Digital Pre-Distortion

Danila Doroshin

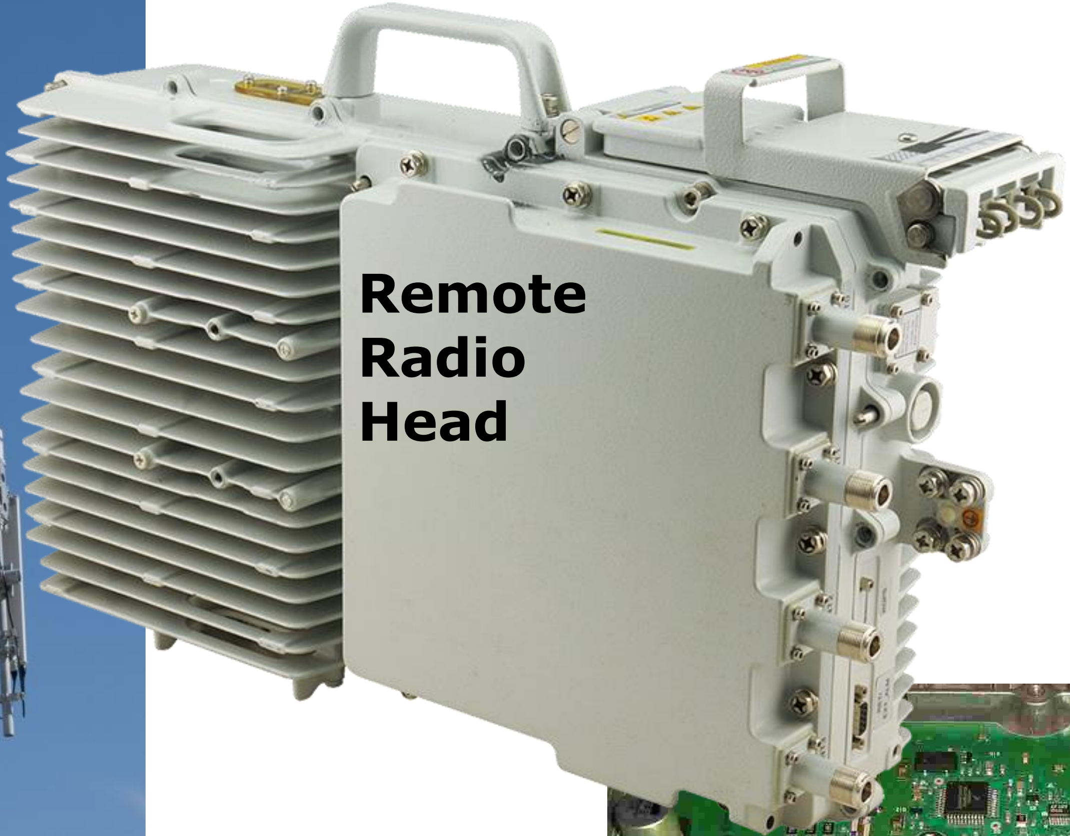
Principal Engineer, PhD

www.huawei.com

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Digital Pre-Distortion for Power Amplifiers



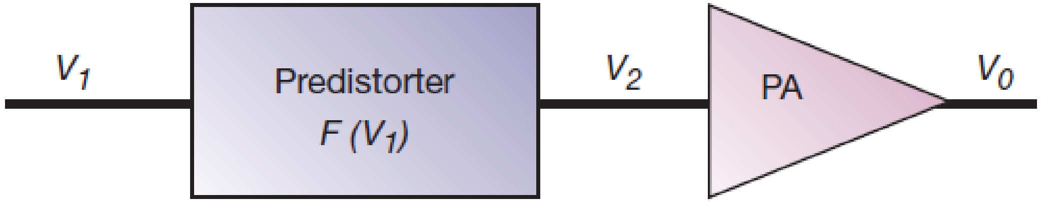
Digital Pre-Distortion (DPD)

