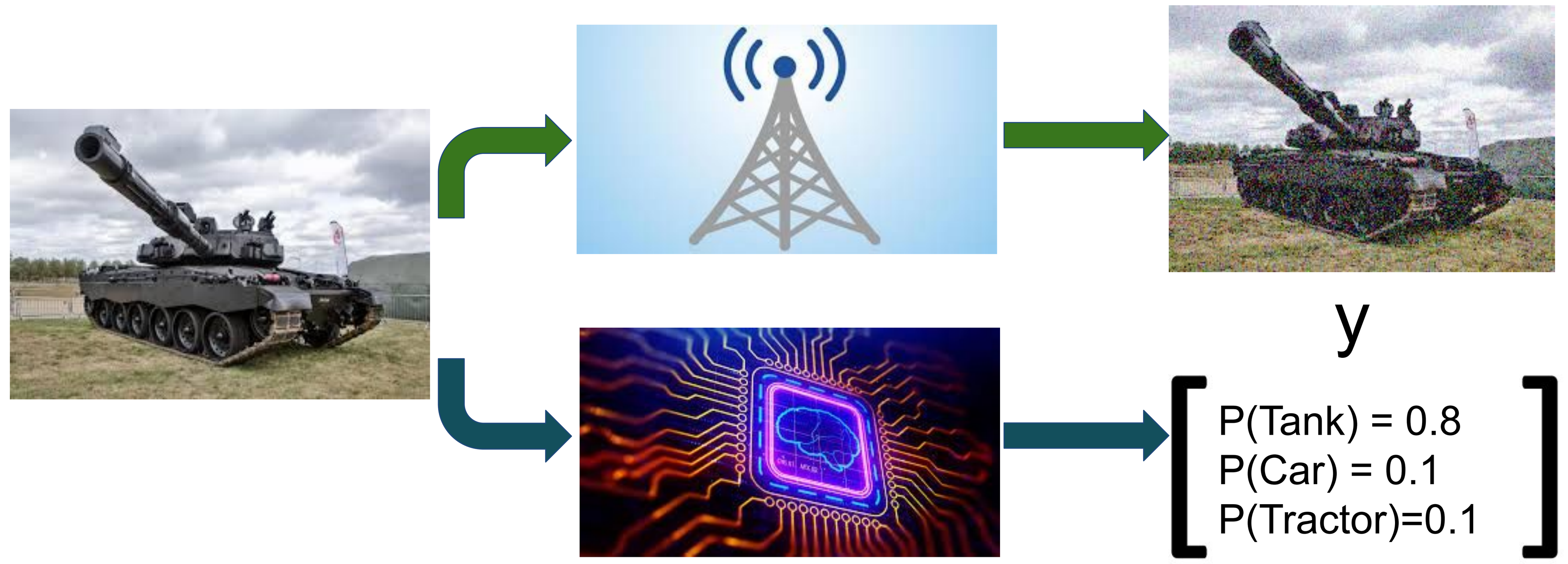
Error Correcting Output Codes Improve Probability Estimation and Adversarial Robustness of Deep Neural Networks

Gunjan Verma, Ananthram Swami (CCDC Army Research Laboratory, USA)

Objective: To develop robust machine learning models that

- yield better probability estimates across a range of data (benign, noisy/degraded, adversarial)
- are more resistant to adversarial perturbations
- are light-weight/efficient during training and testing



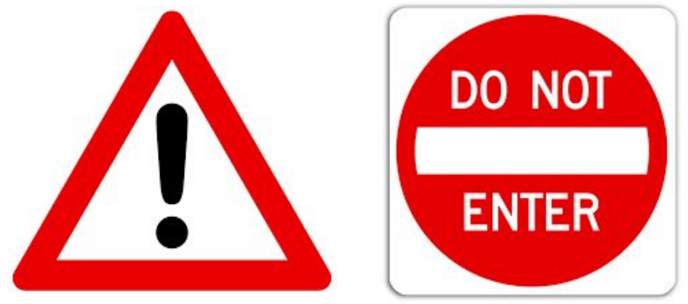
Communications pipeline.
Input: x, passes through a channel
Output: y, noisy version of x
Goal: reconstruct x from y

