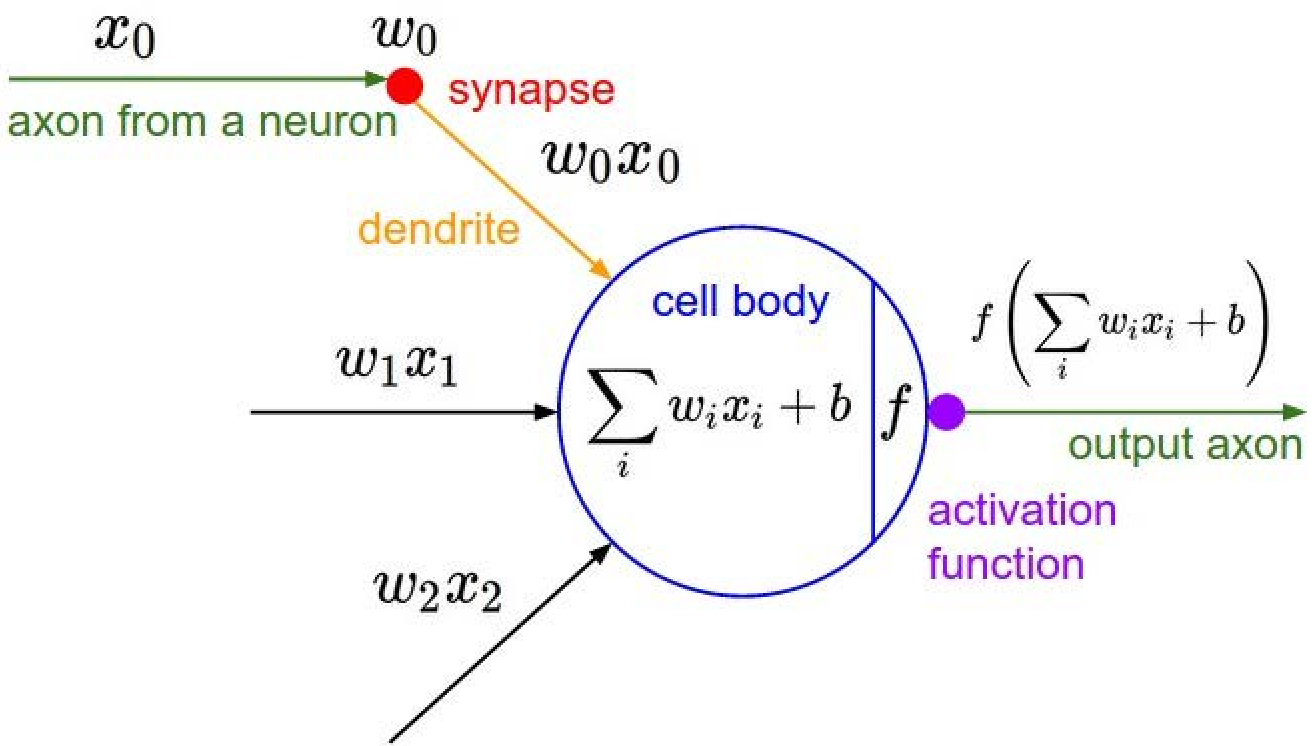
Why and Why Not Convolutional Neural Networks (CNNs)?

December 2017

C.-C. Jay Kuo
University of Southern California

Part I: Why CNNs?

Computational Neuron (Convolution + Nonlinear Activation)



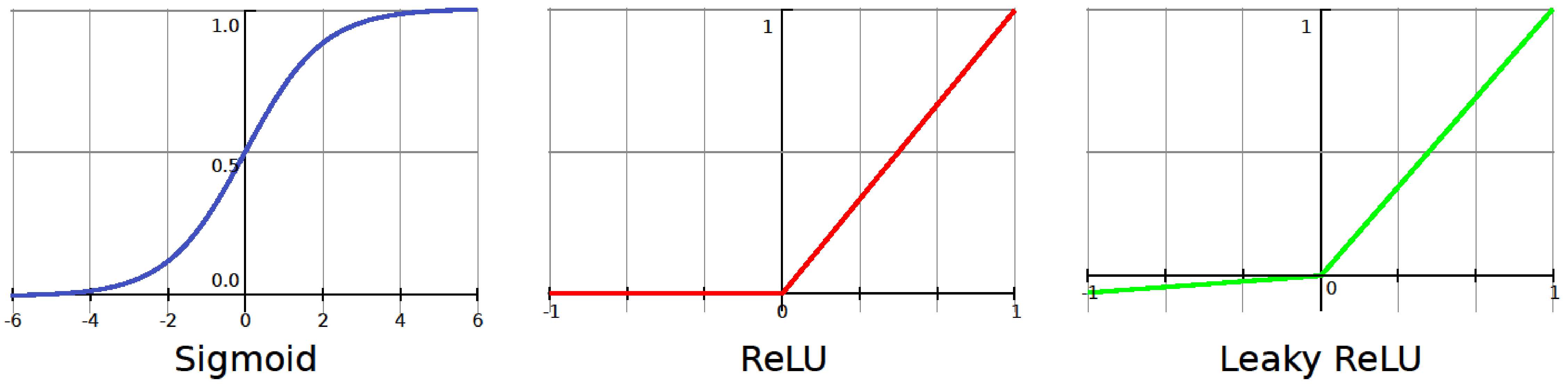
Nonlinear activation functions: sigmoid, ReLU, Leaky ReLU

Figure Credit: Stanford CS231n Course Instruction

Understanding Filter Weights

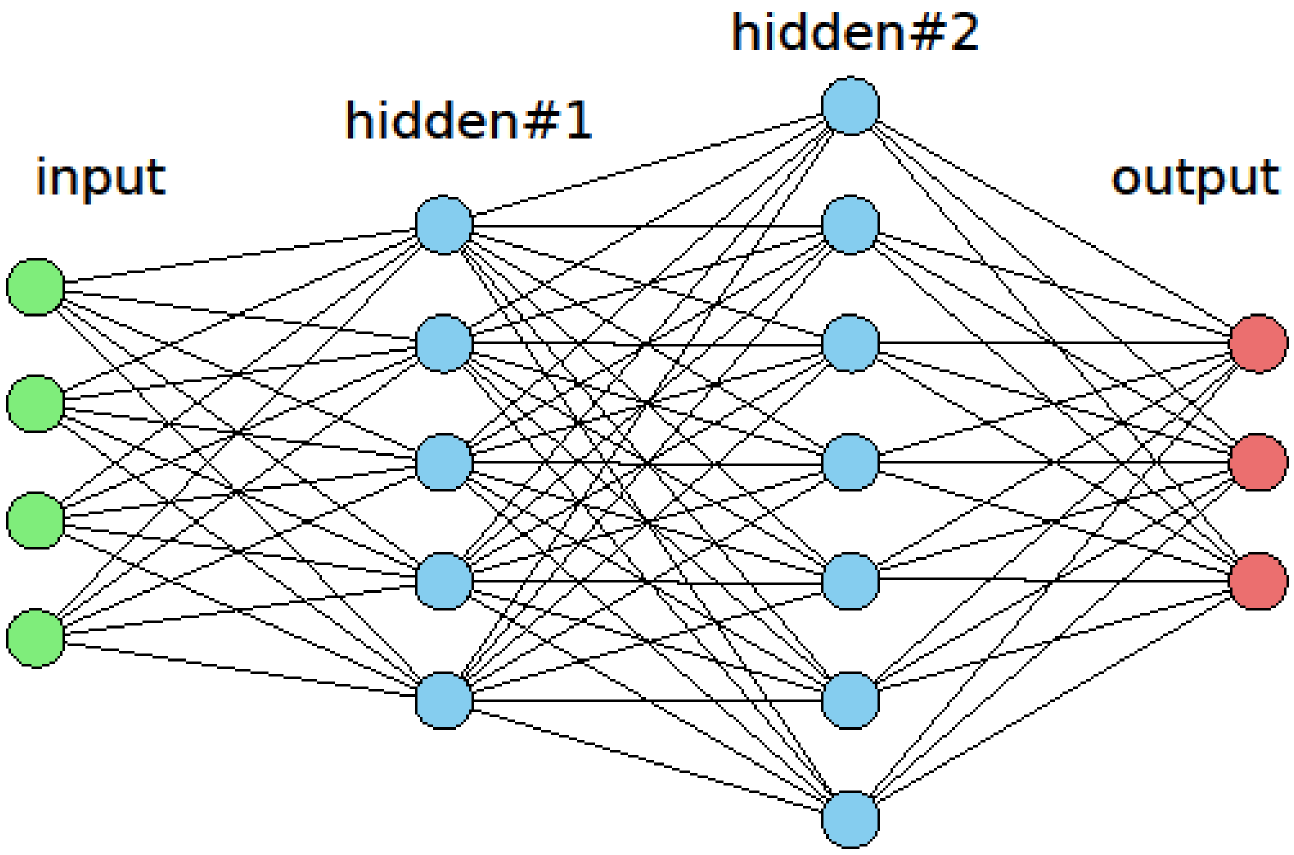
- 1st viewpoint
  - Parameters to optimize in large nonlinear networks
  - Backpropagation – SGD
- 2nd viewpoint
  - Matched filters
  - k-means clustering
- 3rd viewpoint
  - Bases (or kernels) for a linear space
  - Subspace approximation

