Learning to Detect

Presentor: Zijian Qin

Nov. 1 2024

Table of Contents

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

MIMO Detection Formulation

Linear MIMO Model

$$\bar{\mathbf{y}} = \bar{\mathbf{H}}\bar{\mathbf{x}} + \bar{\mathbf{w}}$$

$\bar{\mathbf{y}} \in \mathbb{C}^{\mathrm{N}}$: observation vector
$\bar{\mathbf{H}} \in \mathbb{C}^{\mathrm{N}\times\mathrm{K}}$: linear transformation matrix
$\bar{\mathbf{x}} \in \bar{\mathbb{S}}^{\mathrm{K}}$: original vector
$\bar{\mathbf{w}} \in \mathbb{C}^{\mathrm{N}}$: noise vector
$\bar{\mathbb{S}}$: finite discrete constellation set

MIMO Detection Reformulation

Equivalent Model without Complex Numbers

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w}$$

$$\mathbf{y} = \begin{bmatrix} \Re(\bar{\mathbf{y}}) \\ \Im(\bar{\mathbf{y}}) \end{bmatrix} \in \mathbb{R}^{2\mathrm{N}}\text{: observation vector}$$

$$\mathbf{H} = \begin{bmatrix} \Re(\bar{\mathbf{H}}) & -\Im(\bar{\mathbf{H}}) \\ \Im(\bar{\mathbf{H}}) & \Re(\bar{\mathbf{H}}) \end{bmatrix} \in \mathbb{R}^{2\mathrm{N}\times 2\mathrm{K}}\text{: linear transformation matrix}$$

$$\mathbf{x} = \begin{bmatrix} \Re(\bar{\mathbf{x}}) \\ \Im(\bar{\mathbf{x}}) \end{bmatrix} \in \mathbb{S}^{2\mathrm{K}}\text{: original vector}$$

$$\mathbf{w} = \begin{bmatrix} \Re(\bar{\mathbf{w}}) \\ \Im(\bar{\mathbf{w}}) \end{bmatrix} \in \mathbb{R}^{2\mathrm{N}}\text{: noise vector}$$

$\mathbb{S}$: finite discrete constellation set

Maximum Likelihood Estimation

Assume AWGN channel: $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$

Maximum Likelihood Estimation

Assume AWGN channel: $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$
$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w}$, $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{H}\mathbf{x}, \sigma^2 \mathbf{I})$

Maximum Likelihood Estimation

Assume AWGN channel: $\mathbf{w} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w}$, $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{H}\mathbf{x}, \sigma^2\mathbf{I})$

The likelihood function of observing $\mathbf{y}$ given $\mathbf{x}$

$$\mathrm{L}(\mathbf{x}) = \mathrm{p}(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi\sigma^2\mathbf{I})^{2\mathrm{N}/2}} \exp\left(-\frac{1}{2\sigma^2\mathbf{I}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2\right)$$

Maximum Likelihood Estimation

Assume AWGN channel: $\mathbf{w} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w},\ \mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{H}\mathbf{x}, \sigma^2\mathbf{I})$

The likelihood function of observing $\mathbf{y}$ given $\mathbf{x}$

$$\mathrm{L}(\mathbf{x}) = \mathrm{p}(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi\sigma^2\mathbf{I})^{2\mathrm{N}/2}} \exp\left(-\frac{1}{2\sigma^2\mathbf{I}}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2\right)$$

Log-Likelihood Function

$$\log \mathrm{L}(\mathbf{x}) = -\mathrm{N}\log(2\pi\sigma^2\mathbf{I}) - \frac{1}{2\sigma^2\mathbf{I}}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$$

Maximum Likelihood Estimation

Assume AWGN channel: $\mathbf{w} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w}$, $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\mathbf{H}\mathbf{x}, \sigma^2\mathbf{I})$

The likelihood function of observing $\mathbf{y}$ given $\mathbf{x}$

$$\mathrm{L}(\mathbf{x}) = \mathrm{p}(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi\sigma^2\mathbf{I})^{2\mathrm{N}/2}} \exp\left(-\frac{1}{2\sigma^2\mathbf{I}}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2\right)$$

Log-Likelihood Function

$$\log \mathrm{L}(\mathbf{x}) = -\mathrm{N}\log(2\pi\sigma^2\mathbf{I}) - \frac{1}{2\sigma^2\mathbf{I}}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$$

Maximize $\mathrm{L}(\mathbf{x}) \rightarrow$ maximize $\log \mathrm{L}(\mathbf{x}) \rightarrow$ minimize $\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$

One-hot Encoding

$$\mathbb{S} = \{s_1, s_2, \ldots, s_{|\mathbb{S}|}\}$$

One-hot Encoding

$$s_i \leftrightarrow \mathbf{u}_i \; (1 \leq i \leq |\mathbb{S}|)$$

$$\mathbf{u}_i \in \{0, 1\}^{|\mathbb{S}|}, \; |\mathbf{u}_i| = 1$$

One-hot Encoding

$$\mathbb{S} = \{s_1, s_2, \ldots, s_{|\mathbb{S}|}\}$$

One-hot Encoding

$$s_i \leftrightarrow \mathbf{u}_i \ (1 \leq i \leq |\mathbb{S}|)$$

$$\mathbf{u}_i \in \{0,1\}^{|\mathbb{S}|}, \ |\mathbf{u}_i| = 1$$

Examples

In 16QAM, $\mathbb{S} = \{-3, -1, 1, 3\}$

$s_1 = -3 \leftrightarrow \mathbf{u}_1 = \{1, 0, 0, 0\}$

$s_2 = -1 \leftrightarrow \mathbf{u}_2 = \{0, 1, 0, 0\}$

$s_3 = 1 \leftrightarrow \mathbf{u}_3 = \{0, 0, 1, 0\}$

$s_4 = 3 \leftrightarrow \mathbf{u}_4 = \{0, 0, 0, 1\}$

One-hot Encoding

$\mathbf{x} \leftrightarrow \mathbf{x}_{\mathrm{oh}}$, one-hot encoding by element