Machine Learning pipeline.
Input: x, passes through a model
Output: y, noisy inference on x
Goal: infer properties of x (e.g., its classification) from y

Principles of error correction:

1. **Selectivity**: Only some sequences (codewords) are allowable
2. **Redundancy**: Valid codewords contain more bits than necessary
3. **Separation**: Valid codewords have large Hamming distance
4. **Independence**: Errors within a codeword are (ideally) minimally correlated

Principles of error correction:
None!



Q: How can we endow ML systems with mechanisms for detection/correction of classification errors?
A: Change the way the output is encoded and mapped to probabilities

Enforcing Principle 1: Denote logits by **z**. For *any* **z** for which $z_k >> z_i$ for i ≠ k, softmax will assign class k large probability even if **z** is outside the training manifold. Thus softmax + standard (one-hot) code has poor selectivity - it fails to discriminate between typical and atypical **z**. The fix: the *sigmoid decoder*

$$\sigma_k(\mathbf{z}) = \frac{1}{1+\exp(-z_k)}$$

Step 1: Map logits to (soft) bits: apply a sigmoidal activation function per logit.

$$p_\sigma(k) = \frac{\max(\sigma(\mathbf{z})\cdot \mathbf{C}_k, 0)+\alpha}{\sum_{i=1}^{M}(\max(\sigma(\mathbf{z})\cdot \mathbf{C}_i, 0)+\alpha)}$$

Step 2: Mapping bits to probabilities; the more bits matching a codeword C_k, the higher class k's probability

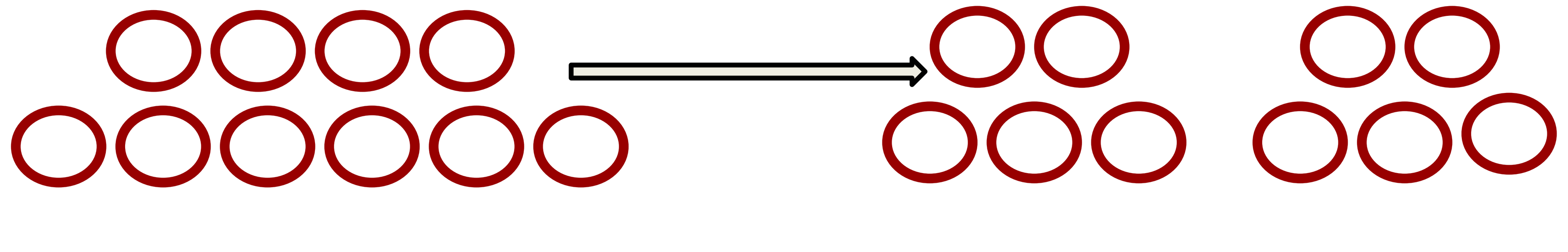
Now, probabiliy of class i is large if and only if the network output matches codeword for i; non-codewords are assigned small probability

Enforcing Principles 2,3: Instead of the standard one-hot encoding, use a code such as Walsh-Hadamard code

