End-to-End Learning of Communications Systems Without a Channel Model

Snehith Aithu EE16BTECH11041

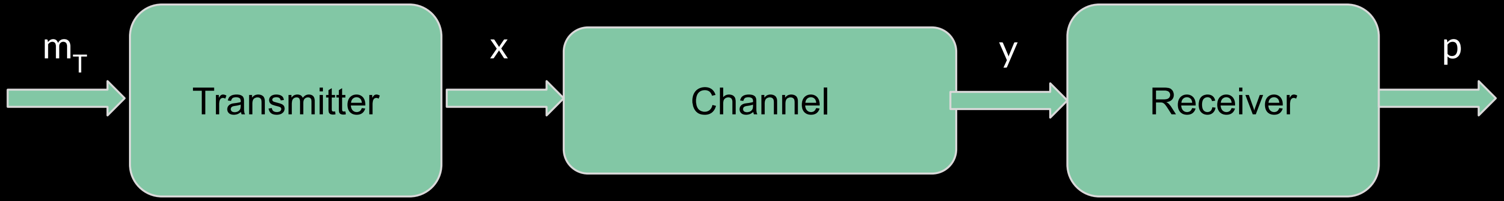
Sumanth Kumar EE16BTECH11009

Anand N Warrier EE16BTECH11042

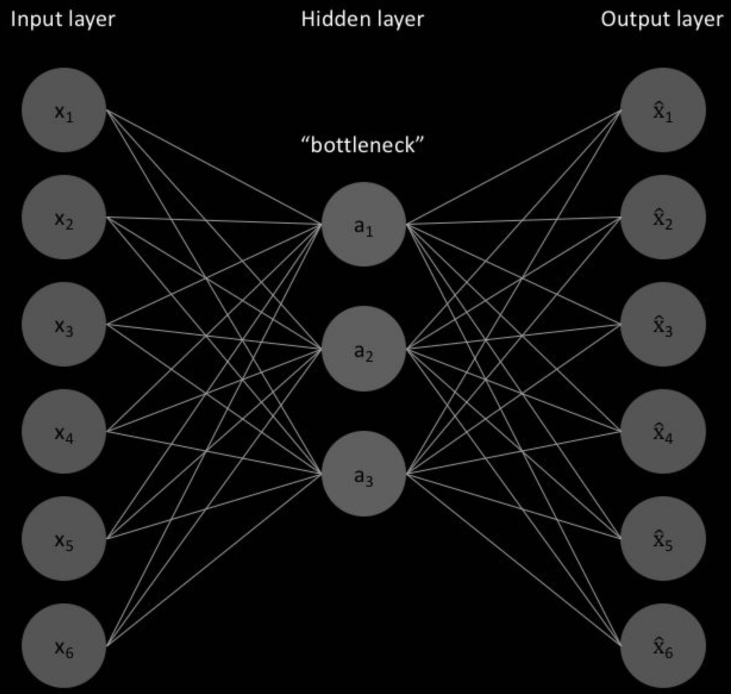
Motivation

- In end-to-end learning of communication systems, our objective is to learn the transmitter and receiver model for a performance metric and channel model.

- If we can somehow learn the model, we can find the message which was transmitted given the received signal.



Previous approaches



- An approach was to represent transmitter and receiver as Neural Networks and considering the whole as an autoencoder.
- Extensions:
    - Extending to joint source-channel coding
    - Using a policy gradient for a non-differentiable receiver which treats detection as a clustering problem

