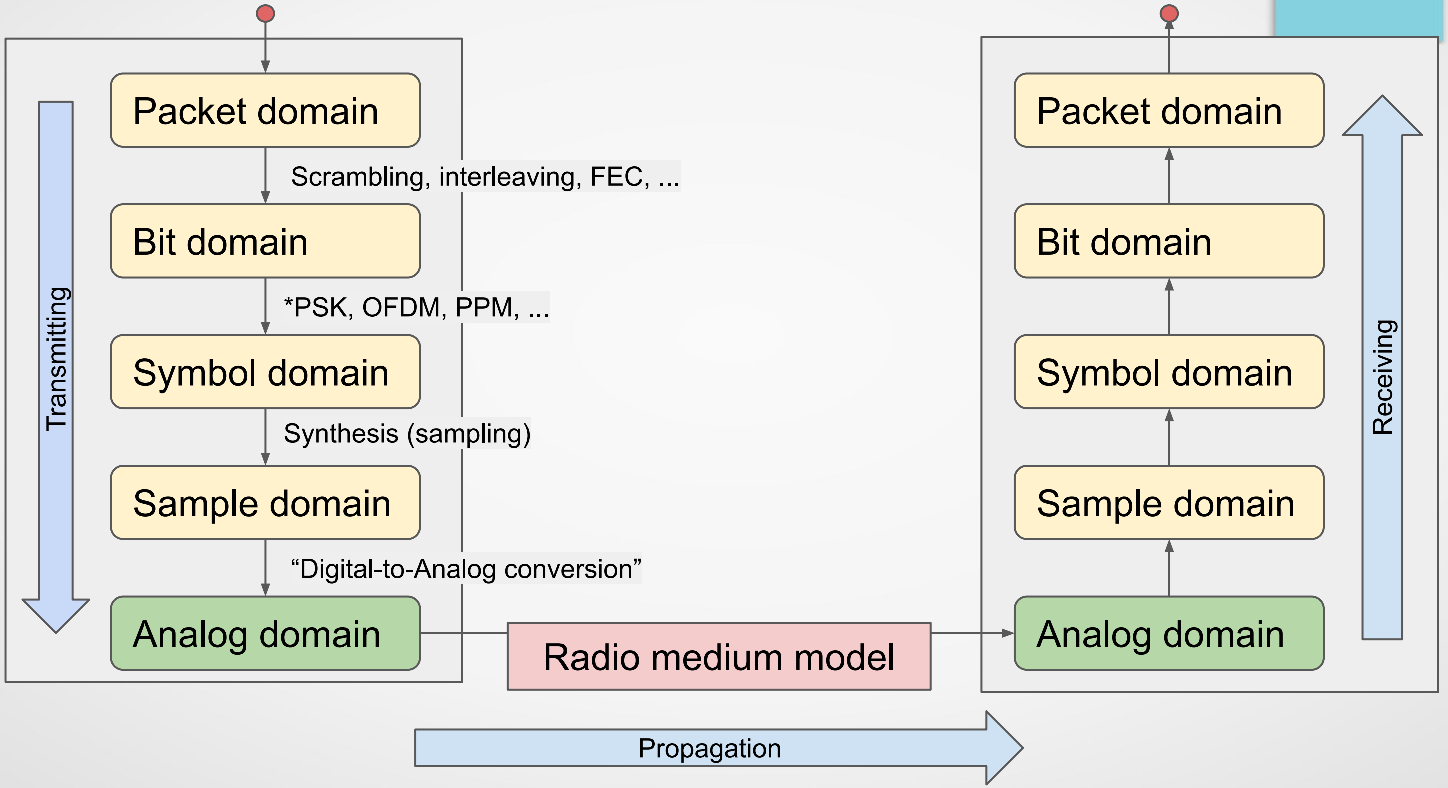
- NOT finished research
- NOT even research underway
- A promising research topic for those looking for one
  - (We see the potential in the idea, and find it exciting, but we don’t have the resources [mostly time] to elaborate it in-house)
- Why?
  - Practically very useful
  - Doable
  - Novel
  - Plenty of questions and degrees of freedom

