RadioML Redux: GTRI Efforts on the Army Signal Classification Challenge

Dr. Raj Bhattacharjea

Georgia Tech Research Institute

Electro-Optical Systems Laboratory

Outline

- Background
- GTRI Approach
- Complex-Valued Correlations
- Learned Linear Transforms
- Results and Final Thoughts

Background: Reconfigurability of SDR

- Protocol is reconfigurable
- Modulation is reconfigurable
- Frequency is reconfigurable
- Data rate is reconfigurable
- Encoding is reconfigurable
- …
- How can we be understand what we need to for decision making?

ML for Reconfigurable Signals

- If its all reconfigurable, how do we make algorithms to make sense of the spectrum?
- Learn the algorithms from data for your specific problem!
- Train in lab, perform in real world
  - Using accelerators available on the market thanks to deep learning for other applications
- Not hype; current “deep learning” is:
  - Linear algebra
  - Weak nonlinearities
  - Optimizers
- Data driven, requires good datasets

RadioML 2016.10A Dataset

- Open source dataset generation code
  - https://github.com/radioML/dataset
- Dataset license is Creative Commons Attribution - NonCommercial - ShareAlike 4.0
  - https://www.deepsig.io/datasets/
- Part of the GNURadio Extended Universe
- Labelled I/Q examples, synthetically created using GNURadio, pushed through channel models

ASCC Dataset

- Data usage under terms of an agreement
- Labelled I/Q examples
- Drawn from a larger repository
- Synthetically created, channel impairments

ASCC vs. RadioML dataset

ASCC

- 24 classes
  - **BPSK**, **QPSK**, **8PSK**, 16PSK, **QAM16**, **QAM64**, 2FSK-5KHz, 2FSK-75KHz, **GFSK**-75KHz, **GFSK**-5KHz, GMSK, MSK, **CPFSK**-75KHz, **CPFSK**-5KHz, APSK16-c34, APSK32-c34, QAM32, OQPSK, PI4QPSK, **FM**-NB, **FM**-WB, **AM-DSB**, **AM-SSB**, NOISE

RadioML 2016.10A

- 11 classes
  - **BPSK**, **QPSK**, **8PSK**, PAM4, **QAM16**, **QAM64**, **GFSK**, **CPFSK**, **FM**, **AM-DSB**, **AM-SSB**

ASCC vs. RadioML dataset

ASCC

- **Training examples are 1024x2 matrices**
- **Python pkl files**
- **Various SNRs**
- Various samples/symbol
- Channel…? Unknown
- ~30 GB of raw data

RadioML 2016.10A

- **Training examples are 1024x2 matrices**
- **Python pkl files**
- **Various SNRs**
- Passed through channel models
- ~600 MB of raw data

ASCC vs. RadioML Conclusions

GTRI ASCC Team Approach

- ML Stack: Keras/TensorFlow/CUDA/GPU
- Prototyped on RadioML, real runs on ASCC
- Generally 90%/10% train/test split
- Several parallel efforts
  - New ideas for RF signals ML
  - Hand-tuned network design
  - Evolutionary algorithms for architecture search

GTRI ASCC Team Approach

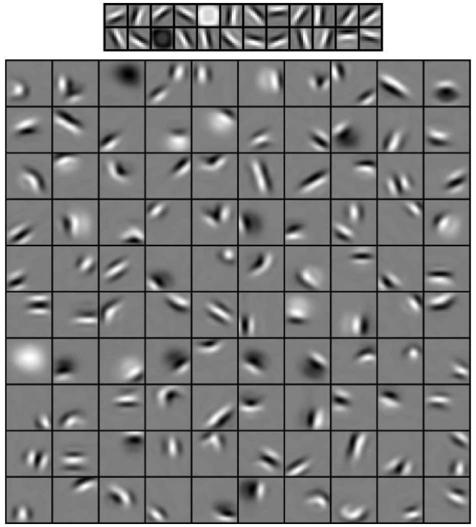
- ML Stack: Keras/TensorFlow/CUDA/GPU
- Prototyped on RadioML, real runs on ASCC
- Generally 90%/10% train/test split
- Several parallel efforts
  - New ideas for RF signals ML
  - Hand-tuned network design
  - Evolutionary algorithms for architecture search