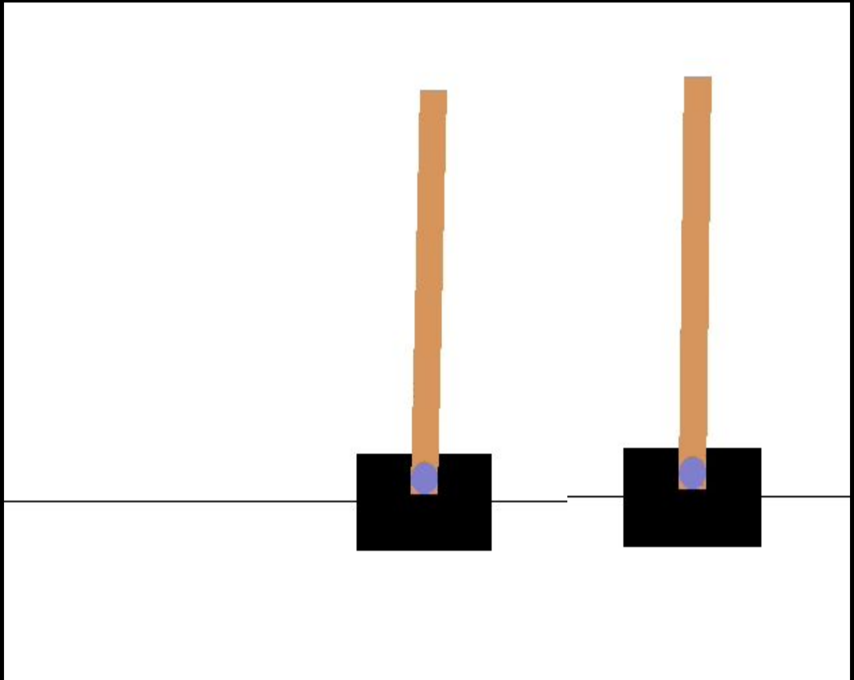
Drawbacks

- A channel model, or the gradient of the instantaneous channel transfer function, must be known.
- But, the channel is generally a black box.
- So, In general, fine-tuning of the receiver is done after initial learning.
- But, Transmitter is not fine-tuned.

Our Approach

- Reinforcement Learning provides a theoretical foundation for obtaining an estimate of the gradient of an arbitrary loss function with respect to w.r.t. actions taken by an agent.
- Knowledge of the channel model and the instantaneous channel transfer function is not needed.
- Our approach iterates between two phases:
    - supervised training of the receiver
    - RL-based training of the transmitter based on an estimated gradient of the loss given by the receiver.

Reinforcement Learning



- Reinforcement Learning aims at making suitable action to maximize reward in a particular situation.
- The model will return a state and the user will decide to reward or punish the model based on its output.
- In Supervised Learning, we have the answer key in the training data.
- In RL, there is no answer but the agent decides what to do so as to achieve the goal.

Better with Example


180 2 1

Real Life Example


Reinforcement Learning
First trial...
TMC

Policy

- Let $s \in S$ take an action $a \in A$ according to some policy π and per-sample loss *l*.
- Expected per-sample loss is given by:

$$\mathcal{L}(s, a) = \mathbb{E}\left[l|s, a\right]$$

- We need to find:

$$\operatorname{argmin}_\pi L(s, a)$$

- The training in RL is just like a try-and-fail process.
- Usually, π is chosen as stochastic. So, we have to minimize:-

$$J(s, \pi) = \int_{a \in \mathcal{A}} \pi(a|s)\mathcal{L}(s, a)\, da.$$

Policy Learning

- A policy π chosen can be a Gaussian policy
- A Gaussian policy i.e., $\pi_\psi(.|\boldsymbol{s})$ is a probability distribution over a action space conditioned on a state space ***s*** with parameters vector ψ .
- Let's consider a Gaussian policy and agent optimizes the policy using SGD approach where loss gradient is given by

$$\begin{aligned}
\nabla_{\boldsymbol{\psi}} J(s, \pi_{\boldsymbol{\psi}}) &= \int_{a \in \mathcal{A}} \mathcal{L}(s, a) \nabla_{\boldsymbol{\psi}} \pi_{\boldsymbol{\psi}}(a|s) \, da \\
&= \int_{a \in \mathcal{A}} \pi_{\boldsymbol{\psi}}(a|s) \mathcal{L}(s, a) \nabla_{\boldsymbol{\psi}} \log \left(\pi_{\boldsymbol{\psi}}(a|s) \right) \, da \\
&= \mathbb{E}_{\pi_{\boldsymbol{\psi}}} \left[\mathcal{L}(s, a) \nabla_{\boldsymbol{\psi}} \log \left(\pi_{\boldsymbol{\psi}}(a|s) \right) \middle| s \right]
\end{aligned}$$

Why all this for us?