Understanding Nonlinear Activation



Multilayer Perceptron (MLP)

- Supervised learning by backpropagation (BP)

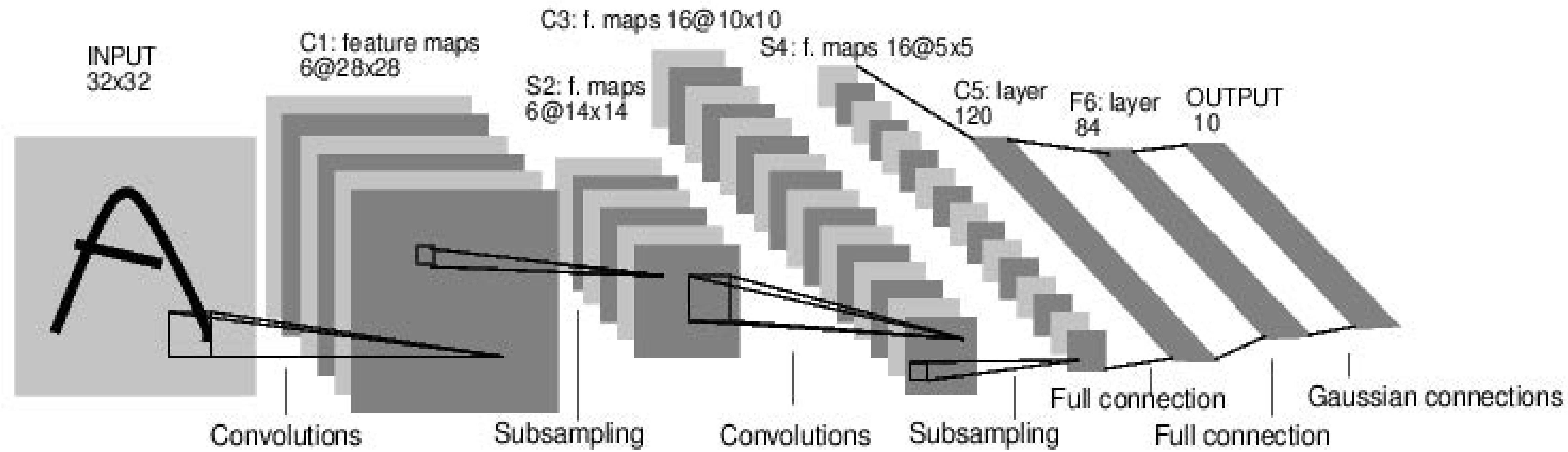


Classic 2-Hidden Layer MLP

Competitions and Limitations

- MLPs were hot in 80's and early 90's
  - Input: n-D feature vector (one feature per node)
- Competitive solutions exist
  - SVM
  - Random Forest
- What happens if the input is the source data?
  (e.g. an image of size 32x32 = 1024)

LeNet-5



Input	→	Convolution/Nonlinear Activation	→	C1
C1	→	Pooling	→	S2
S2	→	Convolution/Nonlinear Activation	→	C3
C3	→	Pooling	→	S3

Single Layer Signal Analysis (1)

- Signal Modeling

$$\mathbf{x} = \mathbf{A}\mathbf{c},$$

$$\mathbf{A} \in R^{N \times M}$$

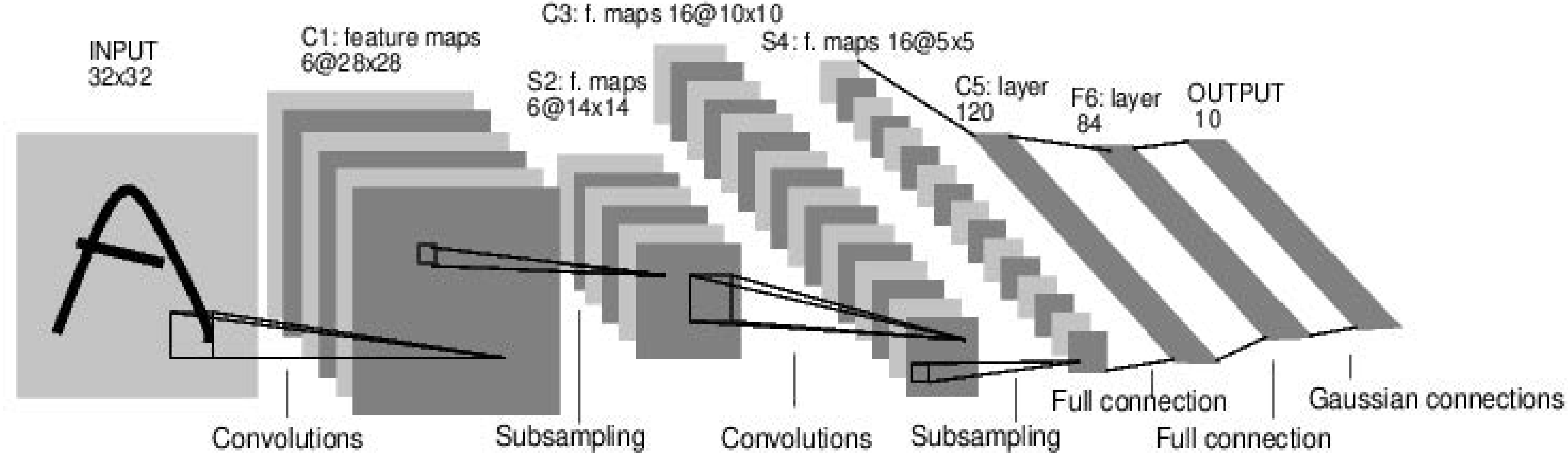
- **x** are a class of observed signals
- **A** and **c** are to be determined

Single Layer Signal Analysis (2)

- Signal Transform (M=N)
  - Fourier transform: sinusoid components in **x**
  - Wavelet transform: multi-scale components in **x**
- Sparse Coding (M>N)
  - Find the most suitable dictionary **A** for **x** under constraints on **c** (e.g. sparsity)
  - Dictionary learning
- Feature extraction
  - Coefficient **c** for an observed instance, **x,** can be used as its features

Where CNN Stores “Learned Knowledge”?

- All training/learning results are summarized in filter weights
  - Filter weights play a critical role in understanding CNN



Each convolutional or fully connected layer defines a transform matrix

CNN as Multi-Layer Signal Transform

- Comparison of single- and multi-layer methods

Single-layer Approach

- There is only one transform matrix
- Learning **A** from a class of signals
- Determine **c** from an instance of **x**
- Use **c** as the features for decision

Multi-layer Approach

- There are multiple transform matrices
- Learning **A's** from a class of signals and their decision labels (**d**)
- Feed an instance of **x** into the network for its decision **d**
- Need a nonlinear activation between layers

Convolution as A Matched Filter

- A convolution operation can be viewed as the inner product to two vectors

    -> Interpreted as “correlation”

- Filter Weights are fixed in the test stage
  - Called anchor vectors

Multiple Parallel Correlators

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \quad \mathbf{A}^T = [\mathbf{a}_1 \cdots \mathbf{a}_k \cdots \mathbf{a}_K]$$

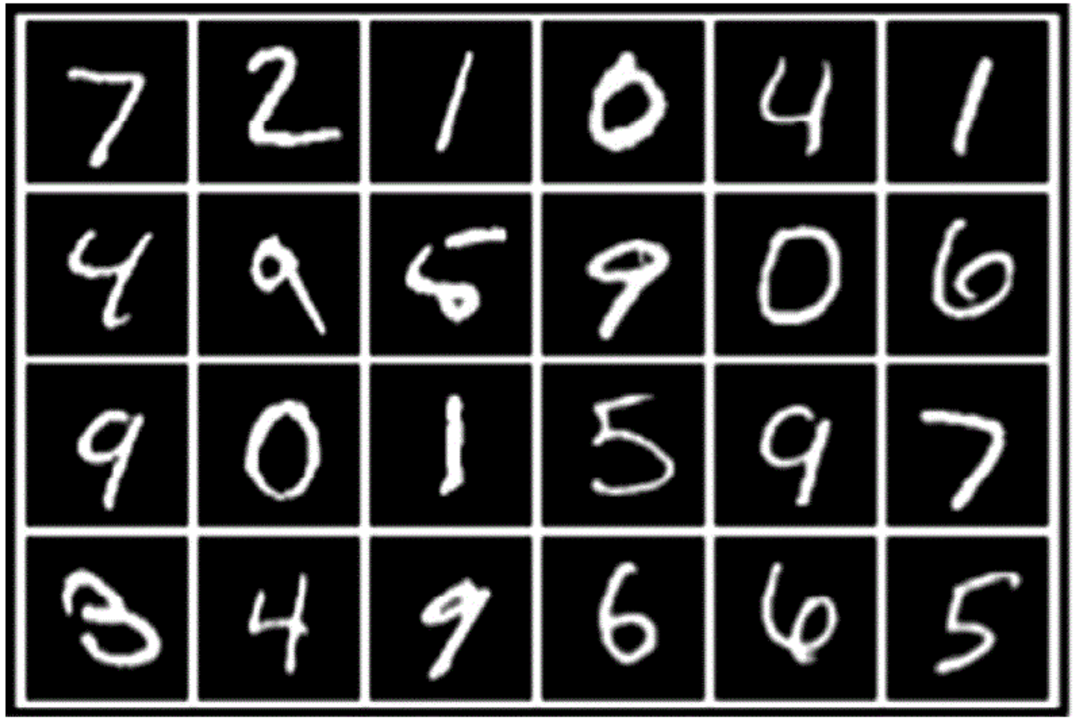
$$y_k = \mathbf{a}_k^T \mathbf{x} \text{ and } \mathbf{A} \in R^{K \times N}$$