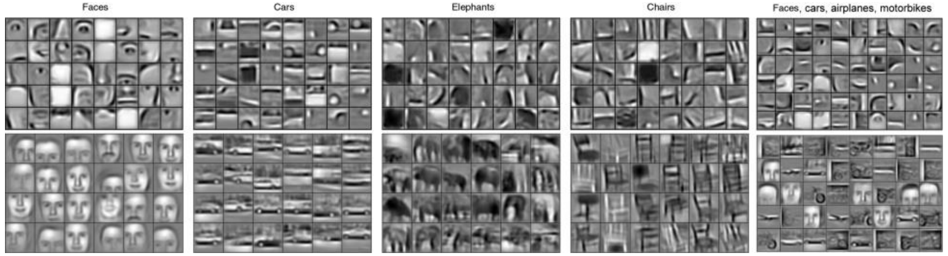
ML for Signals: New Ideas

- ~~Convoluted~~ Convolutional neural nets for complex-valued signals
  - Using real-valued packages (Tensorflow)
  - I will freely say convolution or correlation, they are equivalent when the weights are discovered through optimization
- New activation functions
  - CoReLU
- Complex max-pooling
- Learned linear transformations (LLT)
  - Update the weights in the linear part
  - Helps answer "which domain is best"

ML for Signals: New Ideas

- ~~Convoluted~~ Convolutional neural nets for complex-valued signals
    - Using real-valued packages (Tensorflow)
    - I will freely say convolution or correlation, they are equivalent when the weights are discovered through optimization
- New activation functions
    - CoReLU
- Complex max-pooling
- Learned linear transformations (LLT)
    - Update the weights in the linear part
    - Helps answer “which domain is best”





Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. *Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks*

CNN Performance Review

- Uses correlations and weak nonlinearity to find a snippet / feature
- Then looks for patterns of features
- Then patterns of those patterns
- And so on until you get a high-level list of features for an input
- Then map features to labels

$$f(Z_1) = \text{BPSK}$$

$$f(Z_2) = \text{QPSK}$$

$$f(Z_3) = \text{QAM-16}$$

$$f(Z_4) = \text{QPSK}$$

$$\vdots$$

$$f(Z_N) = \text{MOD}_N$$

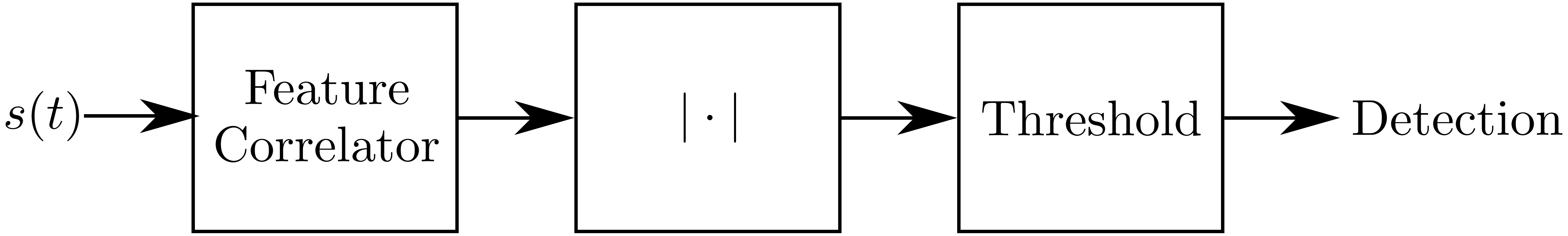
$f(Z)$ is differentiable

$f(Z)$ has free parameters (often millions)