# What's an Error Model?

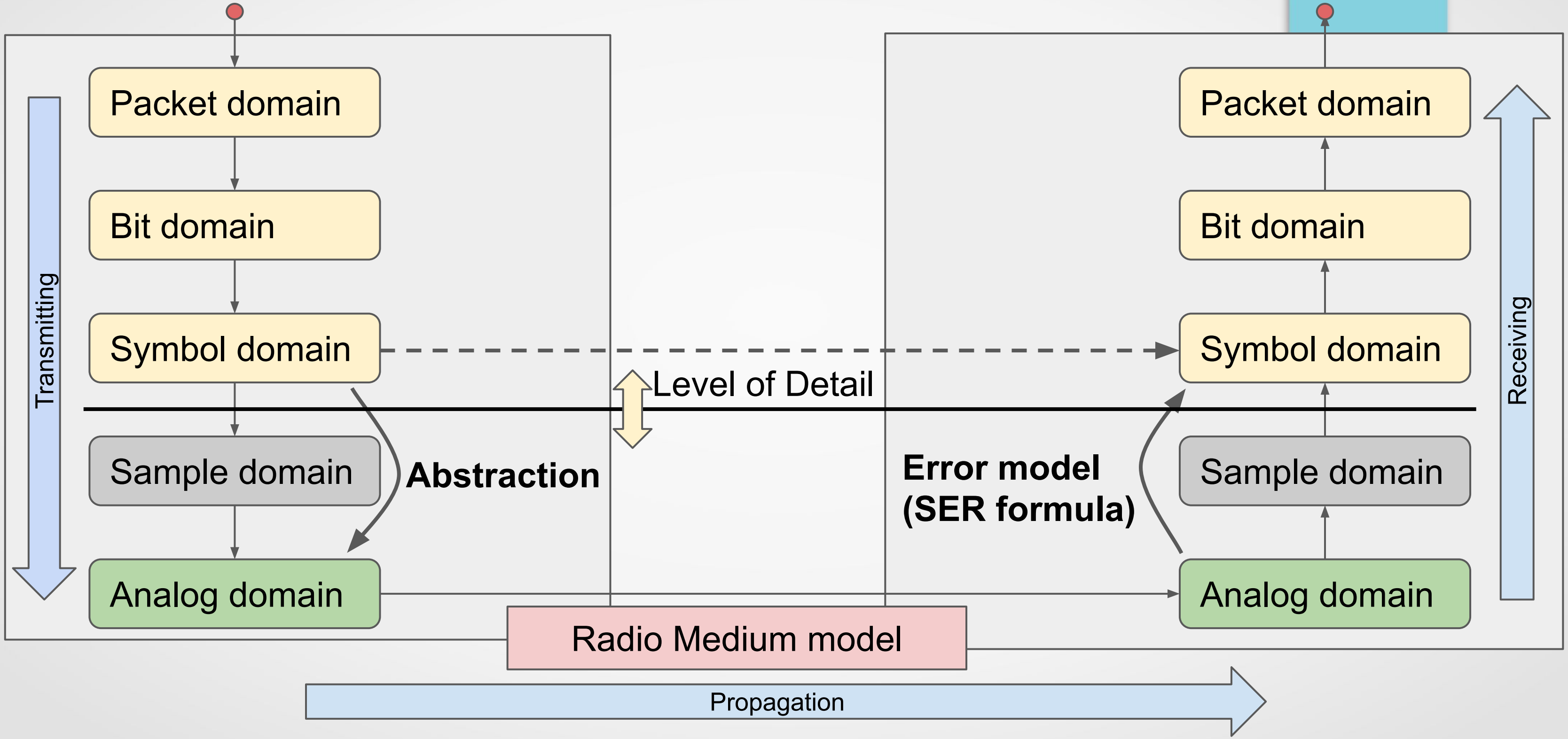
- Describes how the signal-to-noise (and interference) ratio (SNIR) affects the amount of errors at the receiver