- In our case agent is the transmitter and reward is loss provided by the receiver
- The message set **M** corresponds to the state space
- Encoded message set $\mathbf{C}^N$ corresponds to the action space
- The model can be trained from pure observations alone without any knowledge of the underlying channel model
- Policy learning technique is helpful in RL- based training of the transmitter based on an estimated gradient of the loss

Approach in brief

- Algorithm for end-to-end training iterates between two phases
  1. Supervised training at the receiver
  2. RL-based training at the transmitter
- Hence it is also known as *alternating training algorithm*
- Training is done on additive white Gaussian noise(AWGN) and Raleigh block-fading (RBF) channels
- The number of iterations carried out for training can depend on some stop criterion, e.g., stop when no more significant progress is observed

Problem Formulation

- As channel acts as a stochastic system, output y follows a probability distribution conditioned on its input x, i.e., $y \sim P(y|x)$.
- Implement transmitter and receiver as two separate parametric functions that can be jointly optimized to meet specific performance requirements
- Choose two NNs as parametric functions which are differentiable and are denoted by $f^{(T)}_{\boldsymbol{\theta}_T}, f^{(R)}_{\boldsymbol{\theta}_R}$

Parametric Function of Tx NN

- Let's assume a NN of K layers
- Parametric function can be written as $f_{\boldsymbol{\theta}} : \mathbb{R}^{N_0} \rightarrow \mathbb{R}^{N_K}$ which maps an input vector $r_0 \in \mathbf{R}^{No}$ to an output vector $r_K \in \mathbf{R}^{Nk}$ through K layers
- In our case $N_0 = |M|$ and $r_K \in \mathbf{C}^N$ where $|M|$ is cardinality of message set and N is the number of channel uses
- The transmitter is represented by

$$f_{\boldsymbol{\theta}_T}^{(T)} : \mathbb{M} \rightarrow \mathbb{C}^N$$

Parametric equation of Rx NN

- Job of the receiver is to receive output from channel and return a probability vector on message set conditioned on received signal
- The receiver is implemented by

  $$f_{\boldsymbol{\theta}_R}^{(R)} : \mathbb{C}^N \to \left\{ \mathbf{p} \in \mathbb{R}_+^M \mid \sum_{i=1}^{M} p_i = 1 \right\}$$

  where $\boldsymbol{\theta}_R$ is the set of parameters and **p** a probability vector

Problem Formulation

- Purpose of receiver is to predict actual message *m* given **y** by estimating the conditional probability *P(m|***y***)*
- Can be done by learning the conditional log-likelihood estimator

$$\boldsymbol{\theta}_R^* = \arg\min_{\boldsymbol{\theta}_R} L(\boldsymbol{\theta}_R)$$

where *L* is the cross-entropy (CE) defined as

$$L(\boldsymbol{\theta}_R) = \frac{1}{S}\sum_{i=1}^{S} \underbrace{-\log\left(\left[f_{\boldsymbol{\theta}_R}^{(R)}(\mathbf{y}^{(i)})\right]_{m^{(i)}}\right)}_{l^{(i)}}$$

- *L* assumes that the training examples are i.i.d samples, *S* is the size of the training set,
- $m^{(i)}$ is the *i*th training example, $l^{(i)}$ is the *per-example loss*, and $\mathbf{y}^{(i)}$ is the corresponding received signal

Training Process Overview

- we implement transmitter and receiver by two different parametric functions that can be independently optimized.
- Assumption: Transmitter and Receiver have access to a sequence of training samples.
- Two Training models and Architectures:
  - Receiver Training
  - Transmitter Training
- Alternating Training : Algorithm alternates between training of receiver and transmitter