$\mathbf{x} \in \mathbb{S}^{\mathrm{K}}$, $\mathbf{x}_{\mathrm{oh}} \in \{0, 1\}^{\mathrm{K}|\mathbb{S}|}$

$\mathbf{x} = \mathbf{f}_{\mathrm{oh}}(\mathbf{x}_{\mathrm{oh}})$

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Basic Machine Learning Components

- Model function: $\hat{\mathbf{x}}_{\mathrm{oh}}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})$
- Loss function: $\min_{\boldsymbol{\theta}} \mathrm{E}\left\{\mathrm{l}\left(\mathbf{x}_{\mathrm{oh}}; \hat{\mathbf{x}}_{\mathrm{oh}}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})\right)\right\}$
- Non-linear unit: $\rho(\mathrm{x}) = \max\{0, \mathrm{x}\}$

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Decoder Design

Iteration Process

$$\mathbf{q}_1 = \mathbf{y}$$

$$\mathbf{q}_{\mathrm{k}+1} = \rho\left(\mathbf{W}_{\mathrm{k}}\mathbf{q}_{\mathrm{k}} + \mathbf{b}_{\mathrm{k}}\right)$$

$$\hat{\mathbf{x}}_{\mathrm{oh}} = \mathbf{W}_{\mathrm{L}}\mathbf{q}_{\mathrm{L}} + \mathbf{b}_{\mathrm{L}}$$

$$\hat{\mathbf{x}} = \mathrm{f}_{\mathrm{oh}}\left(\hat{\mathbf{x}}_{\mathrm{oh}}\right)$$

- parameters: $\boldsymbol{\theta} = \{\mathbf{W}_{\mathrm{k}}, \mathbf{b}_{\mathrm{k}}\}_{\mathrm{k}=1}^{\mathrm{L}}$
- loss function: $\mathrm{l}\left(\mathbf{x}_{\mathrm{oh}}; \hat{\mathbf{x}}_{\mathrm{oh}}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})\right) = \left\|\mathbf{x}_{\mathrm{oh}} - \hat{\mathbf{x}}_{\mathrm{oh}}\right\|^2$

Decoder Architecture

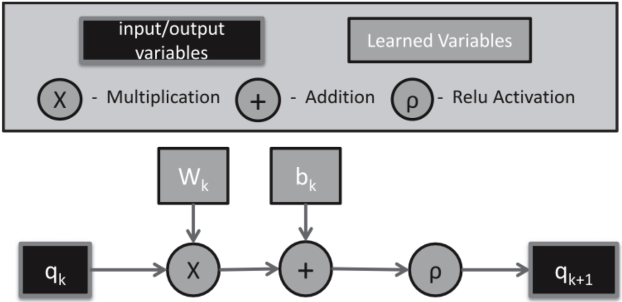

Figure: Fully connected network

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Projected Gradient Descent

Definition

Iterative projected gradient descent is an optimization method to solve optimization problems where the solution is required to stay in a feasible set.

Projected Gradient Descent

Definition

Iterative projected gradient descent is an optimization method to solve optimization problems where the solution is required to stay in a feasible set.

- Initialize: x_0 s.t. $\mathrm{x}_0 \in C$

Projected Gradient Descent

Definition

Iterative projected gradient descent is an optimization method to solve optimization problems where the solution is required to stay in a feasible set.

- Initialize: x_0 s.t. $\mathrm{x}_0 \in C$
- Iterate: $\mathrm{x}'_{\mathrm{k}+1} = \mathrm{x}_\mathrm{k} - \alpha \nabla \mathrm{f}(\mathrm{x})\Big|_{\mathrm{x}=\mathrm{x}_\mathrm{k}}$

Projected Gradient Descent

Definition

Iterative projected gradient descent is an optimization method to solve optimization problems where the solution is required to stay in a feasible set.

- Initialize: x_0 s.t. $\mathrm{x}_0 \in C$
- Iterate: $\mathrm{x}'_{\mathrm{k+1}} = \mathrm{x}_\mathrm{k} - \alpha \nabla \mathrm{f}(\mathrm{x})\Big|_{\mathrm{x}=\mathrm{x}_\mathrm{k}}$
- Projection: $\mathrm{x}_{\mathrm{k+1}} = \mathrm{Proj}_C(\mathrm{x}'_{\mathrm{k+1}})$

Projected Gradient Descent

Definition

Iterative projected gradient descent is an optimization method to solve optimization problems where the solution is required to stay in a feasible set.