- Input:
  - SNIR, computed from:
    - The attenuated transmission at the receiver
    - Interfering signals
- Output:
  - Error rates for units of given domain (bit, symbol, packet)

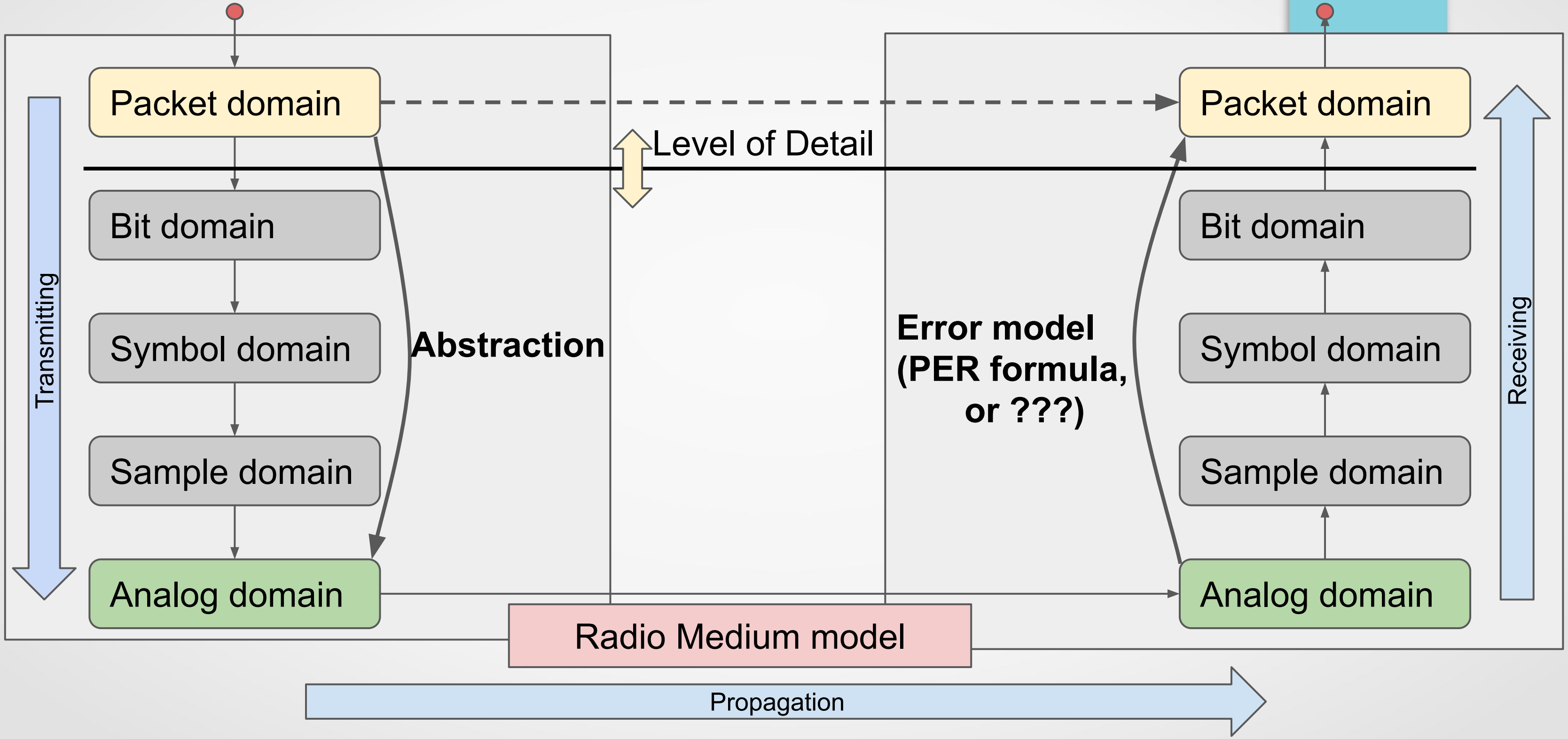