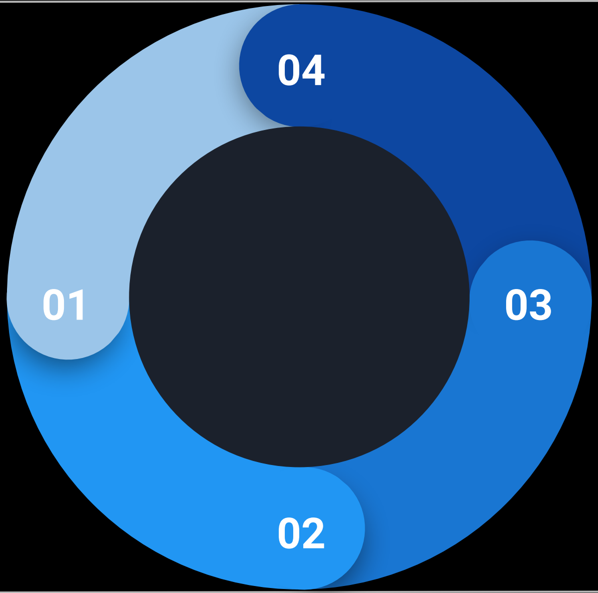
Training Overview

Fix Transmitter

Encode the signal using Tx model and send to receiver

Train Receiver

Decode the signal , find error and update parameters of Rx using SGD Algorithm



Train Transmitter

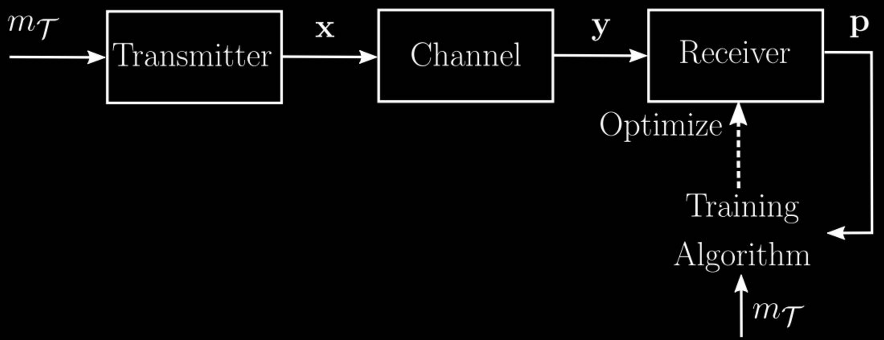
update the Tx parameters using Policy learning and SGD to minimise the Rx loss

Fix Receiver

For a Decoded signal Pass the error in receiver to Tx without training the Rx

Receiver Training

- Supervised Learning
- Involves 3 steps:
    a. Probability Distribution Generation over $\mathbf{M}$
    b. Computing Loss (CE)
    c. Optimising (SGD)

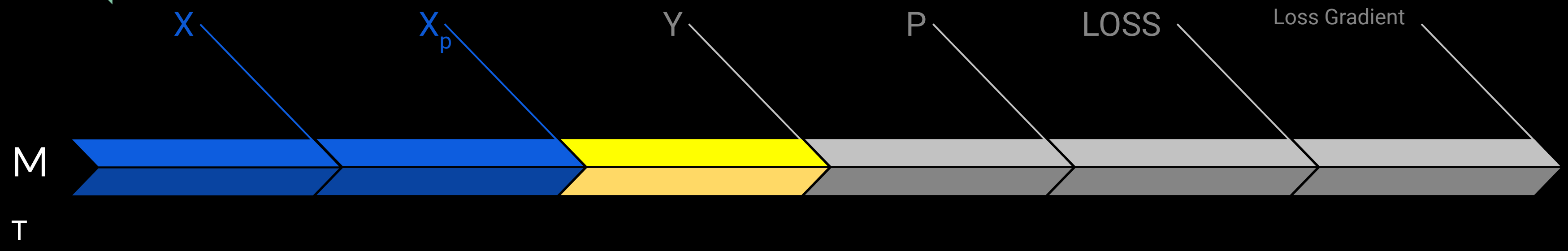


function TRAINRECEIVER()
    repeat
        $\triangleright$ Transmitter:
        $\mathbf{m}_{\mathcal{T}} \leftarrow \textsc{TrainingSource}(B_R)$
        $\mathbf{X} \leftarrow f^{(T)}_{\boldsymbol{\theta}_T}(\mathbf{m}_{\mathcal{T}})$
        $\textsc{Send}(\mathbf{X})$
        $\triangleright$ Receiver:
        $\mathbf{Y} \leftarrow \textsc{Receive}()$
        $\mathbf{P} \leftarrow f^{(R)}_{\boldsymbol{\theta}_R}(\mathbf{Y})$
        $\mathbf{m}_{\mathcal{T}} \leftarrow \textsc{TrainingSource}()$
        $\mathrm{SGD}(\boldsymbol{\theta}_R, \mathbf{m}_{\mathcal{T}}, \mathbf{P})$
    until Stop criterion is met
end function

Transmitter Training

- Objective : To generate channel symbols that minimize a scalar loss provided by the receiver.
- Reinforcement Learning Approach
- State Space : $\mathbf{M}$ Action Space : $\mathbf{C}^{\mathbf{N}}$
- Stochastic RL Policy(enables exploration) : $\mathbf{x}_p \sim \pi_\psi(\cdot|\mathbf{x})$
- Loss is an indication of the end-to-end performance and depends on the channel dynamics.