- Initialize: x_0 s.t. $\mathrm{x}_0 \in C$
- Iterate: $\mathrm{x}'_{k+1} = \mathrm{x}_k - \alpha \nabla \mathrm{f}(\mathrm{x})\Big|_{\mathrm{x}=\mathrm{x}_k}$
- Projection: $\mathrm{x}_{k+1} = \mathrm{Proj}_C(\mathrm{x}'_{k+1})$

α: learning rate
$\mathrm{f}(\cdot)$: objective function
Proj_C: projection operator, finds the nearest point in C to x'_{k+1}

Projected Gradient Descent

Example

$$\text{minimize } f(x) = \|ax - b\|^2, \text{ subject to } l \leq x \leq u$$

Projected Gradient Descent

Example

$$\text{minimize } f(x) = \|ax - b\|^2, \text{ subject to } l \leq x \leq u$$

- Initialize: $x_0 \in [l, u]$
- Iterate: $x'_{k+1} = x_k - \alpha \nabla f(x)\Big|_{x=x_k}$
- Project: $x_{k+1} = \min(\max(x'_{k+1}, l), u)$

Naive Decoder Design

- Objective function: $\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$

Naive Decoder Design

- Objective function: $\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$
- Initialize: $\hat{\mathbf{x}}_0$

Naive Decoder Design

- Objective function: $\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$
- Initialize: $\hat{\mathbf{x}}_0$
- Iterate:

$$\begin{aligned}\hat{\mathbf{x}}'_{k+1} &= \hat{\mathbf{x}}_k - \delta_k \nabla \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\ &= \hat{\mathbf{x}}_k - \delta_k \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k}\end{aligned}$$

Naive Decoder Design

- Objective function: $\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2$
- Initialize: $\hat{\mathbf{x}}_0$
- Iterate:

$$\begin{aligned}\hat{\mathbf{x}}'_{k+1} &= \hat{\mathbf{x}}_k - \delta_k \nabla \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\ &= \hat{\mathbf{x}}_k - \delta_k \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k}\end{aligned}$$

- Project: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}'_{k+1})$

Naive Decoder Design

Calculate the gradient descent:

$$\nabla \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} = \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k}$$

Naive Decoder Design

Calculate the gradient descent:

$$\nabla \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} = \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} = \frac{\partial (\mathbf{y} - \mathbf{H}\mathbf{x})^T (\mathbf{y} - \mathbf{H}\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k}$$

Naive Decoder Design

Calculate the gradient descent:

$$
\begin{aligned}
\nabla \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} &= \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial (\mathbf{y} - \mathbf{H}\mathbf{x})^T (\mathbf{y} - \mathbf{H}\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial (\mathbf{y}^T - \mathbf{x}^T \mathbf{H}^T)(\mathbf{y} - \mathbf{H}\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_k}
\end{aligned}
$$

Naive Decoder Design

Calculate the gradient descent:

$$
\begin{aligned}
\nabla \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{\mathrm{k}}} &= \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{\mathrm{k}}} \\
&= \frac{\partial (\mathbf{y} - \mathbf{H}\mathbf{x})^{\mathrm{T}} (\mathbf{y} - \mathbf{H}\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{\mathrm{k}}} \\
&= \frac{\partial (\mathbf{y}^{\mathrm{T}} - \mathbf{x}^{\mathrm{T}} \mathbf{H}^{\mathrm{T}}) (\mathbf{y} - \mathbf{H}\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{\mathrm{k}}} \\
&= \frac{\partial (\mathbf{y}^{\mathrm{T}} \mathbf{y} - \mathbf{y}^{\mathrm{T}} \mathbf{H}\mathbf{x} - \mathbf{x}^{\mathrm{T}} \mathbf{H}^{\mathrm{T}} \mathbf{y} + \mathbf{x}^{\mathrm{T}} \mathbf{H}^{\mathrm{T}} \mathbf{H}\mathbf{x})}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\hat{\mathbf{x}}_{\mathrm{k}}}
\end{aligned}
$$

Naive Decoder Design

Calculate the gradient descent:

$$
\begin{aligned}
\nabla \|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} &= \frac{\partial \|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial (\mathbf{y}-\mathbf{H}\mathbf{x})^T(\mathbf{y}-\mathbf{H}\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial (\mathbf{y}^T-\mathbf{x}^T\mathbf{H}^T)(\mathbf{y}-\mathbf{H}\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial (\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{H}\mathbf{x}-\mathbf{x}^T\mathbf{H}^T\mathbf{y}+\mathbf{x}^T\mathbf{H}^T\mathbf{H}\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= -2\mathbf{H}^T\mathbf{y}+2\mathbf{H}^T\mathbf{H}\mathbf{x}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k}
\end{aligned}
$$

Naive Decoder Design

Calculate the gradient descent:

$$
\begin{aligned}
\nabla\|\mathbf{y}-\mathbf{Hx}\|^2\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} &= \frac{\partial\|\mathbf{y}-\mathbf{Hx}\|^2}{\partial\mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial(\mathbf{y}-\mathbf{Hx})^T(\mathbf{y}-\mathbf{Hx})}{\partial\mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial(\mathbf{y}^T-\mathbf{x}^T\mathbf{H}^T)(\mathbf{y}-\mathbf{Hx})}{\partial\mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= \frac{\partial(\mathbf{y}^T\mathbf{y}-\mathbf{y}^T\mathbf{Hx}-\mathbf{x}^T\mathbf{H}^T\mathbf{y}+\mathbf{x}^T\mathbf{H}^T\mathbf{Hx})}{\partial\mathbf{x}}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= -2\mathbf{H}^T\mathbf{y}+2\mathbf{H}^T\mathbf{Hx}\Big|_{\mathbf{x}=\hat{\mathbf{x}}_k} \\
&= -2\mathbf{H}^T\mathbf{y}+2\mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}_k
\end{aligned}
$$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T \mathbf{y} + \delta_k \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_k)$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T \mathbf{y} + \delta_k \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_k)$
Optimization:

- gradient descent: $\mathbf{q}_k = \hat{\mathbf{x}}_{k-1} - \delta_{1k} \mathbf{H}^T \mathbf{y} + \delta_{2k} \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_{k-1}$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T\mathbf{y} + \delta_k \mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}_k)$
Optimization:

- gradient descent: $\mathbf{q}_k = \hat{\mathbf{x}}_{k-1} - \delta_{1k}\mathbf{H}^T\mathbf{y} + \delta_{2k}\mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}_{k-1}$
- lift dimension and non-linearize:
$$\mathbf{z}_k = \rho\left(\mathbf{W}_{1k}\begin{bmatrix}\mathbf{q}_k \\ \mathbf{v}_{k-1}\end{bmatrix} + \mathbf{b}_{1k}\right)$$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T \mathbf{y} + \delta_k \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_k)$

Optimization:

- gradient descent: $\mathbf{q}_k = \hat{\mathbf{x}}_{k-1} - \delta_{1k} \mathbf{H}^T \mathbf{y} + \delta_{2k} \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_{k-1}$
- lift dimension and non-linearize:
  $\mathbf{z}_k = \rho \left(\mathbf{W}_{1k} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{v}_{k-1} \end{bmatrix} + \mathbf{b}_{1k} \right)$
- estimate: $\hat{\mathbf{x}}_{oh,k} = \mathbf{W}_{2k} \mathbf{z}_k + \mathbf{b}_{2k}$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \boldsymbol{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T \mathbf{y} + \delta_k \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_k)$
Optimization:

- gradient descent: $\mathbf{q}_k = \hat{\mathbf{x}}_{k-1} - \delta_{1k} \mathbf{H}^T \mathbf{y} + \delta_{2k} \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_{k-1}$
- lift dimension and non-linearize:
$$\mathbf{z}_k = \rho \left(\mathbf{W}_{1k} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{v}_{k-1} \end{bmatrix} + \mathbf{b}_{1k} \right)$$
- estimate: $\hat{\mathbf{x}}_{oh,k} = \mathbf{W}_{2k} \mathbf{z}_k + \mathbf{b}_{2k}$
- one-hot decode: $\hat{\mathbf{x}}_k = f_{oh}(\hat{\mathbf{x}}_{oh,k})$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T \mathbf{y} + \delta_k \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_k)$
Optimization:

- gradient descent: $\mathbf{q}_k = \hat{\mathbf{x}}_{k-1} - \delta_{1k} \mathbf{H}^T \mathbf{y} + \delta_{2k} \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_{k-1}$
- lift dimension and non-linearize:
  $\mathbf{z}_k = \rho \left(\mathbf{W}_{1k} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{v}_{k-1} \end{bmatrix} + \mathbf{b}_{1k} \right)$
- estimate: $\hat{\mathbf{x}}_{oh,k} = \mathbf{W}_{2k} \mathbf{z}_k + \mathbf{b}_{2k}$
- one-hot decode: $\hat{\mathbf{x}}_k = f_{oh}(\hat{\mathbf{x}}_{oh,k})$
- state and memory: $\hat{\mathbf{v}}_k = \mathbf{W}_{3k} \mathbf{z}_k + \mathbf{b}_{3k}$

Decoder Design

Naive iteration: $\hat{\mathbf{x}}_{k+1} = \mathbf{\Pi}(\hat{\mathbf{x}}_k - \delta_k \mathbf{H}^T \mathbf{y} + \delta_k \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_k)$
Optimization:

- gradient descent: $\mathbf{q}_k = \hat{\mathbf{x}}_{k-1} - \delta_{1k} \mathbf{H}^T \mathbf{y} + \delta_{2k} \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_{k-1}$
- lift dimension and non-linearize: $\mathbf{z}_k = \rho \left(\mathbf{W}_{1k} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{v}_{k-1} \end{bmatrix} + \mathbf{b}_{1k} \right)$
- estimate: $\hat{\mathbf{x}}_{oh,k} = \mathbf{W}_{2k} \mathbf{z}_k + \mathbf{b}_{2k}$
- one-hot decode: $\hat{\mathbf{x}}_k = f_{oh}(\hat{\mathbf{x}}_{oh,k})$
- state and memory: $\hat{\mathbf{v}}_k = \mathbf{W}_{3k} \mathbf{z}_k + \mathbf{b}_{3k}$
- initialize: $\hat{\mathbf{x}}_0 = 0$, $\hat{\mathbf{v}}_0 = 0$

Parameters: $\boldsymbol{\theta} = \{\mathbf{W}_{1k}, \mathbf{b}_{1k}, \mathbf{W}_{2k}, \mathbf{b}_{2k}, \mathbf{W}_{3k}, \mathbf{b}_{3k}, \delta_{1k}, \delta_{2k}\}_{k=1}^{L}$

Decoder Design

Loss function: $l\left(\mathbf{x}_{oh}; \hat{\mathbf{x}}_{oh}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})\right) = \sum_{l=1}^{L} \log(l) \|\mathbf{x}_{oh} - \hat{\mathbf{x}}_{oh,l}\|^2$

- mean squared error: align with ML decoding
- logarithmic weight: prefer later iterations
- loss for each layer: avoid the vanishing gradient problem

Decoder Design

Loss function: $l\left(\mathbf{x}_{\mathrm{oh}};\hat{\mathbf{x}}_{\mathrm{oh}}(\mathbf{H},\mathbf{y};\boldsymbol{\theta})\right)=\sum_{l=1}^{L}\log(l)\|\mathbf{x}_{\mathrm{oh}}-\hat{\mathbf{x}}_{\mathrm{oh},l}\|^2$

- mean squared error: align with ML decoding
- logarithmic weight: prefer later iterations
- loss for each layer: avoid the vanishing gradient problem