# Wireless Frame Transmission Model



# Symbol-level, Analytical Error Model



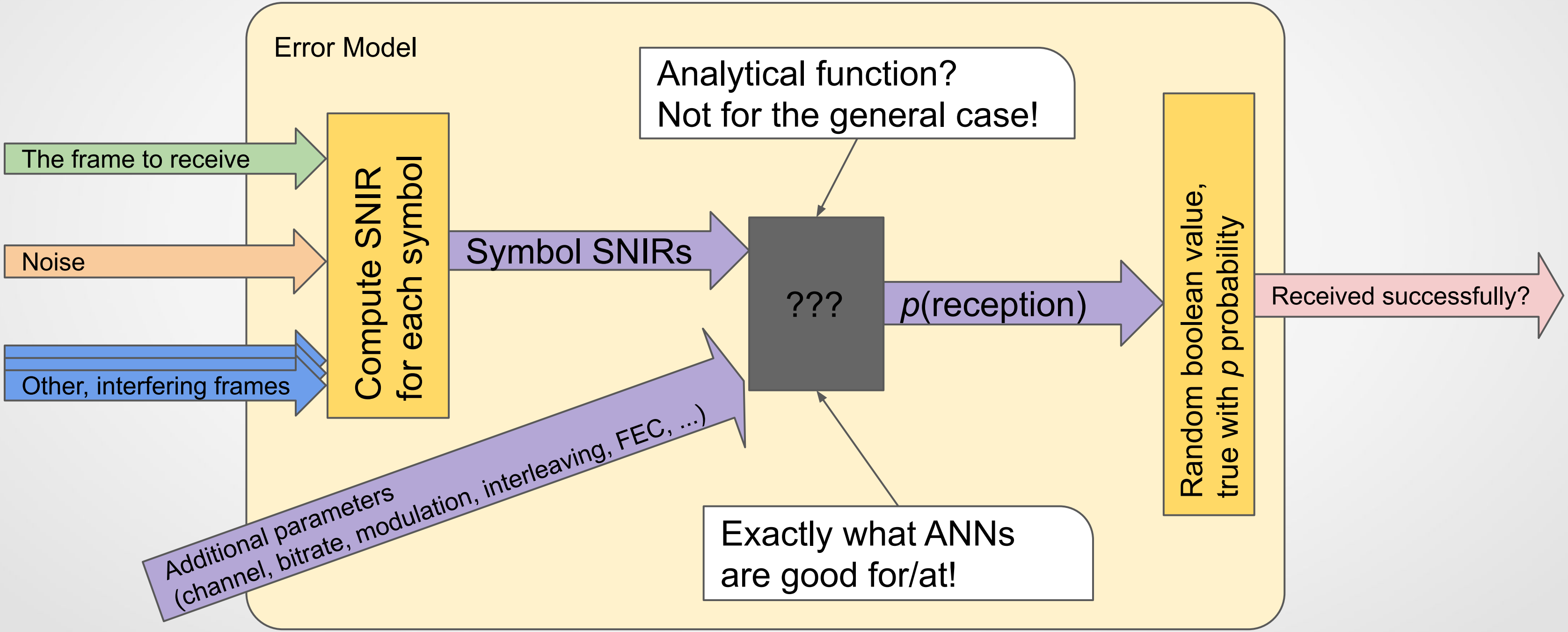
# Packet-level Error Model



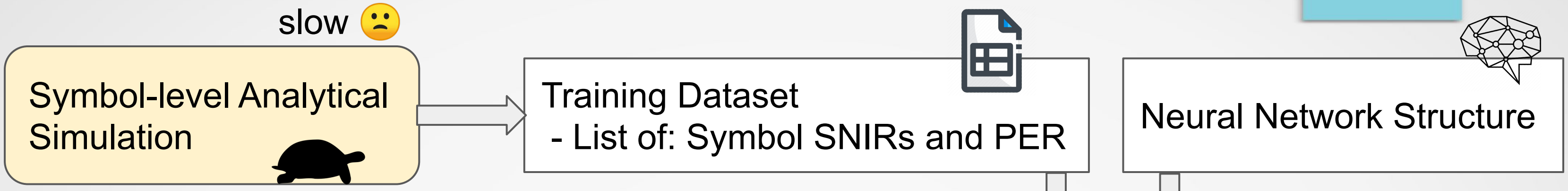
# Motivation

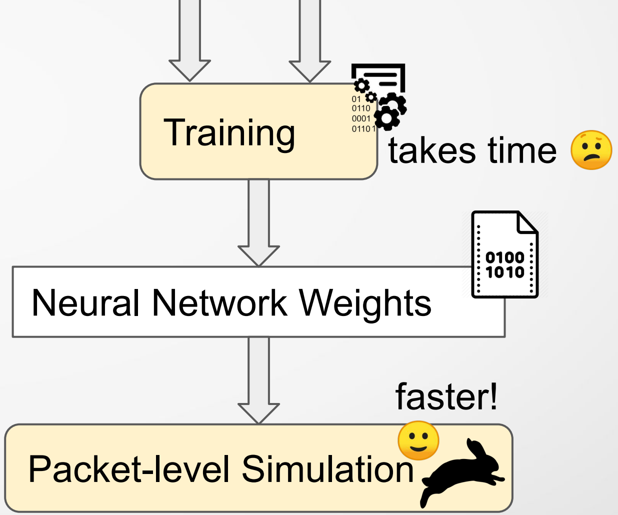
- The existing error model only takes into account **a single SNIR value** for a **whole frame**
  - Minimum or mean
  - Both choices are bad for different cases
- Generally a hard problem to tackle analytically
- Really is just a many-dimensional curve-fitting problem…
  - Exactly what neural networks do!