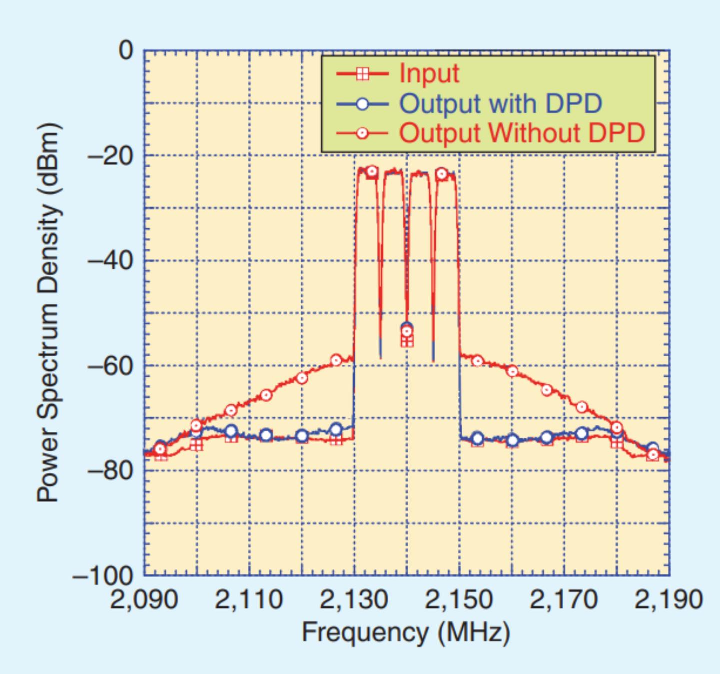
Nonlinear PA behavior disturbs the transmitted signal

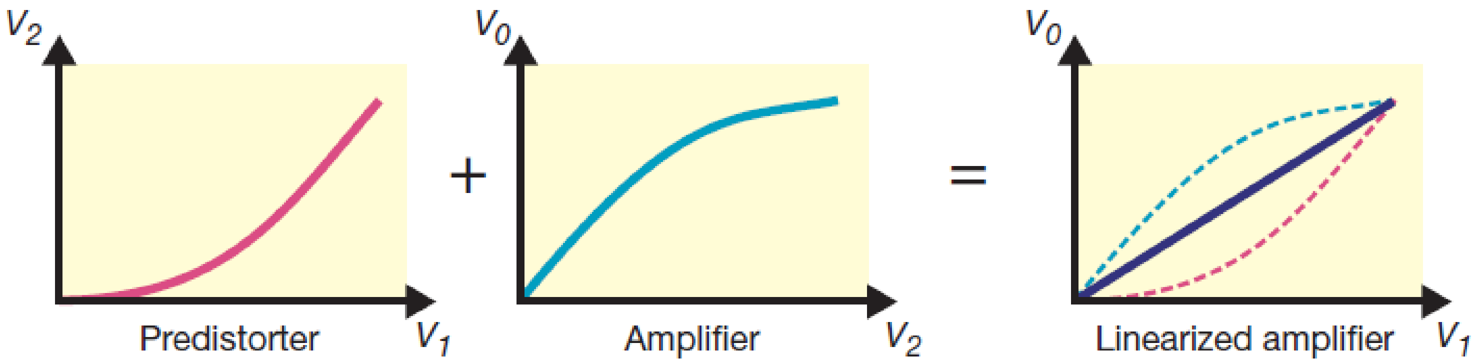
DPD solution is to apply PA^{-1} to the input signal.

The ML problem is to estimate PA^{-1} (*Sec2Sec Regression*)