Residual unit: $\hat{\mathbf{x}}_k=\alpha\hat{\mathbf{x}}_{k-1}+(1-\alpha)\hat{\mathbf{x}}_k$

- a weighted average with the output of the previous layer

Decoder Architecture

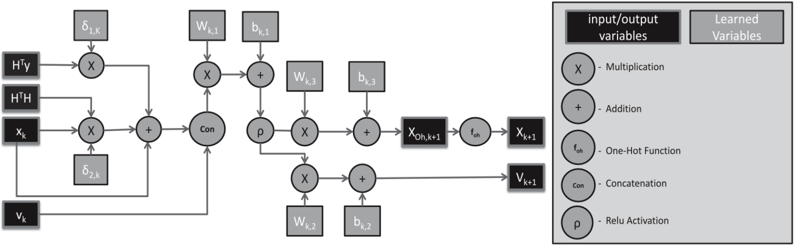

Figure: DetNet

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Soft Output

Scalar soft output: $P(x = s_i | y)$ $(s_i \in S, 1 \leq i \leq |S|)$

Soft Output

Scalar soft output: $P(x = s_i | y)$ $(s_i \in S, 1 \le i \le |S|)$

$$\left[P(x = s_1|y), P(x = s_2|y), \ldots, P(x = s_{|S|}|y) \right]^T$$
$$= P(x = s_1|y) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + P(x = s_2|y) \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + P(x = s_{|S|}|y) \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Soft Output

Scalar soft output: $P(x = s_i|y)$ $(s_i \in S, 1 \leq i \leq |S|)$

$$
\begin{aligned}
&\left[P(x = s_1|y), P(x = s_2|y), \ldots, P(x = s_{|S|}|y)\right]^T \\
&= P(x = s_1|y) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + P(x = s_2|y) \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + P(x = s_{|S|}|y) \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \\
&= E[\mathbf{x}_{oh}|y]
\end{aligned}
$$

Assump: the DetNet output $\hat{\mathbf{x}}_{oh}$ can achieve this expectation

Soft Output

Scalar soft output: $P(x = s_i | y)$ $(s_i \in S, 1 \leq i \leq |S|)$

$$\begin{aligned}
&\left[P(x = s_1|y), P(x = s_2|y), \dots, P(x = s_{|S|}|y)\right]^T \\
&= P(x = s_1|y) \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + P(x = s_2|y) \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + P(x = s_{|S|}|y) \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \\
&= E[\mathbf{x}_{oh}|y] \\
&= \arg\min_{\hat{\mathbf{x}}_{oh}} E\left[\|\mathbf{x}_{oh} - \hat{\mathbf{x}}_{oh}\|^2 \mid y\right]
\end{aligned}$$

Assump: the DetNet output $\hat{\mathbf{x}}_{oh}$ can achieve this expectation

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Channel Model

- Fixed Channel (FC)

$$\mathbf{H} = \begin{bmatrix} 1 & 0.55 & (0.55)^2 & (0.55)^3 \\ 0.55 & 1 & 0.55 & (0.55)^2 \\ (0.55)^2 & 0.55 & 1 & 0.55 \\ (0.55)^3 & (0.55)^2 & 0.55 & 1 \end{bmatrix}$$

- Varied Channel (VC): each element in $\mathbf{H}$, i.i.d, $\mathcal{N}(0, 1)$

Model Training

- Python TensorFlow
- FullyCon $1,000,000$ iterations, DetNet $100,000$ iterations
- FullyCon batch size $1,000$, DetNet batch size $2,000$
- i7-6700 3 days for both architectures
- Noise is randomly generated given SNR value

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Fixed Channel

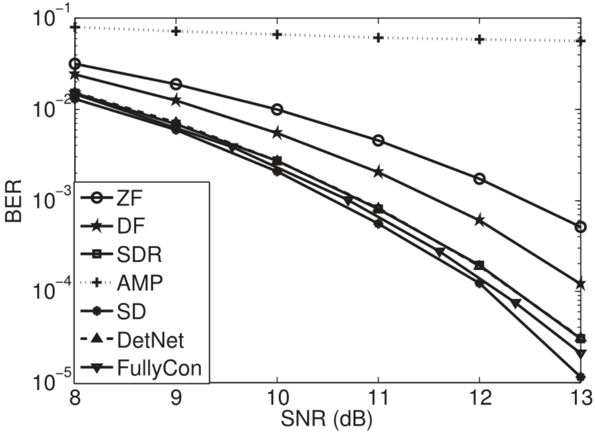

Fig. 3. Comparison of the detection algorithms BER performance in the fixed channel channel case over a BPSK modulated signal.

Varied Channel

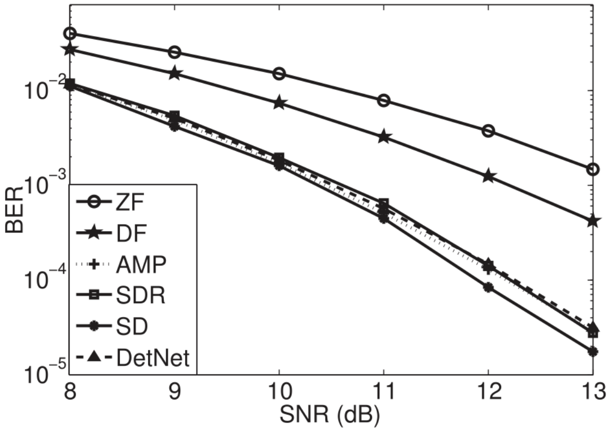

Fig. 4. Comparison of the detection algorithms BER performance in the varying channel case over a BPSK modulated signal. All algorithms were tested channels of size 60×30.