# An Error Model in INET



# Workflow



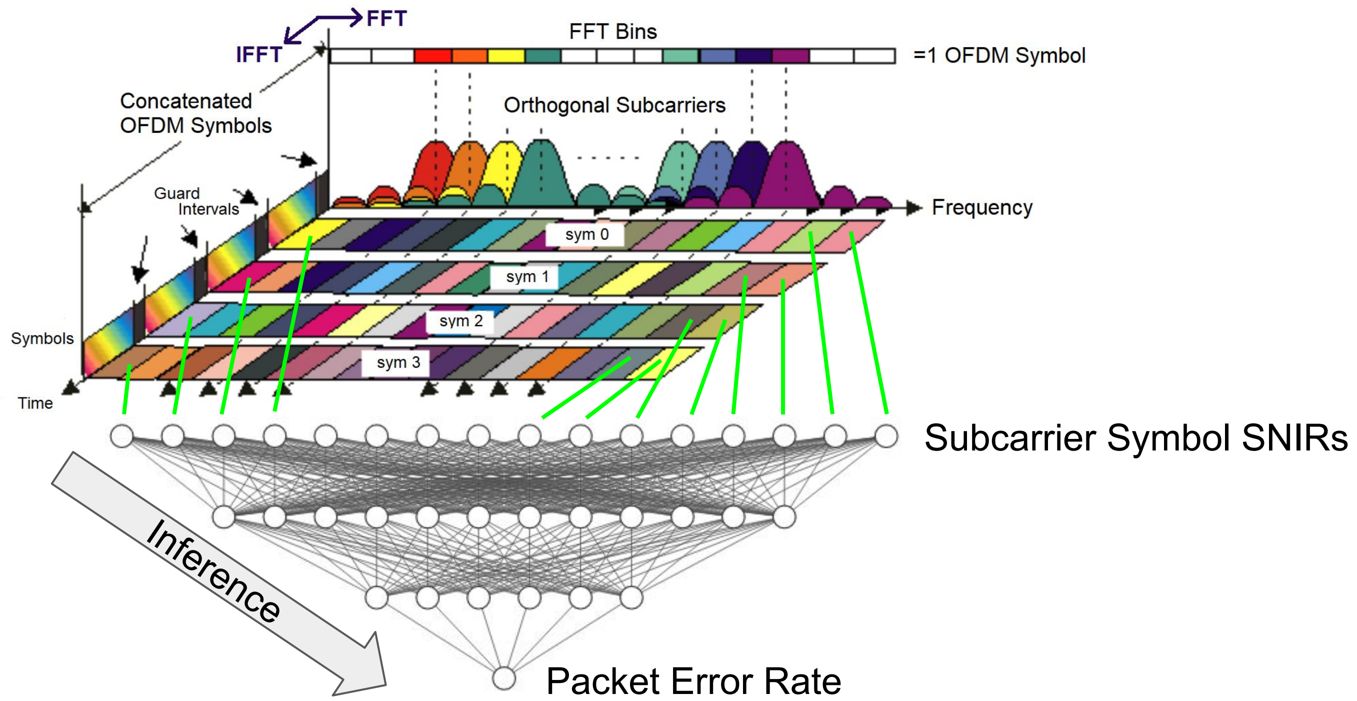




- Tradeoff to make:
  - Many specific models
  - Fewer generic models
- Shifts computational need to the training phase

# Generating Training Data

- Generate random WiFi frames (N~100k+)
  - Variable frame length and modulation
- Add noise and interference
  - Many parameters
- Put them through the symbol-level analytical model
  - Do a complete encoding/decoding of the frames, bits to symbols and back, compare results
  - Estimate probability of successful reception by taking many samples for each set of parameters

# Approach for WiFi

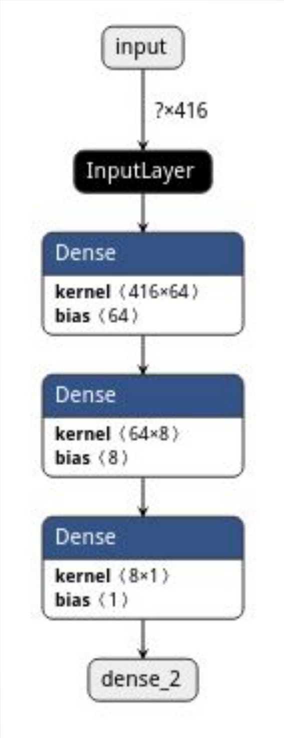


# Operation

- Compute SNIRs per symbol
- Run them through the trained network
- Randomly decide reception success based on PER
- The whole inferred function is in the ANN weights

# Neural Net Architecture

- Sequential
- One input neuron per grid-cell
  - Limits frame length
  - Smaller frames?
- Does not scale well
  - Performance issues
  - Prone to overfitting
- Inspiration from image processing