Learning is updating the free parameters by optimizer

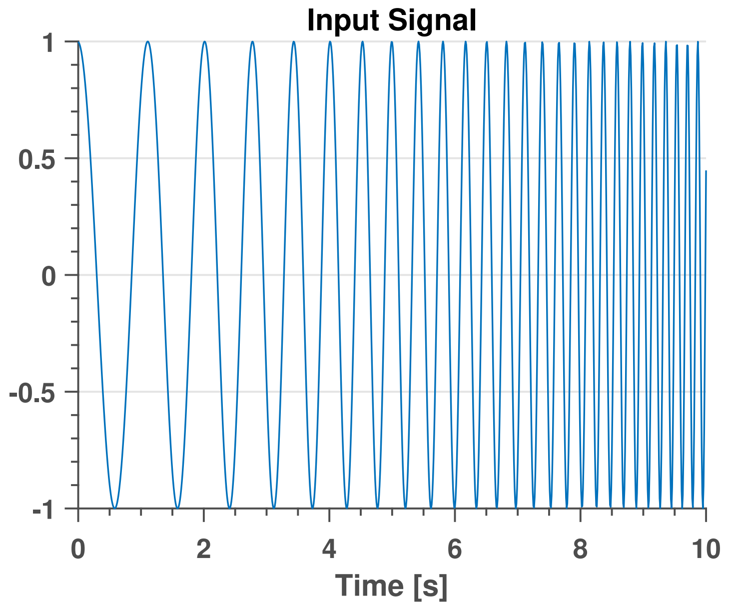
Objective is to minimize an error measure

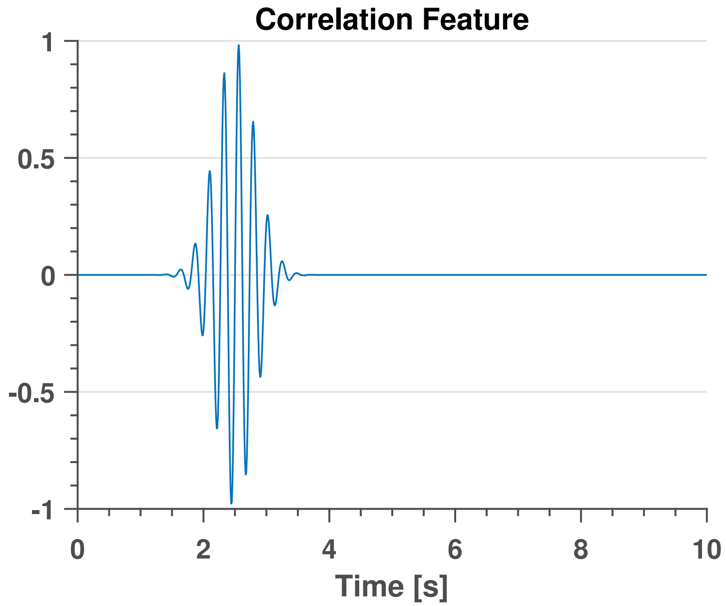
CNN Training Review

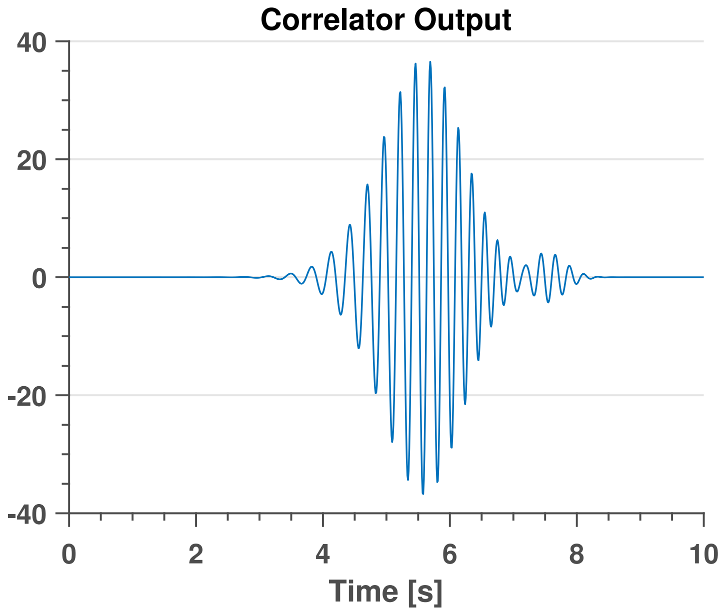


Correlation: Real Valued Example

- Correlator (matched filter)
- Magnitude
- If larger than a threshold, detected that feature