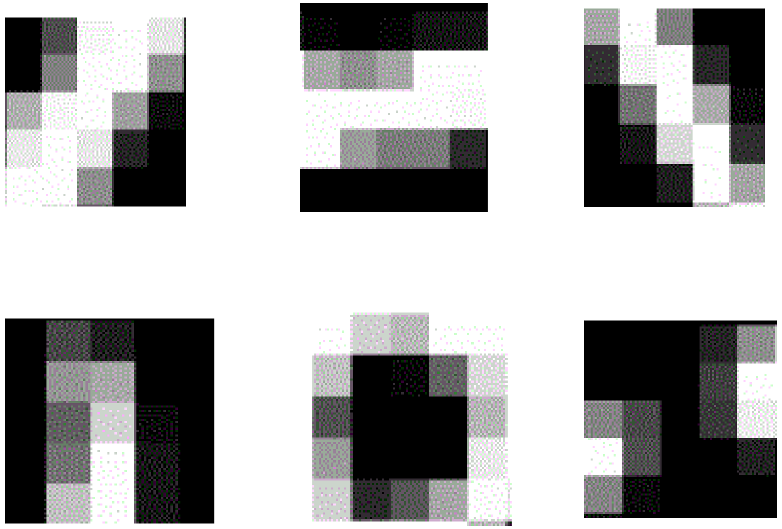
$$\mathbf{y} = (y_1, \cdots, y_k, \cdots y_K)^T \in R^K$$

We view $\mathbf{a}_k$ as a visual pattern

MNIST Dataset

6 Representative Patterns

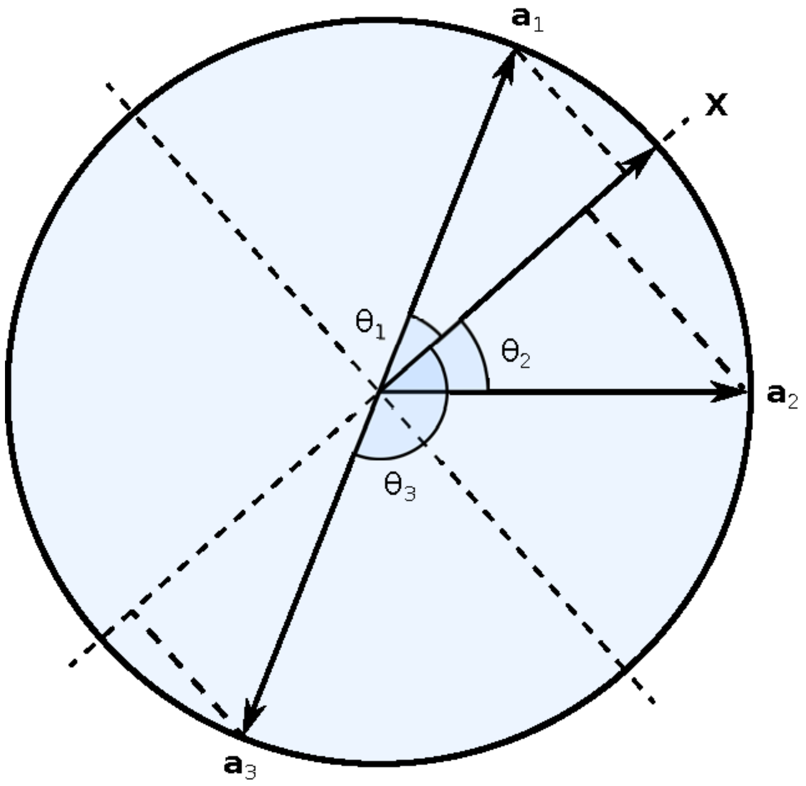




Pattern Matching by Correlation $y_k = \mathbf{a}_k^T \mathbf{x}$

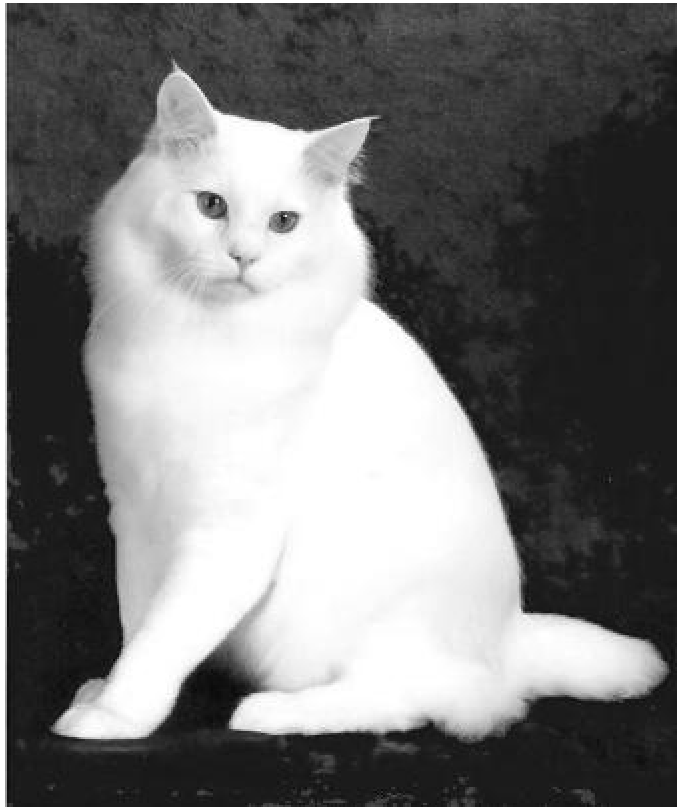
Why Nonlinear Activation?

- REctified COrrelation on a Sphere (RECOS) Model
- Consider clustering in the unit sphere
- The distance is measured by the geodesic distance
- A shorter geodesic distance implies a small intersection angle between two vectors
- What happens to negative correlation (or projection)?



C.-C. Jay Kuo, “Understanding Convolutional Neural Networks with A Mathematical Model”, the Journal of Visual Communication and Image Representation, Vol. 41, pp. 406-413, 2016

Comparison of Positive & Negative Correlations



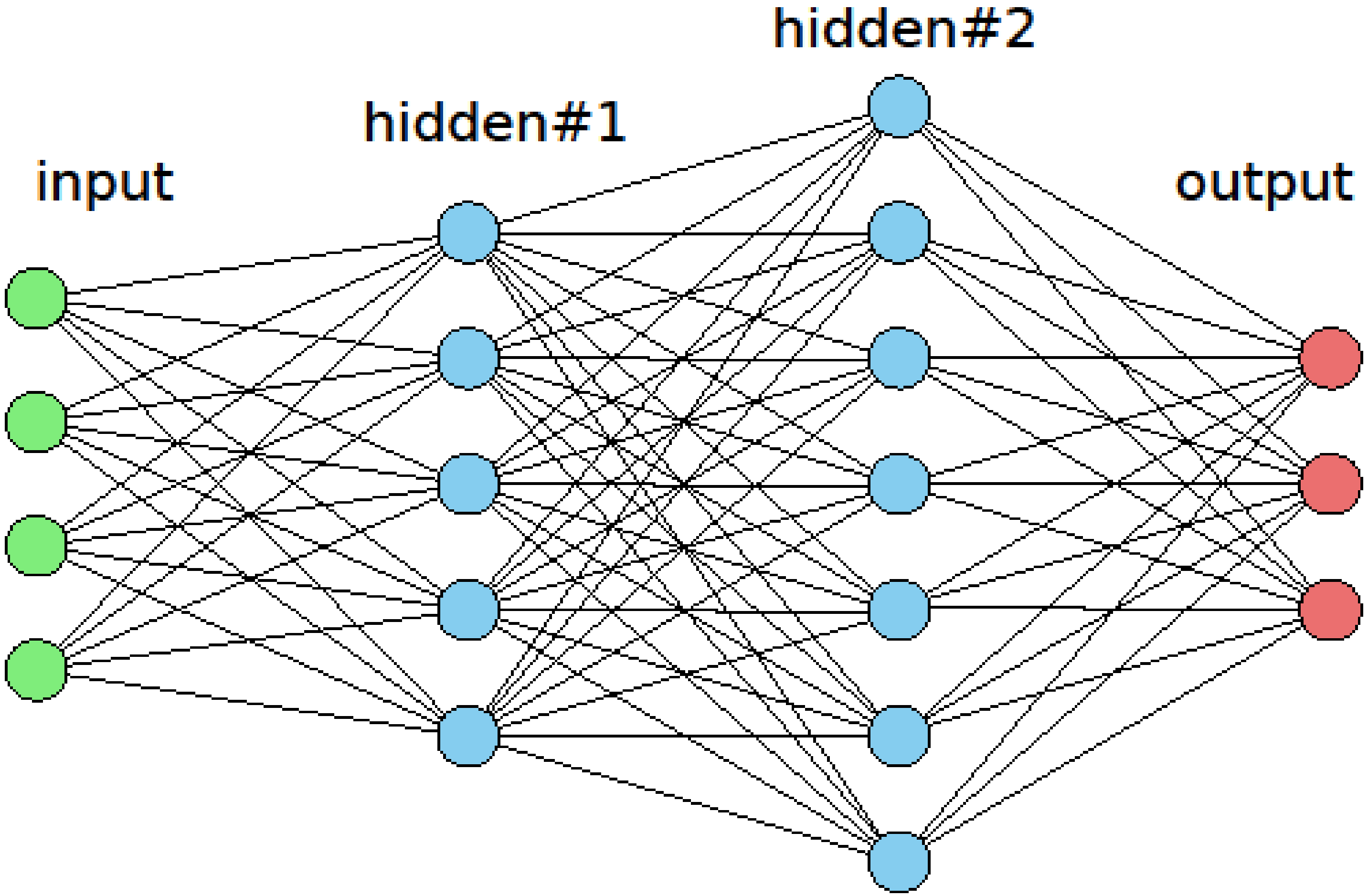
Sign Confusion Problem

- When two convolutional filters are in cascade, the cascaded system cannot differentiate the following scenarios:

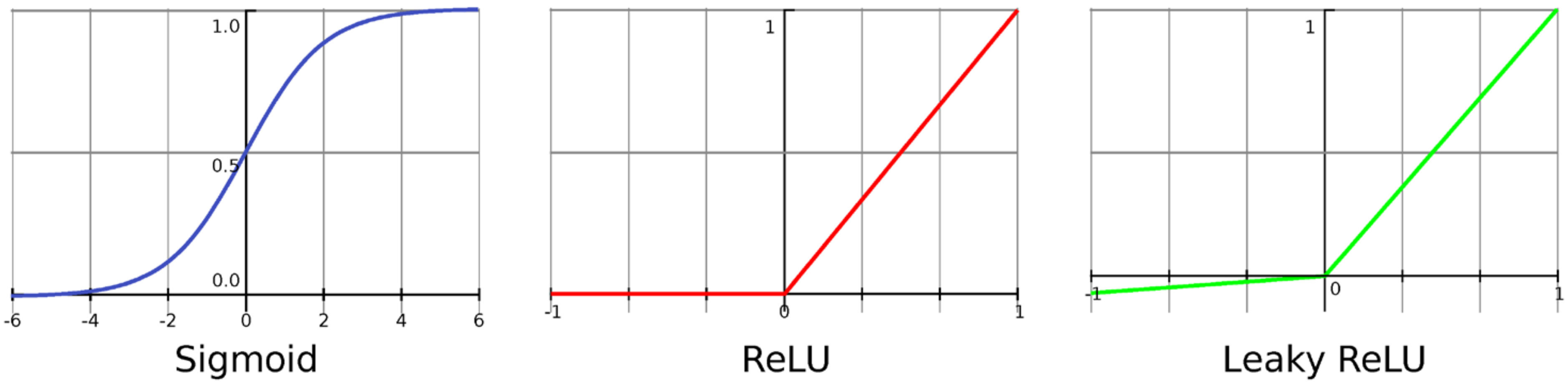
- Confusing Case #1
  - A positive correlation in stage 1 and a positive filter coefficient in stage 2
  - A negative correlation in stage 1 and a negative filter coefficient in stage 2

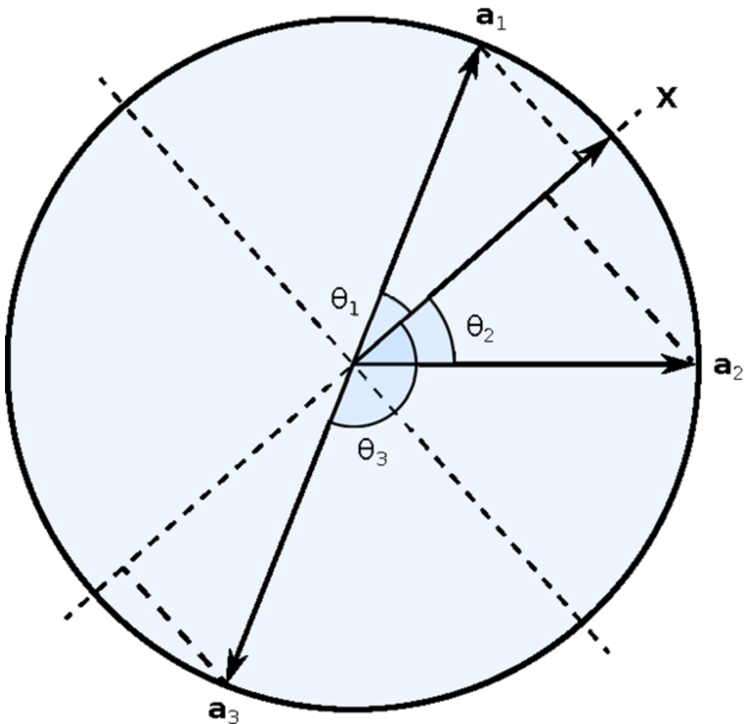
- Confusing Case #2
  - A positive correlation in stage 1 and a negative filter coefficient in stage 2
  - A negative correlation in stage 1 and a positive filter coefficient in stage 2

An Illustration

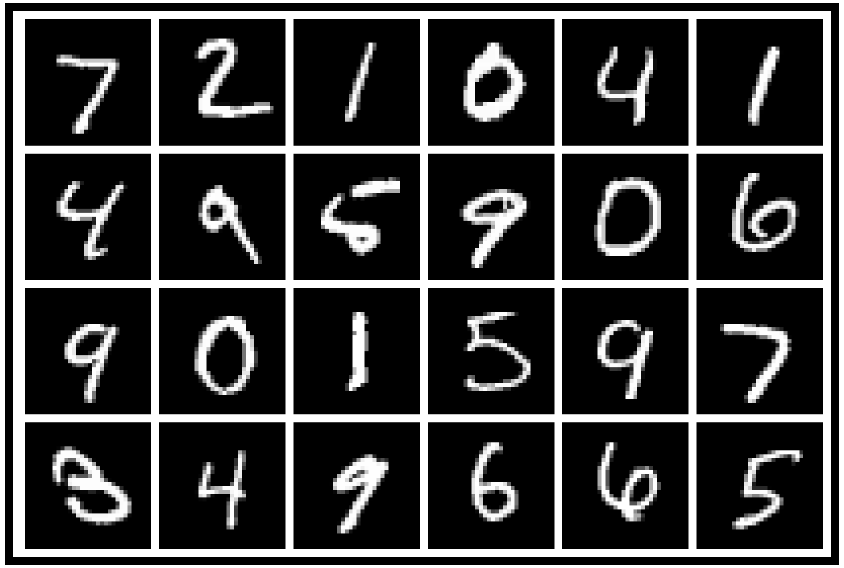


Nonlinear Activation Revisited

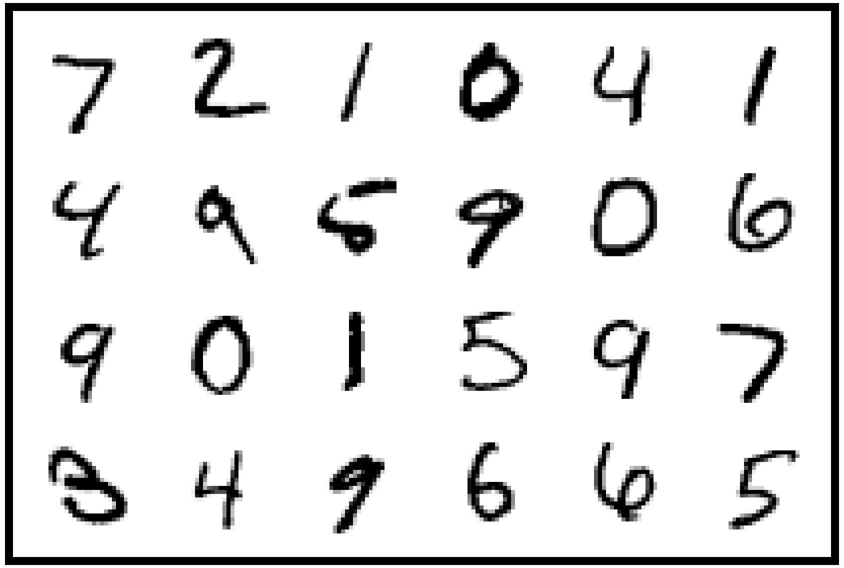




Experiments on MNIST



Original



Negative

Test Performance of LeNet-5

- Original: 98.94% (trained by original)
- Negative: 37.36% (trained by original)