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

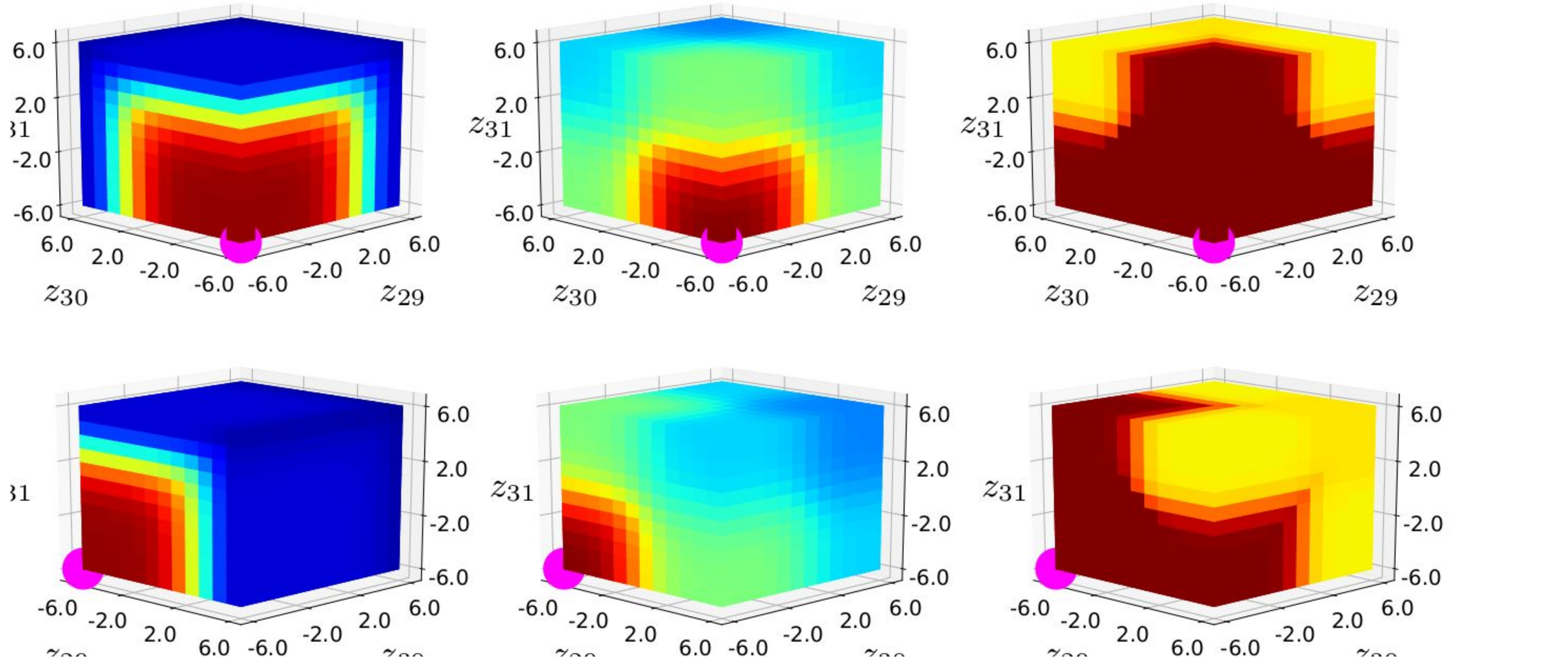
- Each row encodes a separate class
- Flexible design choice (number of bits can > number of classes)
  - enables **redundancy** (extra bits)
  - enables increased codeword **separation**

Enforcing Principle 4: Instead of one monolithic architecture predicting all bits, use multiple smaller networks, each predicting a few bits



Conventional architecture outputs all bits (4 in this case) in a single network ⇒ output bits are highly correlated

Modified ("ensemble") architecture uses multiple (disjoint) models, each outputting a few bits (2 in this case) ⇒ reduced correlation among output bits