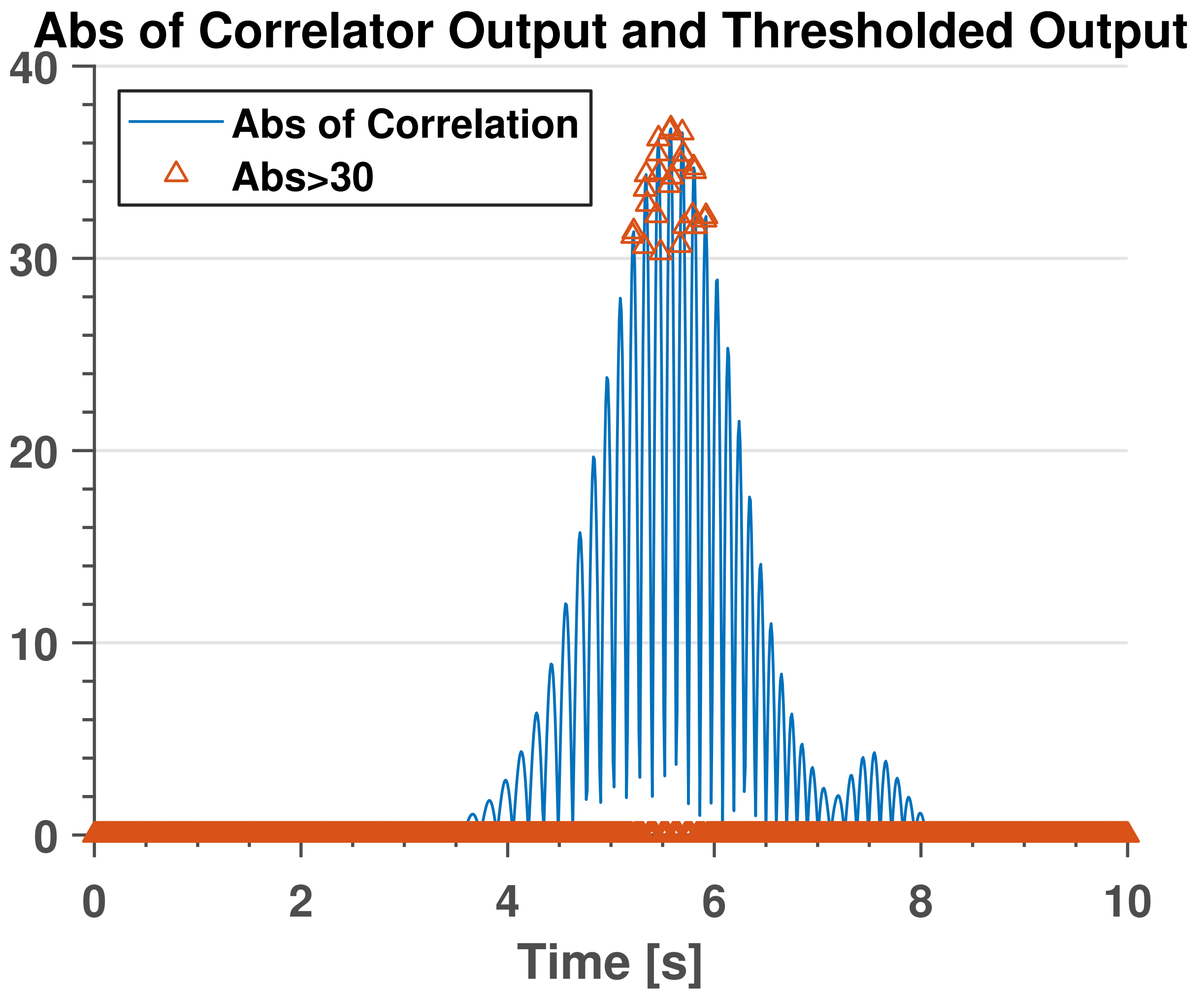




Correlation: Real Valued Example

Correlation: Real Valued Example

- Features are detected by the weak nonlinearity (absolute value and thresholding)
- **What is the natural extension to complex I/Q data?**



Complex Valued Correlation

- A sequence of complex numbers, Z_n
- A set of filter taps, h_m

$$Z_n = I_n + jQ_n \; = \; \begin{array}{|c|c|}\hline I_1 & Q_1 \\ \hline I_2 & Q_2 \\ \hline I_3 & Q_3 \\ \hline \vdots & \vdots \\ \hline I_N & Q_N \\ \hline \end{array} \quad , I_n, Q_n \in \mathbb{R}.$$

$$h_m = h'_m + jh''_m \; = \; \begin{array}{|c|c|}\hline h'_1 & h''_1 \\ \hline h'_2 & h''_2 \\ \hline h'_3 & h''_3 \\ \hline \vdots & \vdots \\ \hline h'_M & h''_M \\ \hline \end{array} \quad , h', h'' \in \mathbb{R}.$$

$$X_{\text{naïve}} = \begin{array}{|c|c|}\hline I_1 & Q_1 \\ \hline I_2 & Q_2 \\ \hline I_3 & Q_3 \\ \hline \vdots & \vdots \\ \hline I_N & Q_N \\ \hline \end{array} * \begin{array}{|c|c|}\hline h'_1 & h''_1 \\ \hline h'_2 & h''_2 \\ \hline h'_3 & h''_3 \\ \hline \vdots & \vdots \\ \hline h'_M & h''_M \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline \uparrow & \uparrow & \uparrow \\ I * h' & I * h'' + Q * h' & Q * h'' \\ \downarrow & \downarrow & \downarrow \\ \hline \end{array}$$

Complex-Valued Correlation

- Naïve real-valued correlation gives a three column result
- The parts look familiar though!

Complex-Valued Correlation in Real Math

$$X_{\text{naïve}} = \begin{array}{|c|c|} \hline I_1 & Q_1 \\ \hline I_2 & Q_2 \\ \hline I_3 & Q_3 \\ \hline \vdots & \vdots \\ \hline I_N & Q_N \\ \hline \end{array} * \begin{array}{|c|c|} \hline h'_1 & h''_1 \\ \hline h'_2 & h''_2 \\ \hline h'_3 & h''_3 \\ \hline \vdots & \vdots \\ \hline h'_M & h''_M \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline \uparrow & \uparrow & \uparrow \\ I * h' & I * h'' + Q * h' & Q * h'' \\ \downarrow & \downarrow & \downarrow \\ \hline \end{array}$$