# Design and Training Tools

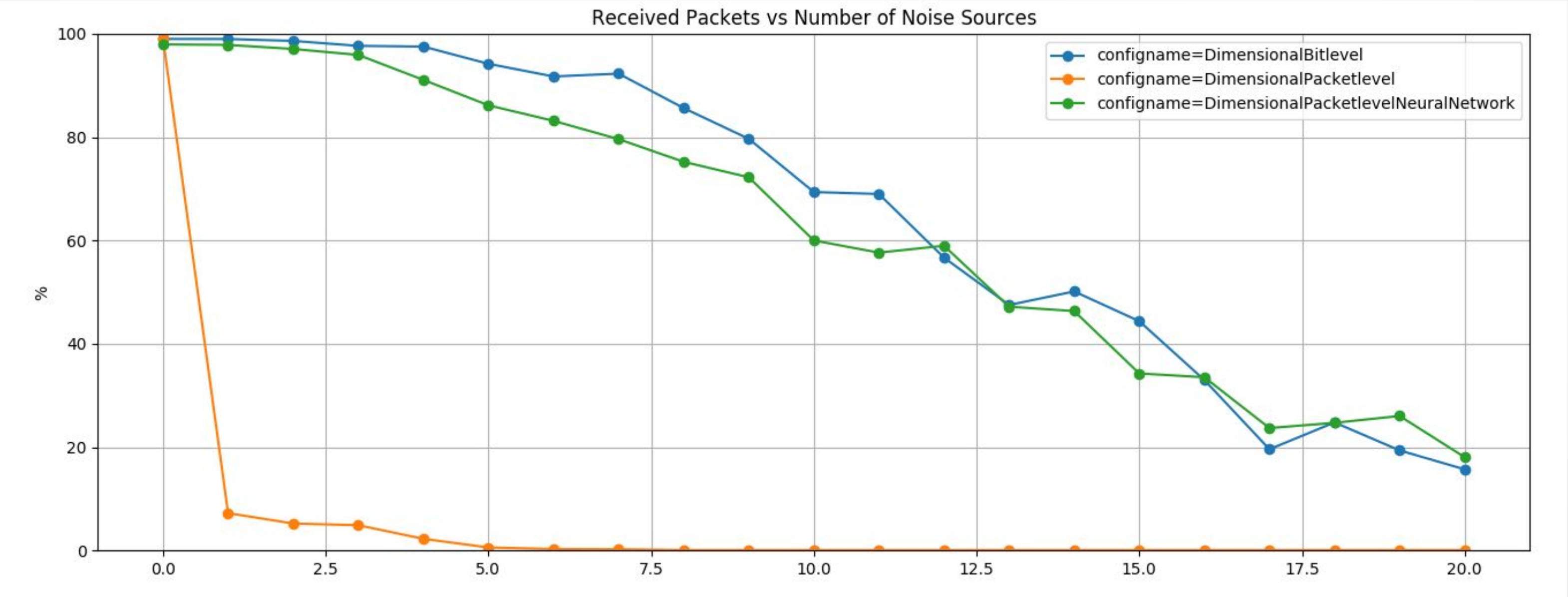
- Python 3
- Keras
- Netron

# Inference Tools

- Using a small, standalone C++ library:
  - https://github.com/gosha20777/keras2cpp
  - CPU only
- Alternatives:
  - https://github.com/moof2k/kerasify
  - https://github.com/GValiente/pocket-tensor
  - TensorFlow Lite
    - Geared more towards Android/iOS, not x86

# Preliminary Results

- 802.11g, 24Mbps, OFDM: 52x QAM-16,
  2:1 code rate, interleaving, scrambling, ...



# Idea: Recurrent Architecture

- LSTM or GRU cells
- More difficult to design and slower to train
- Can potentially work for all frame lengths
- Smaller, might be faster for inference
- Inspiration from text and language processing

# Experiments with Other Tools

- AutoKeras
  - Fixed basic structure, only tunes hyperparameters
  - Still in early phase, fairly incomplete
- Google Cloud AutoML Tables
  - Automatic structure generation
  - Results were not great

# Future...?

- Someone knowledgeable about neural networks could produce great results in a short time
- Proposal article: https://docs.omnetpp.org/articles/neuralnet-errormodels/
- Posted on Reddit: https://www.reddit.com/r/deeplearning/comments/fgb9yb/request_for_advice_on_neural_network_architecture/

# Thank You!

Obligatory Last Empty Slide