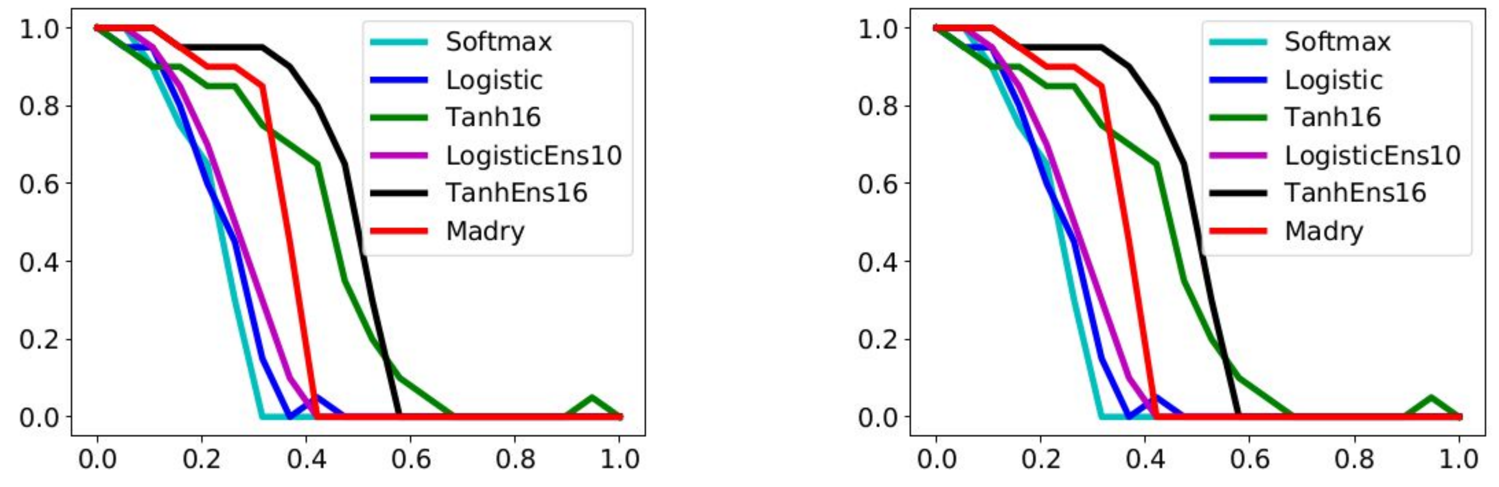


Probability of class 0 for a multi-class classification problem over a 3-D slice of a 32-dim logit space. Red (blue) means high (low) probability of class 0. (**left**) standard softmax scheme, (**middle**) sigmoid decoder with Identity code, (**right**) sigmoid decoder with Hadamard code. Softmax quickly goes from high confidence of class 0 to high confidence of not class 0., Sigmoid-based decoder with is more robust; Hadamard coding further increases robustness

Model	# Params	Benign	PGD	CW	BSA $\alpha = 0.8$	DAA	Rand	+N(0,1)
Softmax	468, 760	.9922	.043	0.45	0.0	-	.395	.35
Logistic	468, 760	.9938	.167	0.75	0.1	-	.612	.419
Tanh16	469, 066	.9931	.643	0.9	0.8	-	.877	.384
LogisticEns10	416, 140	.9932	.355	0.88	0.4	-	.966	.558
TanhEns16	344, 944	.9942	.933	1.0	1.0	.9173	.985	.644
TanhEns32	689, 888	.9946	.924	1.0	1.0	-	.991	.662
Madry	3, 274, 634	.9853	.925	.85	.52	.8879	.351	.119

Model	# Params	Benign	PGD	CW	BSA $\alpha = 0.8$	DAA	Rand	+N(0,1)
Softmax	468, 760	.9922	.043	0.45	0.0	-	.395	.35
Logistic	468, 760	.9938	.167	0.75	0.1	-	.612	.419
Tanh16	469, 066	.9931	.643	0.9	0.8	-	.877	.384
LogisticEns10	416, 140	.9932	.355	0.88	0.4	-	.966	.558
TanhEns16	344, 944	.9942	.933	1.0	1.0	.9173	.985	.644
TanhEns32	689, 888	.9946	.924	1.0	1.0	-	.991	.662
Madry	3, 274, 634	.9853	.925	.85	.52	.8879	.351	.119

Experimental results. Top: MNIST. Bottom: CIFAR10. In all cases, higher numbers are better.



Model accuracy vs. PGD perturbation strength. Left: MNIST. Right: CIFAR10. Our model dominates all others.