$$X = (I + jQ) * (h' + jh'') = (I * h' - Q * h'') + j(I * h'' + Q * h').$$

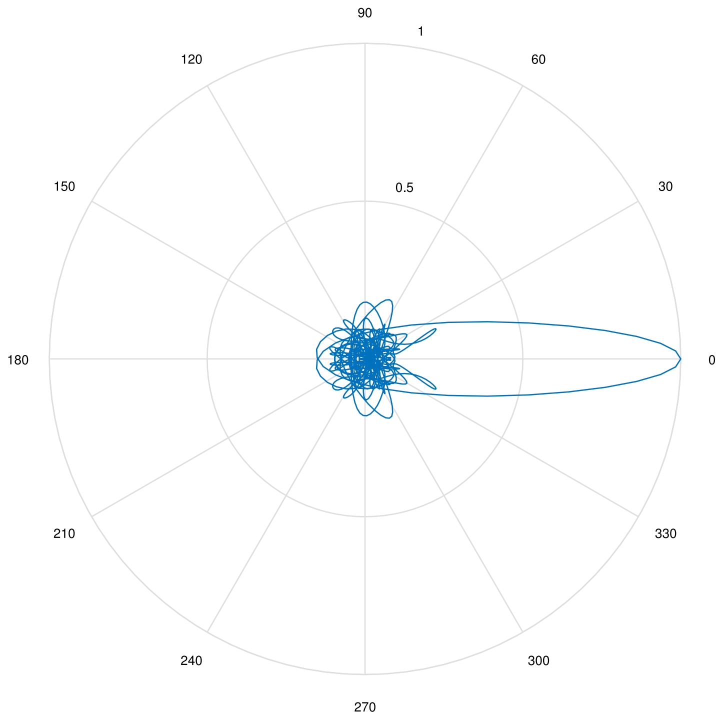
$$X = \begin{array}{|c|c|} \hline \uparrow & \uparrow \\ I * h' - Q * h'' & I * h'' + Q * h' \\ \downarrow & \downarrow \\ \hline \end{array}$$

$$X = X_{\text{naïve}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}$$

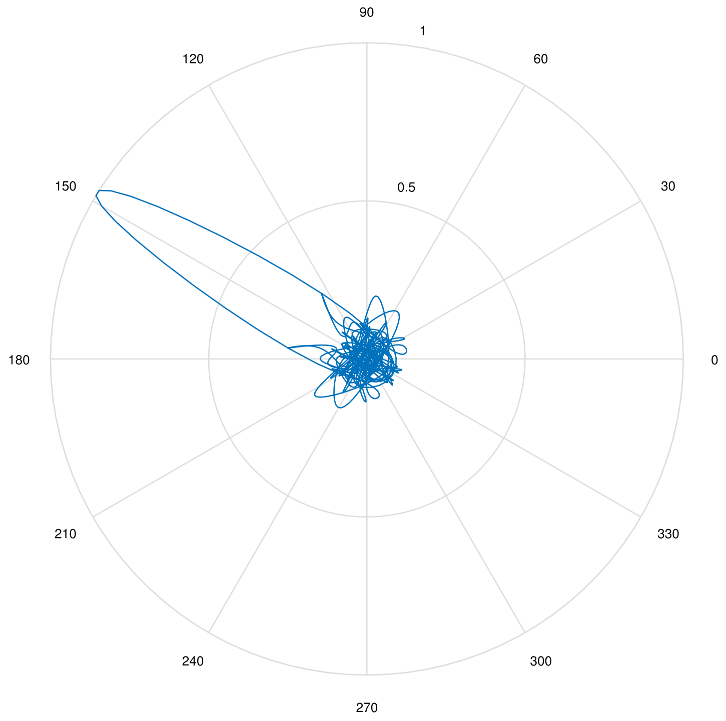
Complex-Valued Correlation

- Note that even if feature phase is off, there is a large (in complex magnitude) peak!
- How to generalize the non-linearity from the real case?