1. Current algorithms do not perform well
2. Inverse function in the multidimensional space







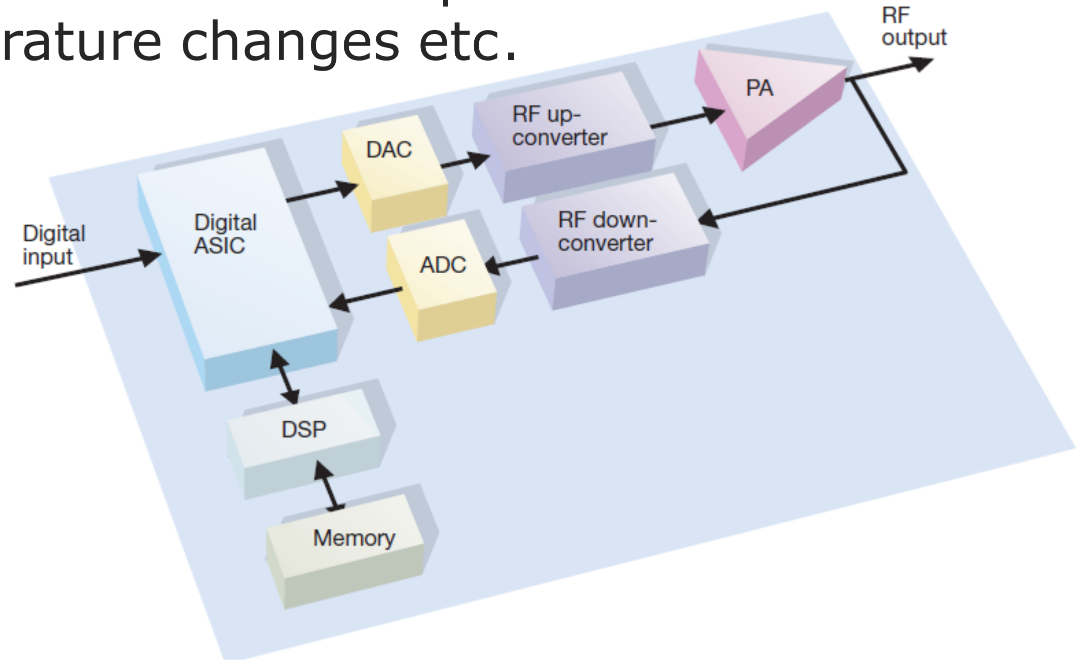
$|V_2| = \arcsin(|V_1|)$ $|V_0| = C^*\sin(|V_2|)$ $|V_2| = C^*|V_1|$

Digital Pre-Distortion (DPD)

Implementation: inside wireless base station DPD is implemented as ASIC.

Discrete time signals: DPD receives a signal in discrete time after Analogue to Digital Converter (ADC).

DPD adaptation: PA output is used as a feedback for DPD model adaptation. It allows to be adaptive to the different signals, temperature changes etc.



ASIC requirements for DPD model

1. Number of parameters < 1000
2. **Fixed point** instead floating point for inference and for **Back Propagation**

Digital Pre-Distortion (DPD)

Volterra Model

$$x_{\text{out}}(n) = \sum_{k=1}^{K} \sum_{i_1=0}^{M} \cdots \sum_{i_p=0}^{M} h_p(i_1, \cdots, i_p) \prod_{j=1}^{k} x_{\text{in}}(n - i_j)$$

Memory Polynomial Model

$$x_{\text{out}}(n) = \sum_{j=0}^{M} \sum_{i=1}^{N} a_{\text{ji}} \cdot x_{\text{in}}(n-j) \cdot |x_{\text{in}}(n-j)|^{i-1}$$

Wiener Model

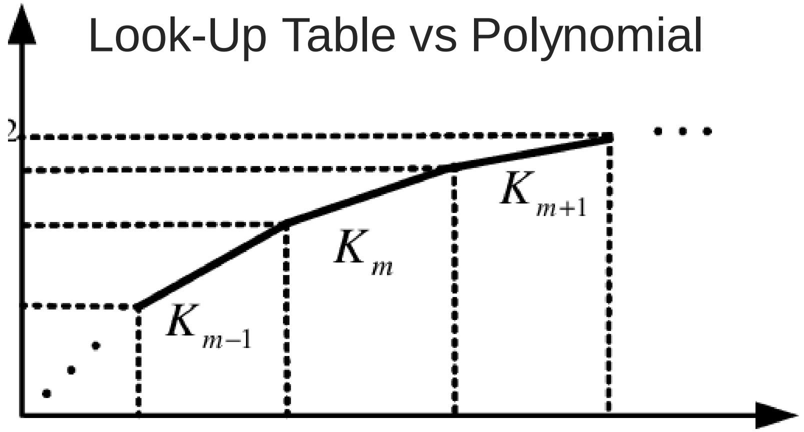
$$x_1(n) = \sum_{j=0}^{M} h(j) \cdot x_{\text{in}}(n-j)$$