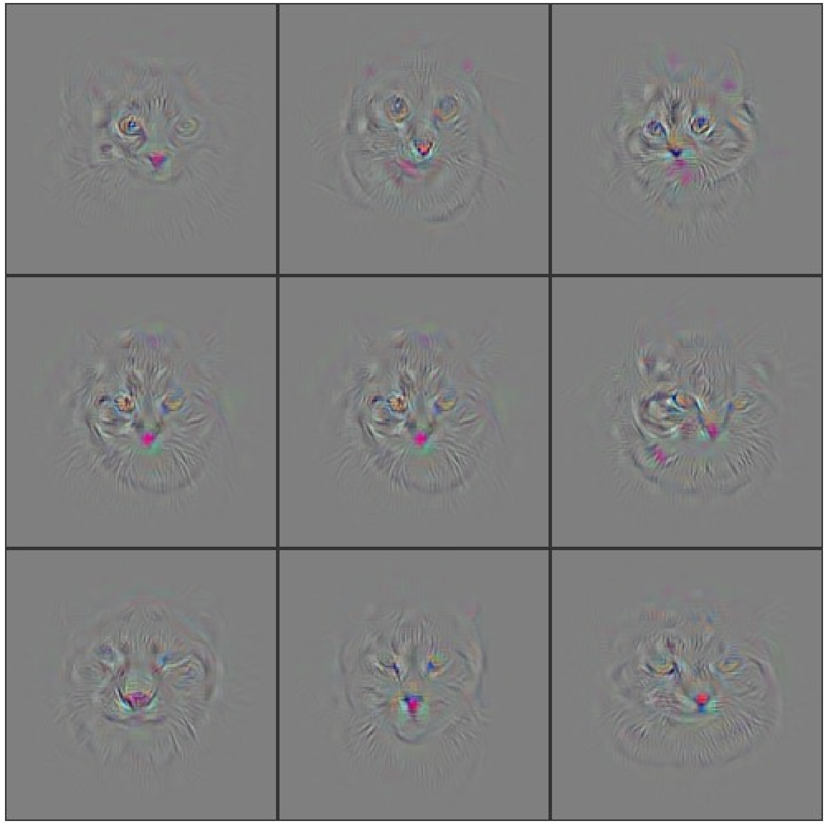
Test Performance of LeNet-5

- Original: 37.36% (trained by negative)
- Negative: 98.94% (trained by negative)

Compound Matched Filtering

What are the common salient regions of all 9 cat Images ?

Top 9 Input Activation Images



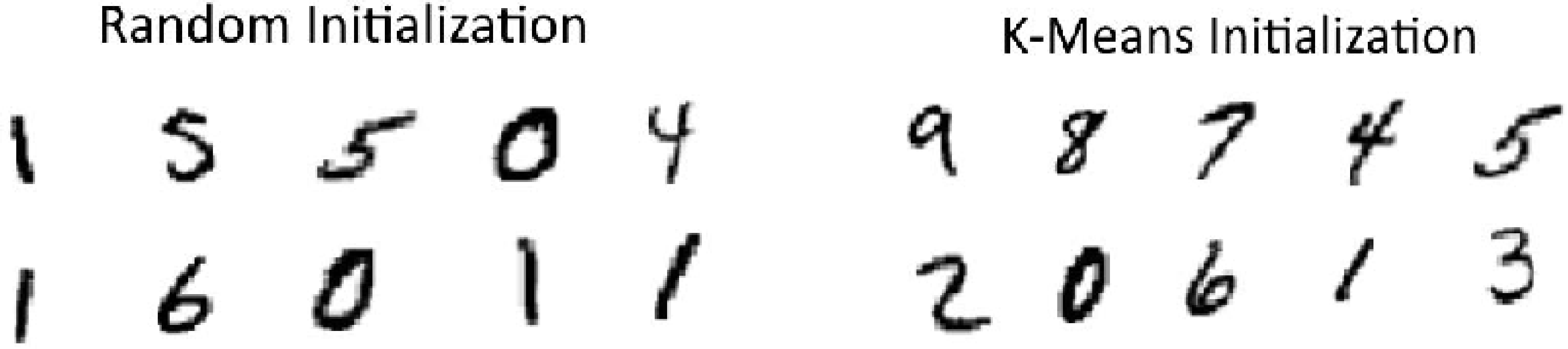
Deconv Image

Can CNN extract them automatically?

Self-Organization and Clustering

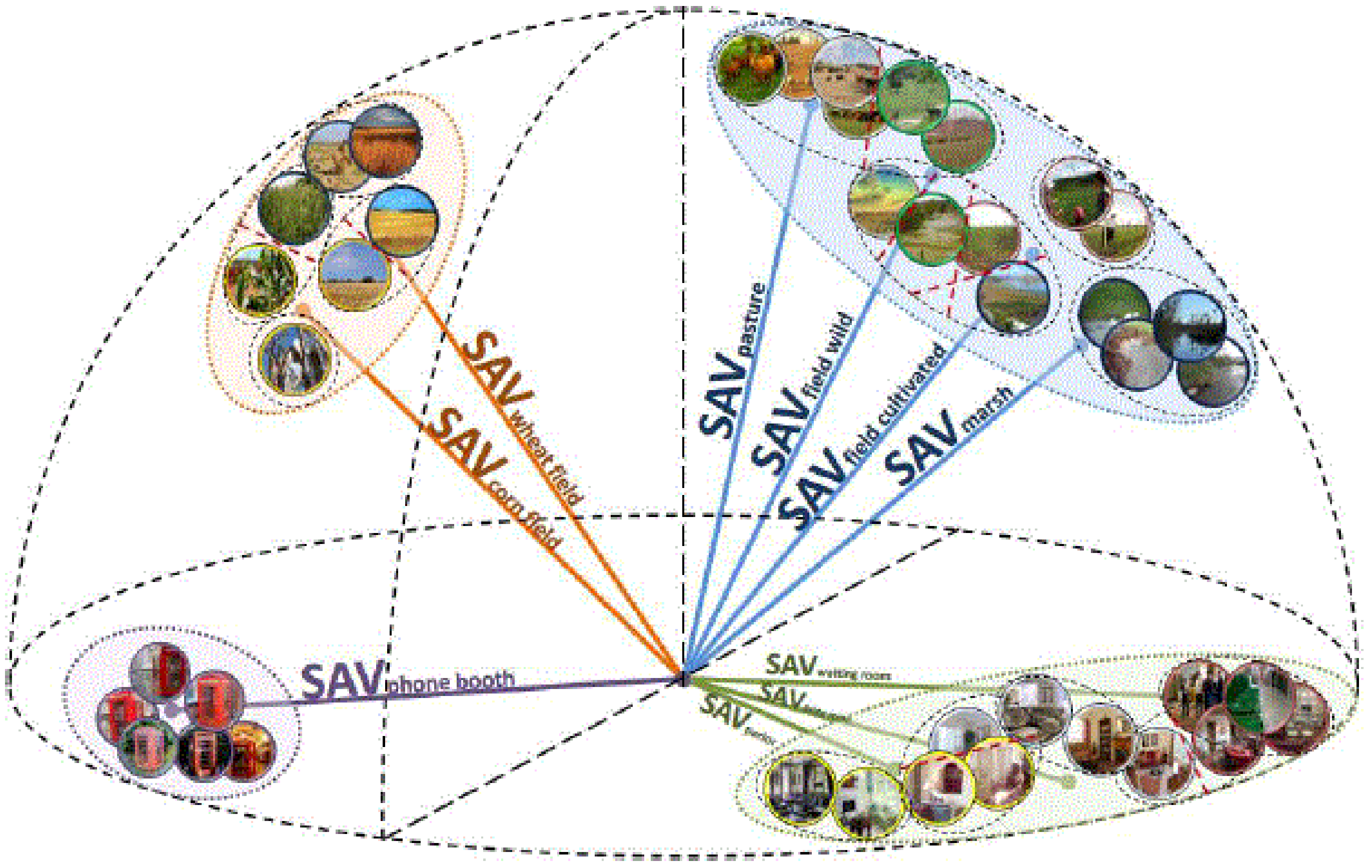
- Self-organization property
  - Learning without a teacher [1]
    - The network is repeatedly presented with a set of stimulus patterns to the input layer, but it does not receive any label about the patterns
    - One can cluster all kinds of dogs together without knowing their names
    - Unsupervised learning
  - This property was examined in depth in 80’s and 90’s, yet its significance is dropped in recent years
- CNN provides a wide spectrum solution
  - From un-supervised to weakly and heavily supervised learning paradigms

Comparison of LeNet-5 Initializations (2)

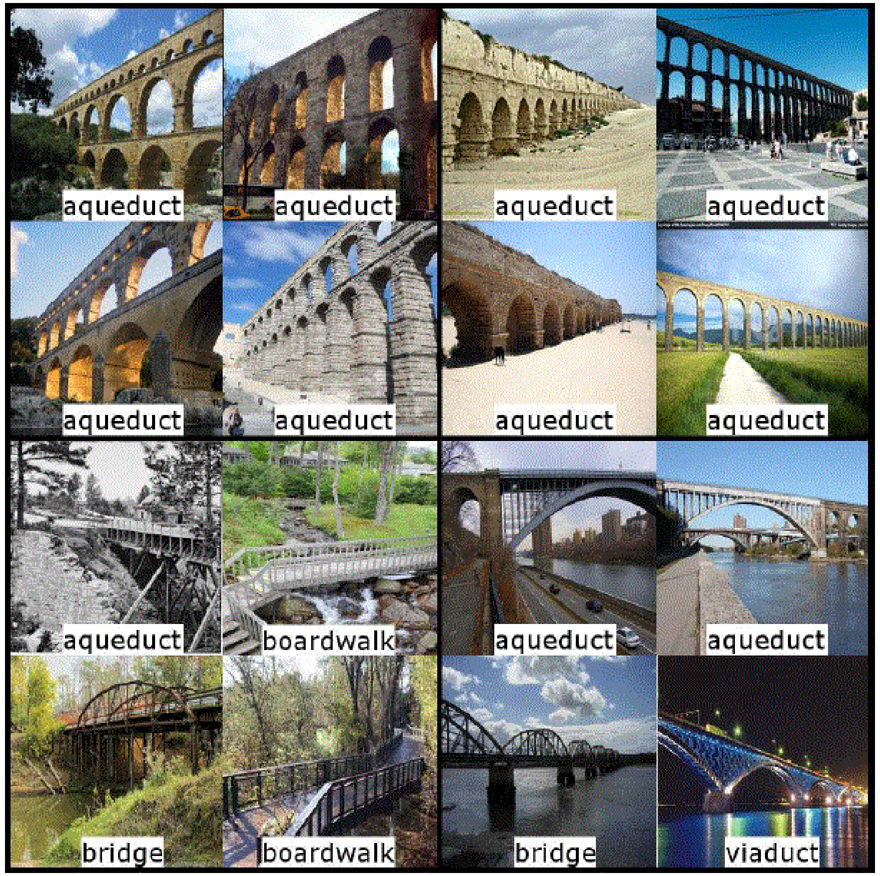


Scene/Object Anchor Vectors

Each anchor vector in the output stage is associated with a scene/object class label