Convolutional Output w/ Phase Alignment

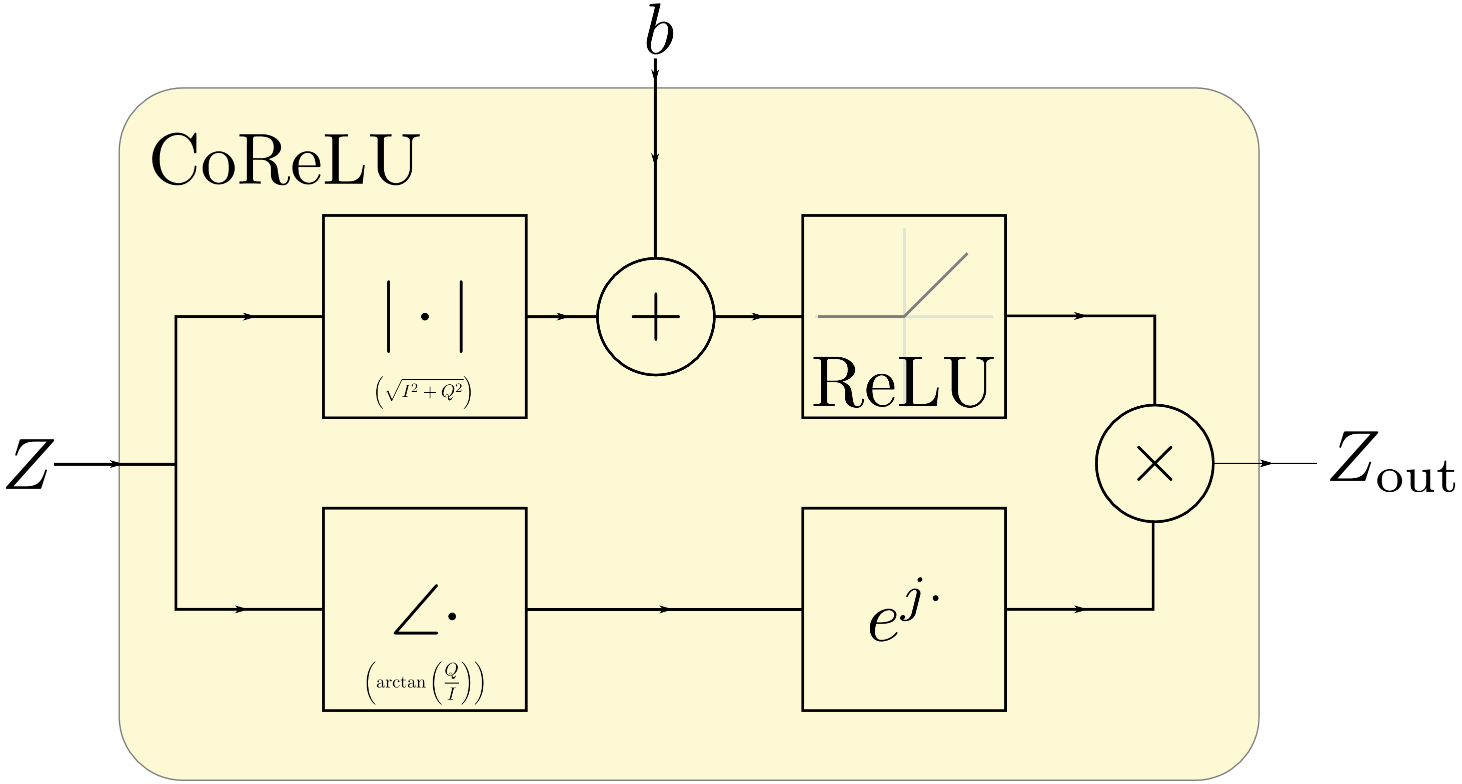


Convolutional Output w/o Phase Alignment

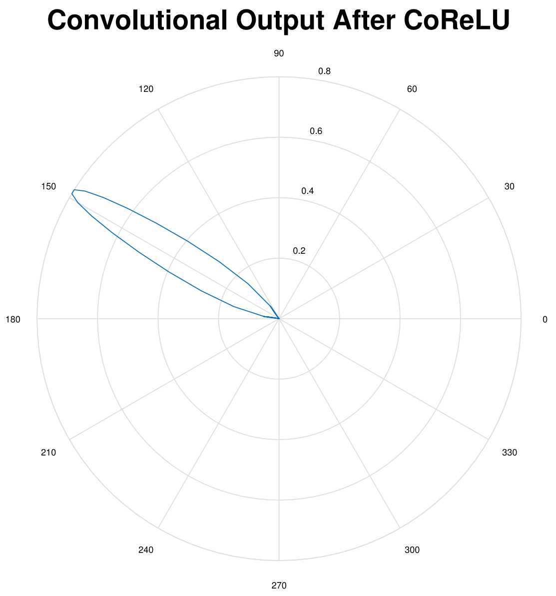


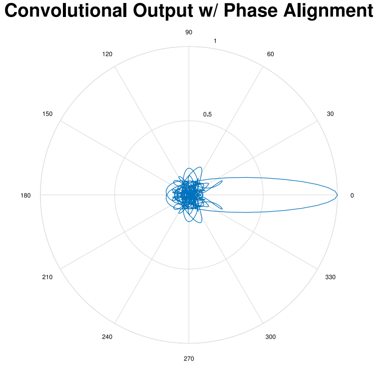
New Activation Function: CoReLU

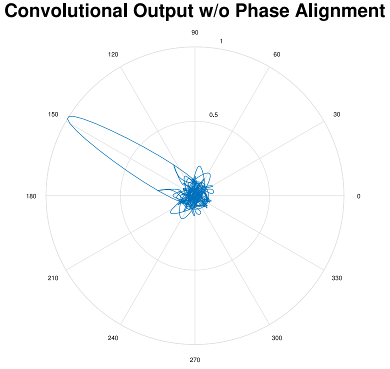
- Keep the phase
- ReLU on the magnitude
- Recombine



$$\text{CoReLU}(Z, b) = \text{ReLU}(|Z| + b)e^{j\angle Z}$$







New Activation Function: CoReLU

- Throws away small correlations, just like in the real-valued case
- This is a I/Q space detection that keeps the phase information
- Maybe that is important further down the line to detect relationships between snippets

ML for Signals: New Ideas

- ~~Convoluted~~ Convolutional neural nets for complex-valued signals
  - Using real-valued packages (Tensorflow)
  - I will freely say convolution or correlation, they are equivalent when the weights are discovered through optimization
- New activation functions
  - CoReLU
- Complex max-pooling
- Learned linear transformations (LLT)
  - Update the weights in the linear part
  - Helps answer “which domain is best”

Learned Linear Transforms

- DFT is a linear transformation
  - Time to frequency
  - N samples in
  - N samples out
- What is the best basis?
  - Time?
  - Frequency?
  - Some wavelets?
- Space of transforms is infinite!
- Learn this transform from the data!

$$\mathbf{W} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

$$\omega = e^{-j2\pi/N}$$

$$\tilde{Z} = \mathbf{W}Z$$

Learned Linear Transforms: Keras Recipe

```
from keras.engine.topology import Layer
...
class LLT(Layer):
        ...
        def build(self, input_shape):