Transmitter Training



Message Encoding

The message is feeded to a NN and is transformed into channel symbols

Policy Sampling

stochastic policy is then sampled to generate the channel symbols

Channel Model

Passing through AWGN or RBF channel

Decoding

Generating probability distributions over M by passing through Neural network and softmax layer in receiver

Computing Loss

The Cross-entropy loss will be computed between the generated probability distributions and ground truth and are sent to transmitter

Optimisation step

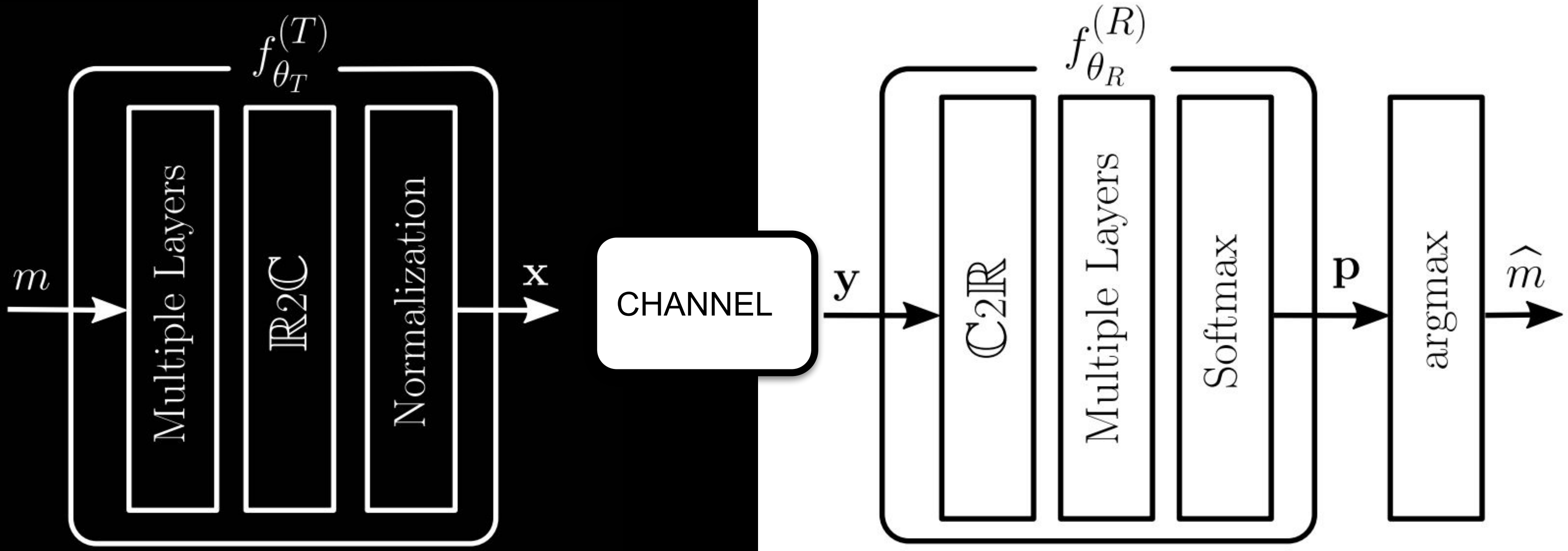
The loss gradients are computed and parameters are updated using SGD

Loss Gradient :

$$\nabla_{\boldsymbol{\theta}_T,\psi} \widetilde{J}(\mathbf{m}_{\mathcal{T}}, \mathbf{l}, \mathbf{X}_p) = \frac{1}{B_T} \sum_{i=1}^{B_T} l^{(i)} \nabla_{\boldsymbol{\theta}_T,\psi} \log\left(\pi_{\psi}\left(\mathbf{x}_p^{(i)} | f_{\boldsymbol{\theta}_T}^{(T)}(m_{\mathcal{T}}^{(i)})\right)\right)$$