$$x_{\text{out}}(n) = G(|x_1(n)|) \cdot x_1(n),$$

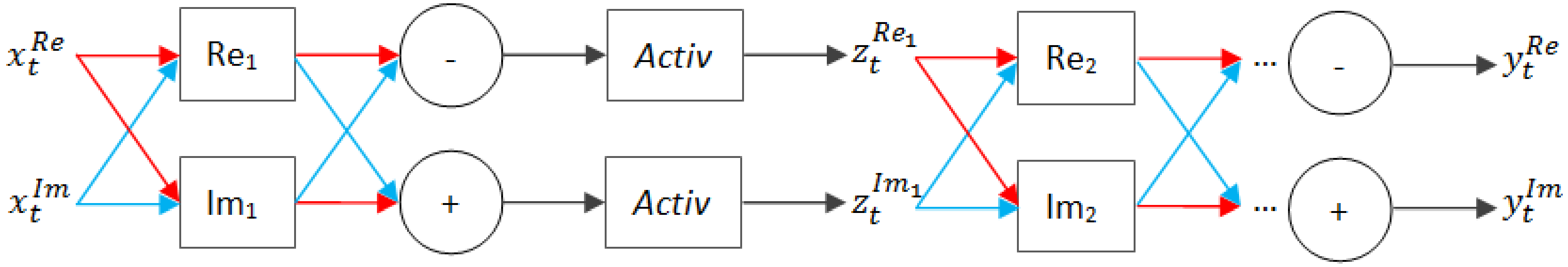
Hammerstein Model

$$x_1(n) = G(|x_{\text{in}}(n)|) \cdot x_{\text{in}}(n)$$

$$x_{\text{out}}(n) = \sum_{j=0}^{M} h(j) \cdot x_1(n-j)$$



DPD and Harmonic network



The idea of using such transform as a layer is used in [Harmonic networks] approach

$$Wx = (W^{Re} + iW^{Im})(x^{Re} + ix^{Im}) = \underbrace{W^{Re}x^{Re} - W^{Im}x^{Im}}_{real\ part} + i\ \underbrace{W^{Re}x^{Im} + W^{Im}x^{Re}}_{imaginary\ part}$$

https://arxiv.org/abs/1612.04642

Sparsification of DL models

Some recent methods

1. Learning Sparse Neural Networks through L_0 Regularization (Christos Louizos et al.) ICLR 2018

$$\mathcal{R}(\boldsymbol{\theta}) = \frac{1}{N}\left(\sum_{i=1}^{N}\mathcal{L}\big(h(\mathbf{x}_i;\boldsymbol{\theta}),\mathbf{y}_i\big)\right) + \lambda\|\boldsymbol{\theta}\|_0, \qquad \|\boldsymbol{\theta}\|_0 = \sum_{j=1}^{|\theta|}\mathbb{I}[\theta_j \neq 0],$$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\{\mathcal{R}(\boldsymbol{\theta})\},$$

2. Variational Dropout Sparsifies Deep Neural Networks (Dmitry Molchanov et al.) ICML 2017

Sparsification of DL models

What else?

Brain Surgeon

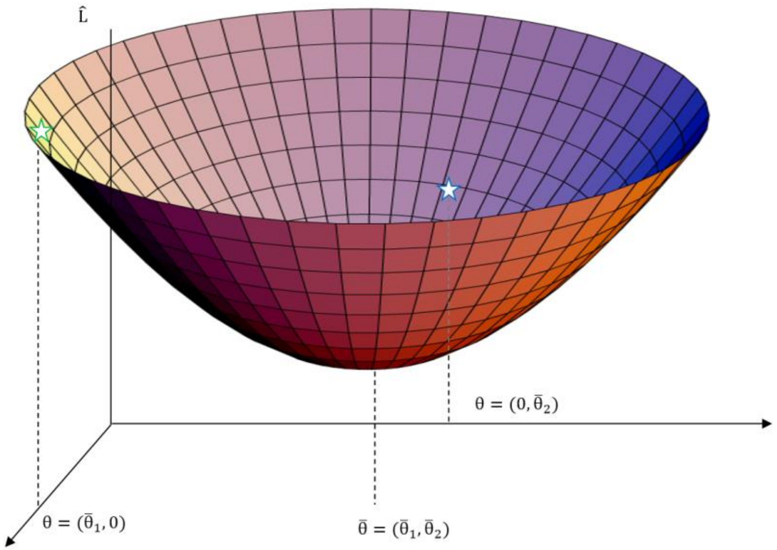
Sparsification of DL models
What else?