Varied Channel

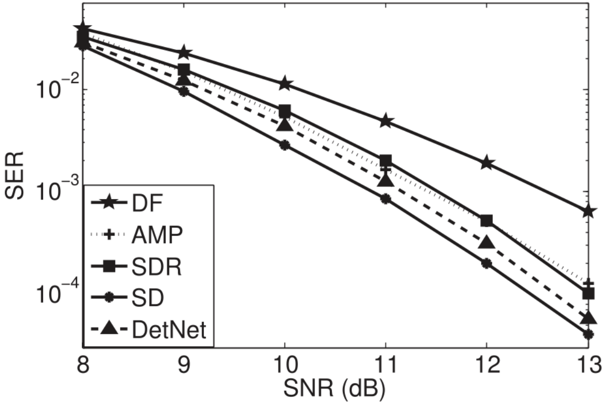

Fig. 5. Comparison of the detection algorithms BER performance in the varying channel case over a QPSK modulated signal. All algorithms were tested on channels of size 30 × 20.

Varied Channel

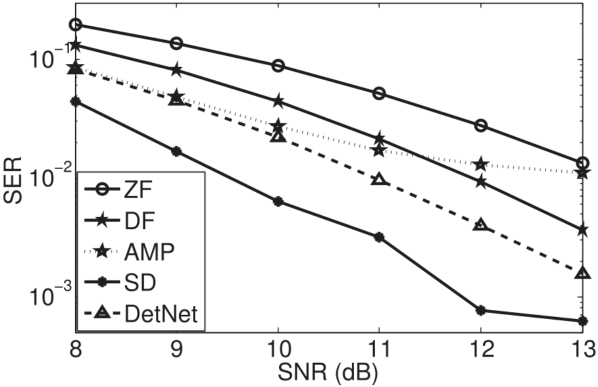

Fig. 6. Comparison of the detection algorithms SER performance in the varying channel case over a 16-QAM modulated signal. All algorithms were tested on channels of size 25 × 15.

Varied Channel

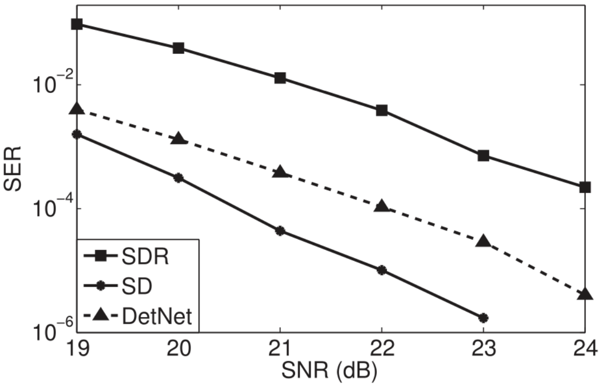

Fig. 7. Comparison of the detection algorithms SER performance in the varying channel case over a 8-PSK modulated signal. All algorithms were tested on channels of size 25 × 15.

Correlated Channel

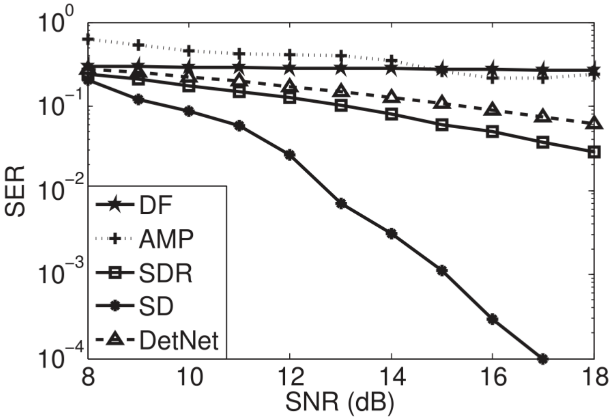

Fig. 8. Comparison of the detection algorithms SER performance in the varying channel where each column is correlated according to a one-ring model correlation matrix created with a random parameter of the angular spread over a QPSK modulated signal. All algorithms were tested on channels of size 15 × 10.

Soft Output

Jensen-Shannon divergence

Measures the dissimilarity between two distributions

$$\mathrm{JSD}(\mathrm{P},\mathrm{Q}) = \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(\mathrm{P},\mathrm{M}) + \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(\mathrm{Q},\mathrm{M})$$

$$\mathrm{M} = \frac{1}{2}(\mathrm{P}+\mathrm{Q})$$

$$\mathrm{D}_{\mathrm{KL}}(\mathrm{P},\mathrm{Q}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{P}(\mathrm{i}) \log\left(\frac{\mathrm{P}(\mathrm{i})}{\mathrm{Q}(\mathrm{i})}\right)$$

Soft Output

Jensen-Shannon divergence

Measures the dissimilarity between two distributions

$$
\begin{aligned}
\mathrm{JSD}(\mathrm{P},\mathrm{Q}) &= \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(\mathrm{P},\mathrm{M}) + \frac{1}{2}\mathrm{D}_{\mathrm{KL}}(\mathrm{Q},\mathrm{M}) \\
\mathrm{M} &= \frac{1}{2}(\mathrm{P}+\mathrm{Q}) \\
\mathrm{D}_{\mathrm{KL}}(\mathrm{P},\mathrm{Q}) &= \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{P}(\mathrm{i}) \log\left(\frac{\mathrm{P}(\mathrm{i})}{\mathrm{Q}(\mathrm{i})}\right)
\end{aligned}
$$

Evaluation: compare between the estimated posterior and the exact posterior in small models