                self.W = self.add_weight(name=‘W’,
                shape=...,
                initializer=..., trainable=True)
                ...
        def call(self, x):
                return K.dot(x, self.W)
        ...
(some Reshape layers to make it all work)
```

Learned Linear Transforms

- Different initial conditions before optimizer runs
  - Glorot
  - Uniform
  - Identity matrix – start with time representation
  - DFT matrix – start with frequency representation

GTRI ASCC Result

- We put all that together and...
- Congrats to 1) Platypus Aerospace, 2) TeamAu, and 3) Deep Dreamers!
- GTRI (YellowJackets) placed 15 out of 49 scored entries
  - Most of our submissions were using the hand-tuned networks
  - Hand-tuned team was able to iterate faster due to better hardware

ASCC Final Notes

- Scoring metric was strange
  - Log-loss is great for optimizing/learning, not as telling for performance
  - Diagonally-ness of the confusion matrix might be better?
  - Top five accuracy?
- Human speech has lots of silence
  - This means that things like human speech over FM can look like just an unmodulated carrier a lot of the time, especially when there is only 1024 samples collected at megasamples per second

Final Thoughts

- ML for signals / radio is really fun
- Go download TensorFlow and Keras
  - You do not have to have a PhD in ML to use these tools
  - If you can GNURadio, you can ML
- Lots of low-hanging fruit still in this area
  - Just by applying what has worked in computer vision, you can probably crank out state-of-the-art results (there is not much published here)

Questions?