Four Sub-classes under Aqueduct Class obtained via unsupervised split

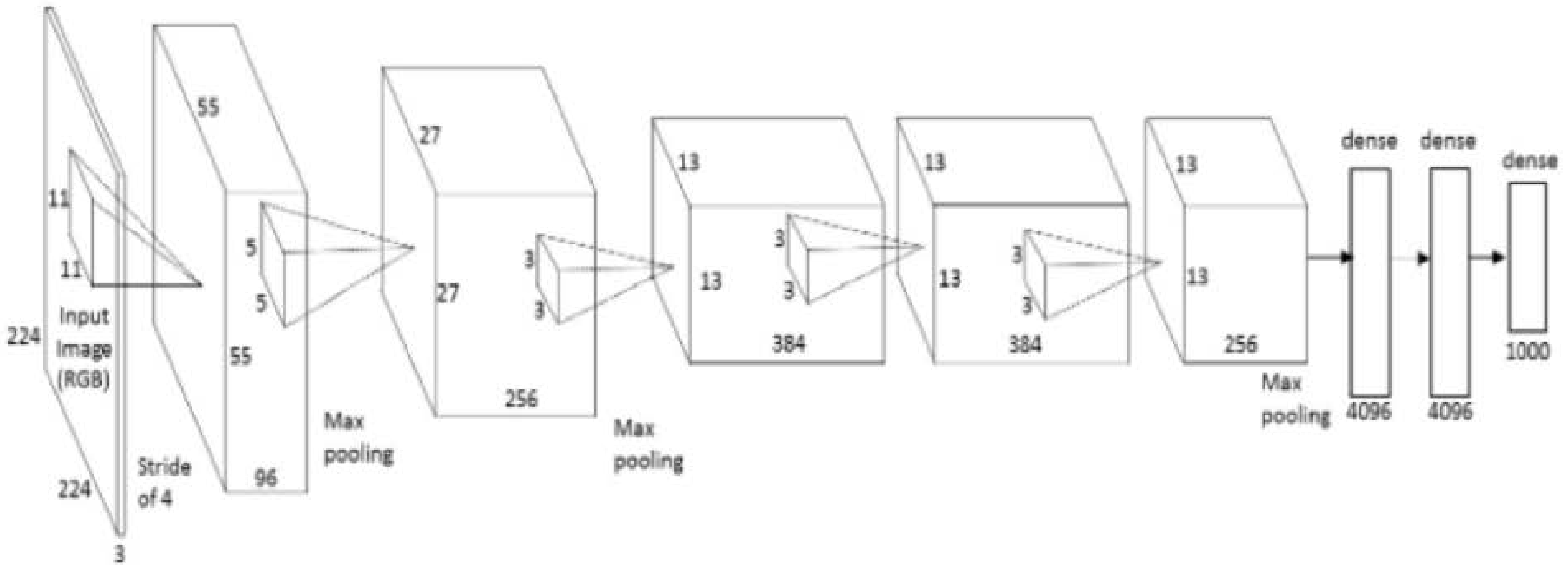


Unsupervised Split of the “Snake” Class

Guidance (BP) to Close "Semantic Gap"

Visually Similarity Clustering

Semantics Grouping

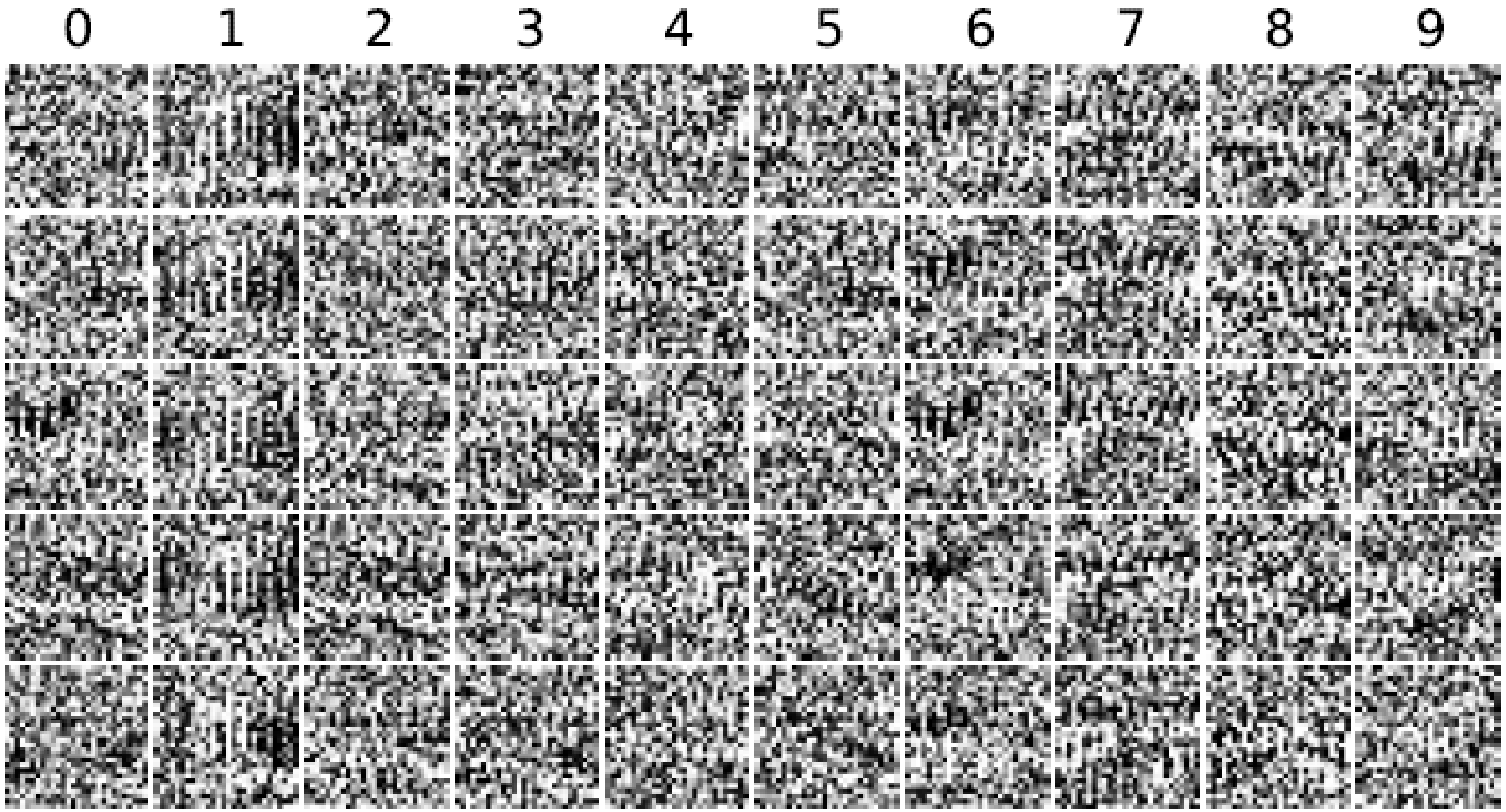


Part II: Why Not CNNs?