Soft Output

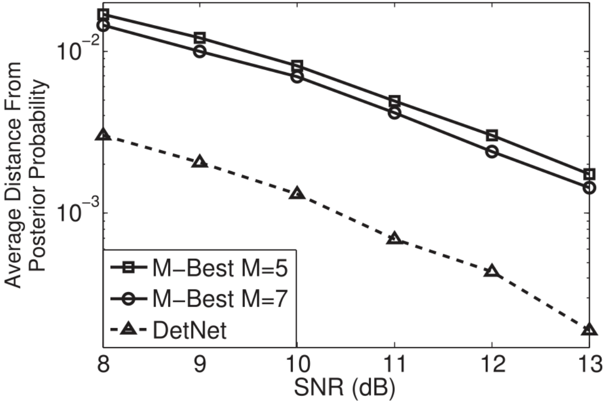

Fig. 9. Comparison of the accuracy of the soft output relative to the posterior probability in the case of a BPSK signal over a 20 × 10 real valued channel.

Soft Output

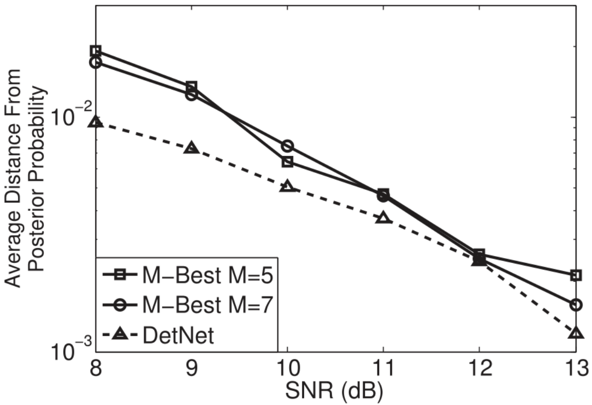

Fig. 10. Comparison of the accuracy of the soft output relative to the posterior probability for a 16-QAM signal over an 8×4 complex valued channel.

Outline

1. Introduction
   - MIMO Detection
   - Machine Learning
2. Proposed Deep MIMO Detectors
   - Fully Connected Network
   - DetNet
   - Soft Decision Output
3. Numerical Simulation
   - Set Up
   - Decoding Accuracy
   - Computational Resources
4. Conclusion

Run Time

TABLE I
FIXED CHANNEL RUNTIME COMPARISON

Channel size	Batch size	FullyCon	DetNet	SDR	AMP	SD
0.55-Toeplitz 60x30	1	4E-4	5E-3	9E-3	5E-3	0.001 -0.01
0.55-Toeplitz 60x30	10	6.6E-05	7E-4	9E-3	E-3	0.001 -0.01
0.55-Toeplitz 60x30	100	2.4E-05	1.6E-04	9E-3	3E-4	0.001 -0.01
0.55-Toeplitz 60x30	1000	1.6E-05	1.1E-04	9E-3	3E-4	0.001 -0.01

Run Time

TABLE II
RUN TIME COMPARISON IN VC. DETNET IS COMPARED WITH THE SDR,AMP AND SPHERE DECODING ALGORITHMS

Constellation channel size	Batch size	DetNet	SDR	AMP	SD
BPSK 60X30	1	0.0066	0.024	0.0093	0.008 -0.1
BPSK 60X30	10	0.0011	0.024	0.0016	0.008 -0.1
BPSK 60X30	100	0.0005	0.024	0.00086	0.008 -0.1
16-QAM 25X15	1	0.006	-	0.01	0.01 -0.4
16-QAM 25X15	10	0.0014	-	0.002	0.01 -0.4
16-QAM 25X15	100	0.0003	-	0.001	0.01 -0.4
8-PSK 25X15	1	0.019	0.021	-	0.004 -0.06
8-PSK 25X15	10	0.0029	0.021	-	0.004 -0.06
8-PSK 25X15	100	0.0005	0.021	-	0.004 -0.06

Run Time

TABLE III
RUN TIME COMPARISON OF SOFT OUTPUT IN VC. THE DETNET IS COMPARED WITH THE M-BEST SPHERE DECODING ALGORITHM

Constellation channel size	Batch size	DetNet	M-Best (M=5)	M-Best (M=7)
BPSK 20X10	1	0.0075	0.006	0.008
BPSK 20X10	10	0.00092	0.006	0.008
BPSK 20X10	100	0.00029	0.006	0.008
16-QAM 8X4	1	0.006	0.008	0.01
16-QAM 8X4	10	0.0008	0.008	0.01
16-QAM 8X4	100	0.0001	0.008	0.01
8-PSK 8X4	1	0.02	0.05	0.07
8-PSK 8X4	10	0.003	0.05	0.07
8-PSK 8X4	100	0.0012	0.05	0.07

Flops Count

TABLE IV
FLOPS COUNT COMPARISON BETWEEN DETNET, SEMIDEFINITE RELAXATION, AMP, SPHERE DECODING AND M-BEST SPHERE DECODING AS A FUNCTION OF K AND THE PARAMETERS OF THE ALGORITHMS

channel size	Number of Flops
DetNet	$\left(K^2 + (3K + 2Aux)Hid\right) L_{DetNet}$
SDR	$\left(6K^3\right) L_{SDR}$
AMP	$(2K \times N + 2Post \times K)\, L_{AMP}$
SD	$2K \times Nodes$
M-Best SD	$(23 + log(\|Con\| \times M)(K \times M \times \|Con\|)$