Transmitter Architecture

Receiver Architecture



Transmitter Training

- Signal To Noise Ratio

$$\mathrm{SNR} = \frac{\mathbb{E}\left[\frac{1}{N}\|\mathbf{x}\|_2^2\right]}{\sigma^2}$$

- Normalisation layer ensures $\mathbb{E}\left[\frac{1}{N}\|\mathbf{x}\|_2^2\right]=1$
- RL exploration is performed by adding a zero-mean complex normal perturbation w to the output of the transmitter.

$$\mathbf{x}_p = \sqrt{1-\sigma_\pi^2}\mathbf{x} + \mathbf{w} \;\; \text{where} \;\; \mathbf{w} \sim \mathcal{CN}(\mathbf{0}, \sigma_\pi^2 \mathbf{I})$$

Implementation

- Implemented on two channels:
  - AWGN : Additive White Gaussian Noise channel
  - RBF : Rayleigh Block-Fading channel

- Optimizer : Adam
- SNR:
  - AWGN : 10 dB
  - RBF : 20 dB

- Size of $\mathbf{M}$: 256
- N (number of channels) : 4
- σ_{π}^{2} : 0.02

Implementation - RL Policy

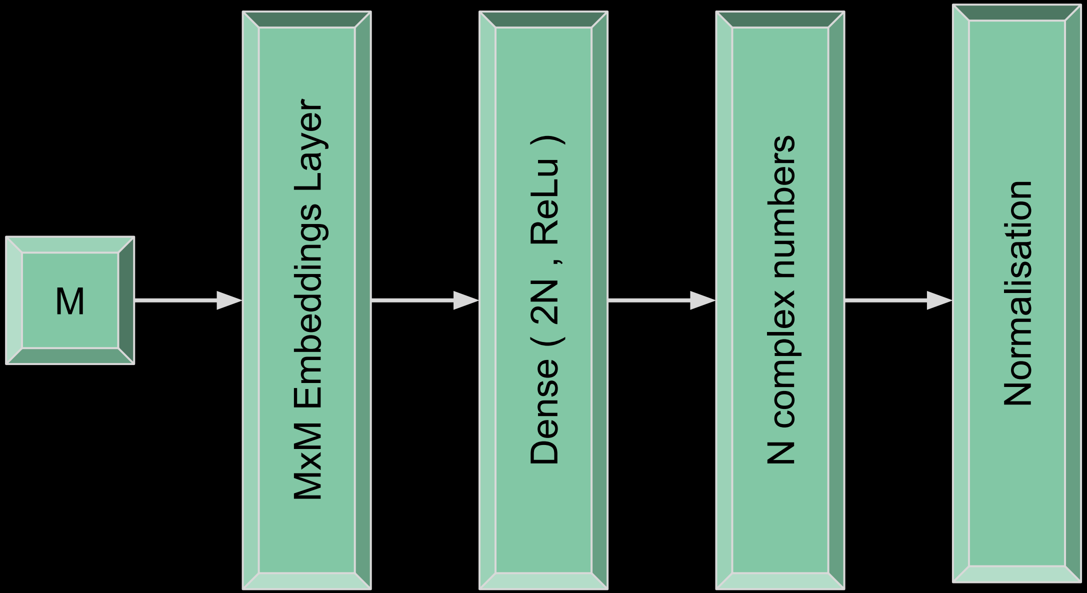
- Transmitter RL policy π_ψ : Guassian
  - Mean : $\sqrt{1-\sigma_\pi^2} f_{\boldsymbol{\theta}_T}^{(T)}$
  - Co-Variance : $\sigma_\pi^2 \mathbf{I}$
- Policy Computing:

$$\pi(\mathbf{x}_p \mid f_{\boldsymbol{\theta}_T}^{(T)}(m)) = \frac{1}{(\pi\sigma_\pi^2)^N} \exp\left(-\frac{\|\mathbf{x}_p - \sqrt{1-\sigma_\pi^2} f_{\boldsymbol{\theta}_T}^{(T)}(m)\|_2^2}{\sigma_\pi^2}\right)$$