Reason 1: Robustness

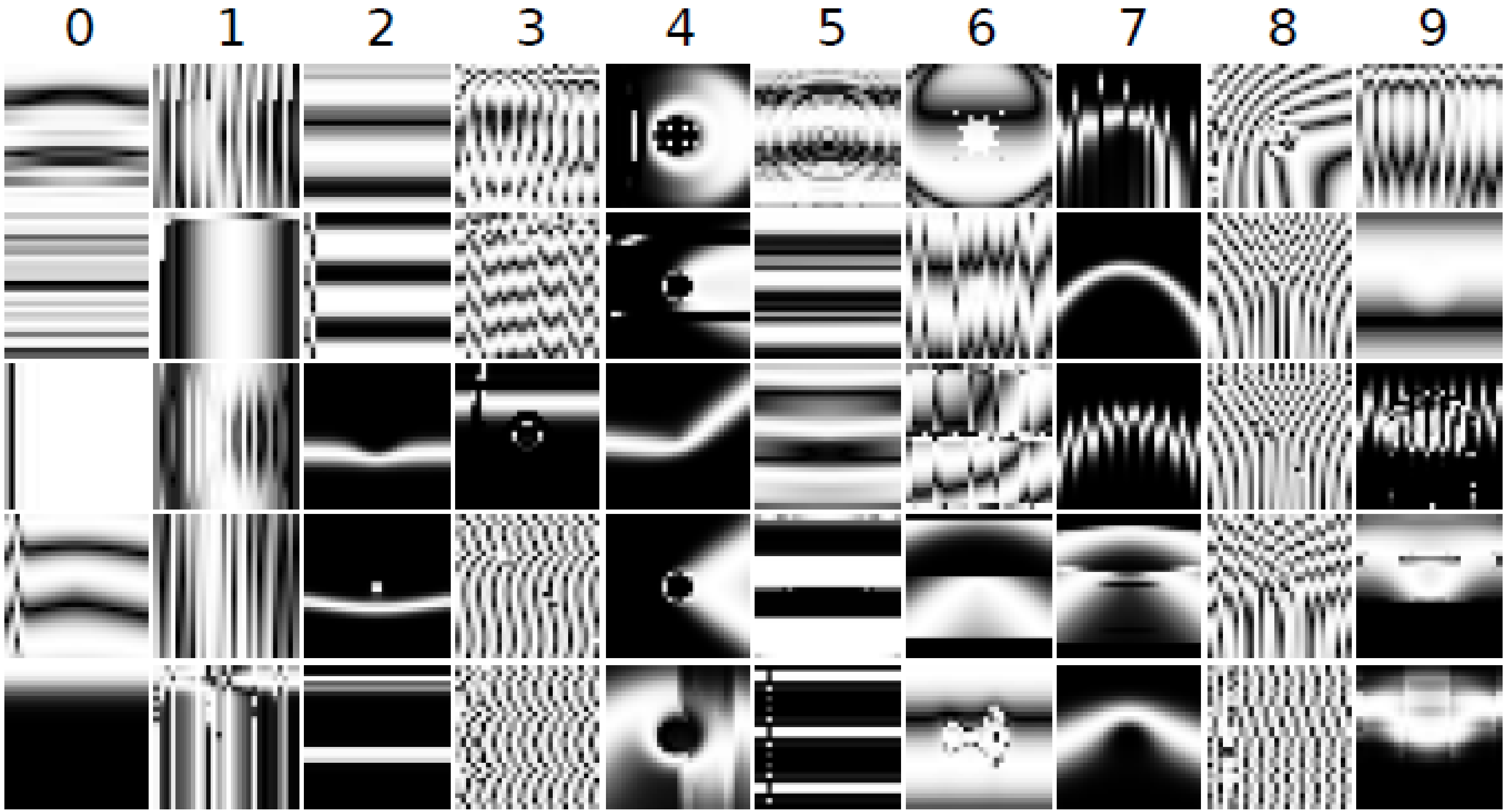
Easily fooled by adversarial perturbation

Example 1:



The CNN has 99.99% confidence in recognizing the images to be the digit in the top row

Example 2:



The CNN has 99.99% confidence in recognizing the images to be the digit in the top row

Nguyen, Anh, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

Reason 2: Scalability

- Scalability with the object class number
  - The ImageNet has 1000 object classes
  - What happens if we want to add or delete one class, the performance of the trained network drops?

- Scalability with the training samples
  - The ImageNet has 1.2 millions training images
  - What happens if we get more training samples?

- Need to re-train the network from the scratch

Reason 3: Portability

Example 1:

Example 2:

INRIA dataset (Dalal and Triggs, 2005)

Caltech dataset (Dollar et al., 2009)

CUHK Square dataset (Wang et al., 2012)

PETS 2009 dataset (Ferryman and Shahrokani, 2009)

What Goes Wrong?

- Feature extraction is not invertible

- Critical information is lost
  - Need to understand the information loss

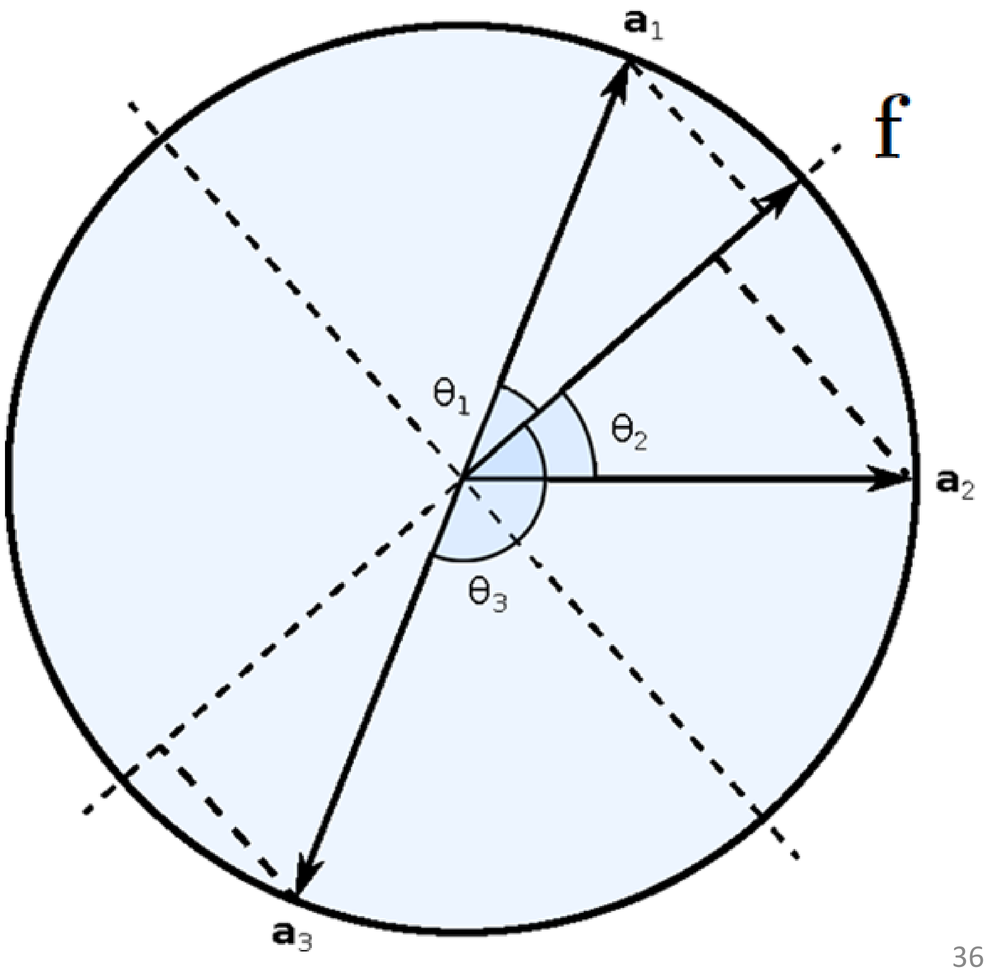
Inverse RECOS Transform

Can we reconstruct input X from its projected values?

$$p_k = \mathbf{a}_k^T \mathbf{f}, \quad k = 0, 1, \cdots K.$$

$$g_k = \begin{cases} p_k, & \text{if} \quad p_k > 0, \\ 0, & \text{if} \quad p_k \leqslant 0. \end{cases}$$

How to reconstruct **f** from p_k or g_k ?



Filter Weight Vectors Form A Signal Subspace

- Multiple filter weight vectors form a subspace
- Two key questions:
  - How to choose these filter weights?
  - How to determine their coefficients

Signal Subspace and Approximation Loss

If the number of anchor vectors is less than the dimension of input **f** , there is an approximation error

$$\mathbf{f} \approx \hat{\mathbf{f}} = \sum_{k=0}^{K} \alpha_k \mathbf{a}_k .$$

Filter weights as spanning vectors for a linear space

$$p_k \approx \mathbf{a}_k^T \hat{\mathbf{f}} = \mathbf{a}_k^T \left(\sum_{k'=0}^{K} \alpha_{k'} \mathbf{a}_{k'} \right)$$

How to Control Approximation Loss?

- Increase the number of anchor filters
- Find optimal anchor filters
  - Truncated Karhunen Loeve Transform (or PCA)
  - Orthogonal eigenvectors

$$\mathbf{a}_i^T \mathbf{a}_j = < \mathbf{a}_i, \mathbf{a}_j > = \delta_{i,j}$$

  - Easy to invert

$$\hat{\mathbf{f}} = \sum_{k=0}^{K} p_k \mathbf{a}_k,$$

Rectification Loss

- Due to Nonlinear Activation
  - Needed to resolve the sign confusion problem

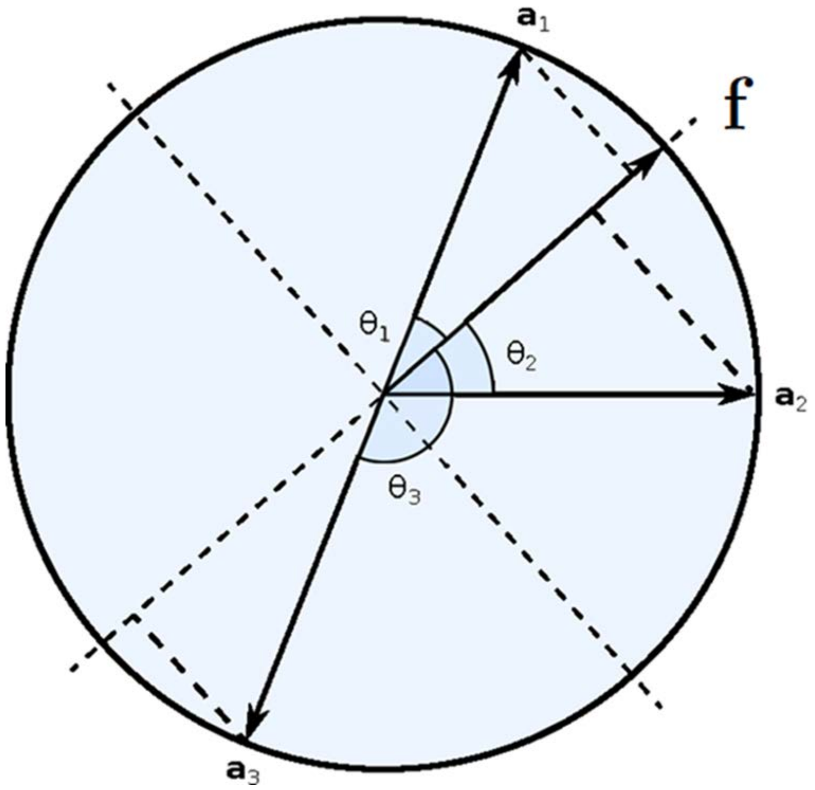
$$\mathbf{f}' = \sum_{q=0}^{Q} \beta_q \mathbf{a}'_q$$