Optimal Brain Surgeon: Extensions and performance comparisons
(Babak Hassibi et al.) *Advances in neural information processing systems*. **1994**

$$\delta E = \underbrace{\left(\frac{\partial E}{\partial \mathbf{w}}\right)^T}_{\approx 0} \cdot \delta \mathbf{w} + \frac{1}{2}\, \delta \mathbf{w}^T \cdot \underbrace{\frac{\partial^2 E}{\partial \mathbf{w}^2}}_{\equiv \mathbf{H}} \cdot \delta \mathbf{w} + \underbrace{O(\|\delta \mathbf{w}\|^3)}_{\approx 0},$$

$$\mathbf{o} = F(\mathbf{w}, \mathbf{in})$$

$$\mathbf{H} = \frac{1}{P} \sum_{k=1}^{P} \frac{\partial d(\mathbf{t}^{[k]}, \mathbf{o}^{[k]})}{\partial \mathbf{o}} \cdot \frac{\partial^2 F(\mathbf{w}, \mathbf{in}^{[k]})}{\partial \mathbf{w}^2}$$



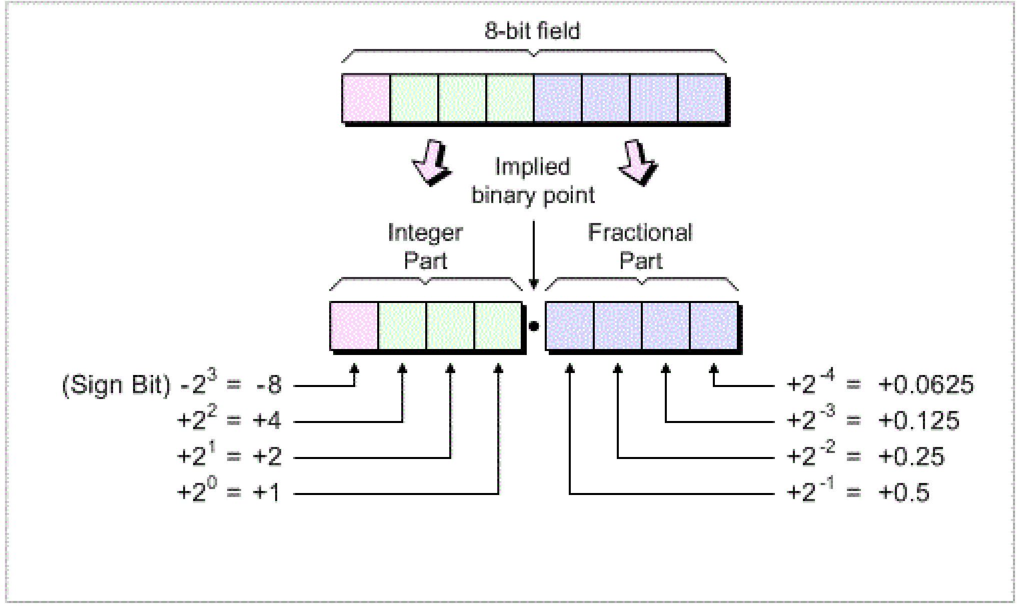
H can be computed as a covariance matrix of gradients
(with the help of Fisher Information).

Fixed-point (FP) arithmetic and Binary parameters

1. Different bit width for forward and backward
2. Special architecture of FP model

$$g(x) = g(x_0) + \nabla g(x_0)^T(x - x_0) + \frac{1}{2}(x - x_0)^T \nabla^2 g(x_0)(x - x_0) + \ldots$$

3. Binary Parameters training (MUX and Delay)



$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

- Each PA has its specific model
- Binary parameters are specific for different PA

- Offline training for Binary parameters
- Online training for Binary parameters

Directions

1. Fixed-point model architecture
2. Binary parameters training. Low-cost architecture reconfiguration on the run
3. New architectures (nonlinearity modeling, regime-switching, models in frequency domain…)
4. Computational complexity reduction, efficient computations

Thank You

www.huawei.com