Flops Count

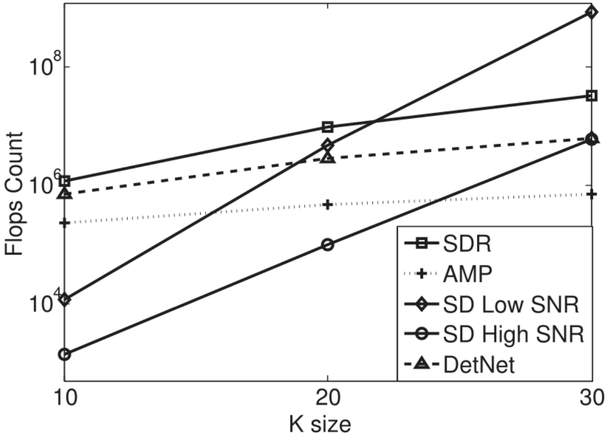

Fig. 12. Flops count for different algorithms over different K sizes for the QPSK constellation and a $K \times K$ channel size.

Flops Count

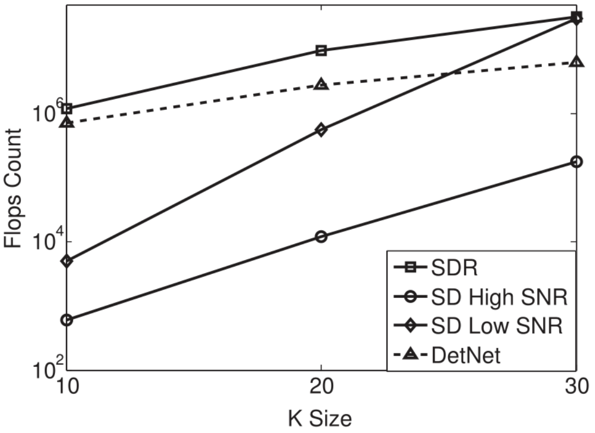

Fig. 13. Flops count for different algorithms over different K sizes for the 8PSK constellation and a $K \times K$ channel size.

Flops Count

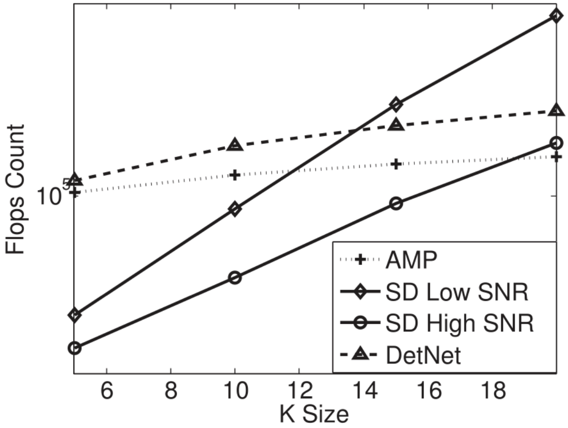

Fig. 14. Flops count for different algorithms over different K sizes for the 16QAM constellation and a $K \times K$ channel size.

Flops Count

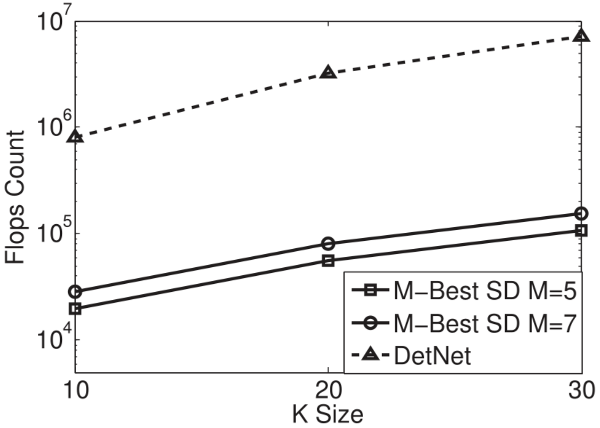

Fig. 15. Flops count for different algorithms over different K sizes for the 16QAM constellation and a $K \times K$ channel size in the soft decision scenario.

Accuracy-Complexity Trade-Off

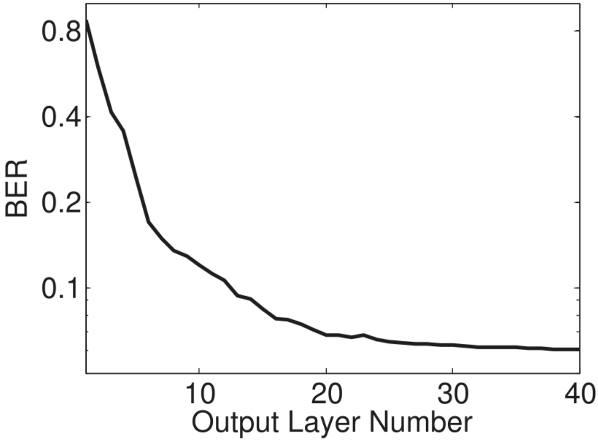

Fig. 16. Comparison of the average BER as a function of the layer chosen to be the output layer in the case of a 60×30 channel and BPSK constellation.

Outline

1 Introduction
- MIMO Detection
- Machine Learning

2 Proposed Deep MIMO Detectors
- Fully Connected Network
- DetNet
- Soft Decision Output

3 Numerical Simulation
- Set Up
- Decoding Accuracy
- Computational Resources

4 Conclusion

Conclusion

- Proposes two NN architectures for MIMO decoding.
- Achieves excellent accuracy with low complexity.
- Is able to produce soft-output.
- DetNet is able to detect multiple channel realizations with a single training

Opinion

- It has a strong assumption of the channel model–AWGN.
- It is not a modern deep NN, but an unfolded iteration where parameters are determined based on data.