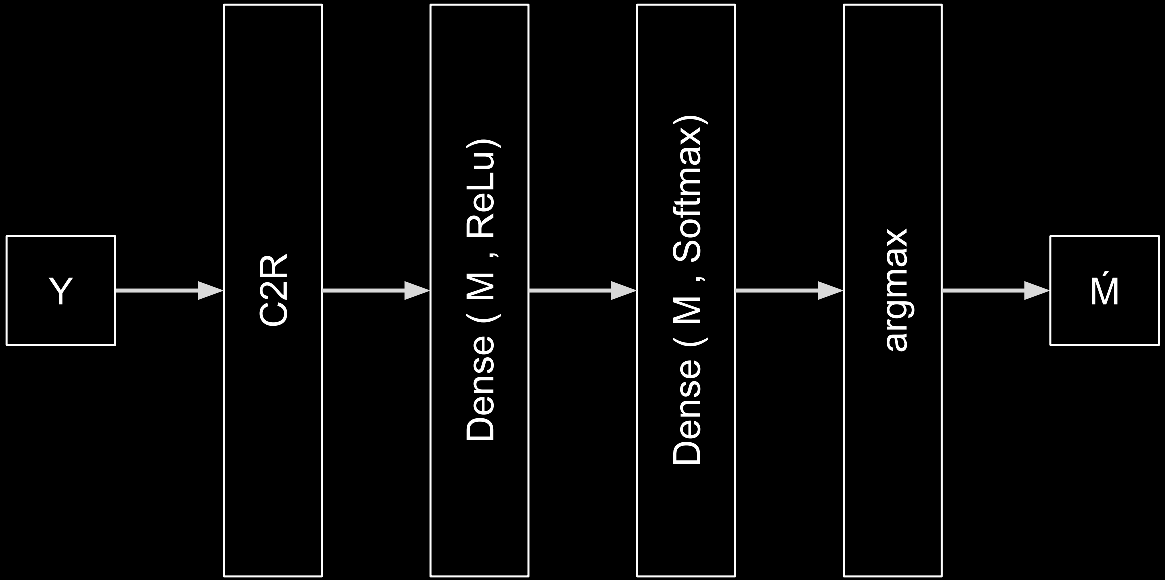
- Log Gradient :

$$\nabla_{\boldsymbol{\theta}_T} \log\left(\pi(\mathbf{x}_p \mid m)\right) = \frac{2\sqrt{1-\sigma_\pi^2}}{\sigma_\pi^2} \left(\nabla_{\boldsymbol{\theta}_T} f_{\boldsymbol{\theta}_T}^{(T)}(m)\right)^{\mathsf{T}} \left(\mathbf{x}_p - \sqrt{1-\sigma_\pi^2} f_{\boldsymbol{\theta}_T}^{(T)}(m)\right)$$

Implementation - Transmitter

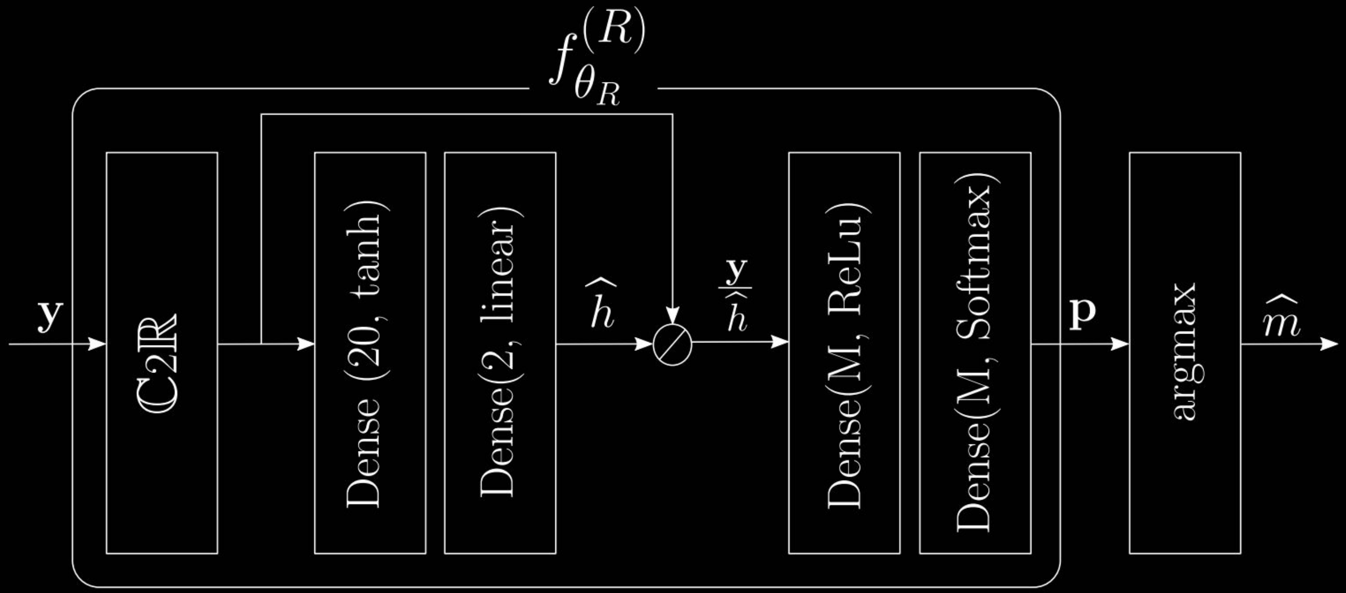


Implementation - Receiver



Receiver architecture for AWGN channels

Implementation - Receiver



Receiver architecture for RBF channels

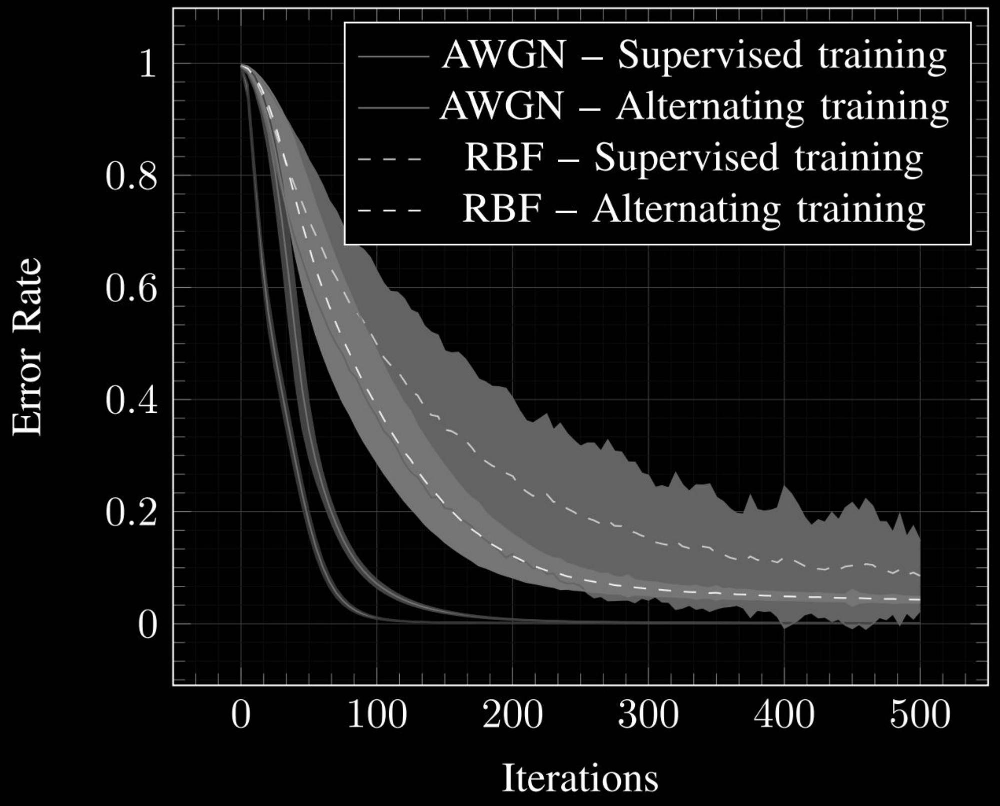
Results

- AWGN channel:

Supervised method leads to faster convergence compared to the alternating method.

- RBF channel:

Alternating method enables faster convergence, and with significantly less variance.



References

- https://arxiv.org/pdf/1804.02276.pdf
- T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," IEEE Trans. on Cogn. Commun. Netw., vol. 3, no. 4, pp. 563–575, Dec. 2017.
- https://deepmind.com/
- https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146

  A blog on Policy gradient method

Thank You