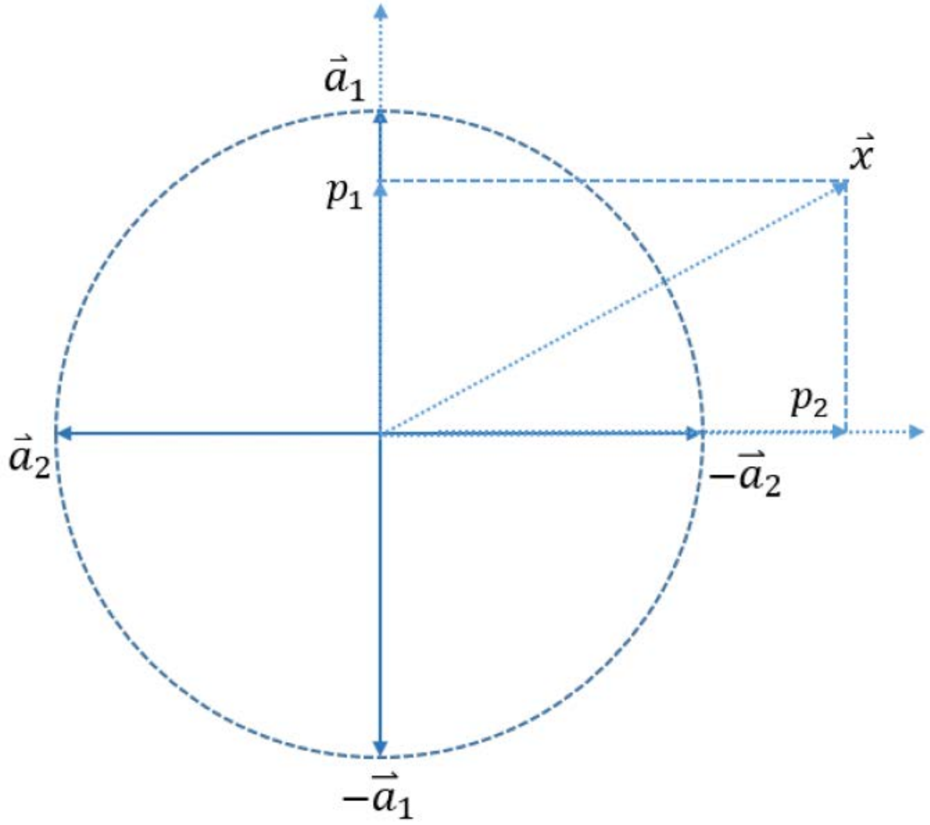
$$p_q \approx \mathbf{a}_q^T \left(\sum_{q'=0}^{Q} \beta_{q'} \mathbf{a}'_{q'} \right)$$

  - If we have the orthogonal basis, $p_q = \beta_q$

How to Overcome Rectification Loss

- Augment anchor vectors





Total Loss

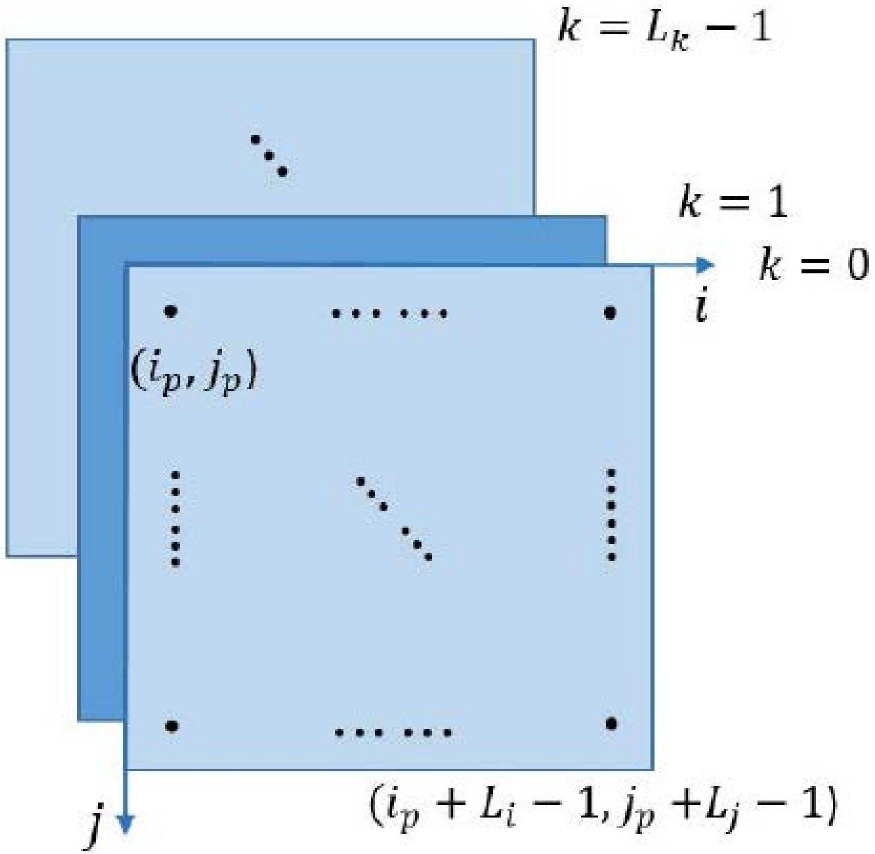
$$E(\mathbf{f}, \mathbf{f}') = ||\mathbf{f} - \mathbf{f}'||^2$$

$$= \underbrace{||\mathbf{f} - \hat{\mathbf{f}}||^2}_{\text{Approximation Loss}} + \underbrace{\cancelto{0}{||\hat{\mathbf{f}} - \mathbf{f}'||^2}}_{\text{Rectification Loss}} + \cancelto{0}{2(\mathbf{f} - \hat{\mathbf{f}})^T(\hat{\mathbf{f}} - \mathbf{f}')}$$

It is possible to remove all loss terms to obtain lossless transform

Saak Transform

- Subspace approximation with augmented kernels
- Input: functions defined on a cuboid
- Output: Saak coefficients



Saak Transform

Pre-processing:

Treat each input as a random vector, remove its mean and find the covariance matrix of these zero-mean random vectors

- **Step 1**: Obtain transform kernels via KLT analysis
- **Step 2**: Augment transform kernels with their negative vectors and compute transform coefficients by projection

$$\mathbf{a}_{2k-1} = \mathbf{b}_k, \quad \mathbf{a}_{2k} = -\mathbf{b}_k, \quad k = 1, \cdots, N-1.$$

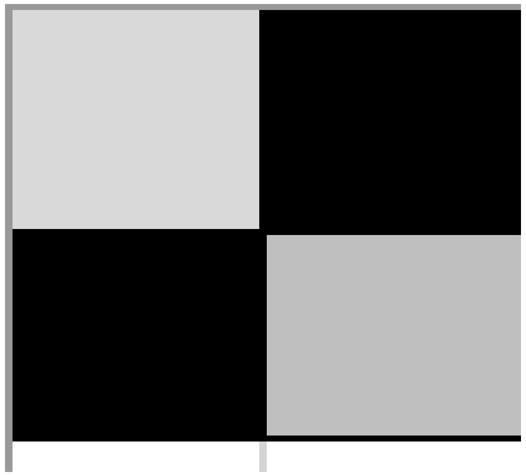
- **Step 3**: ReLU

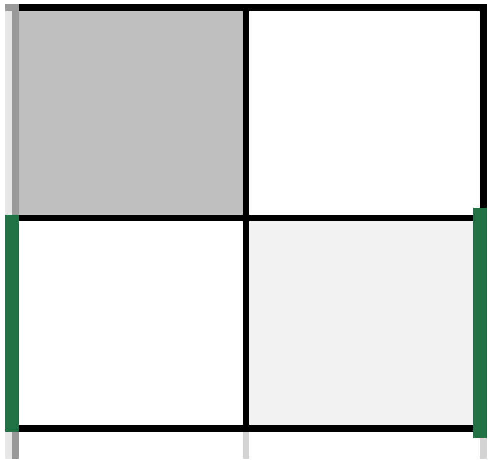
Sign-to-Position (S/P) Format Conversion

- Example:

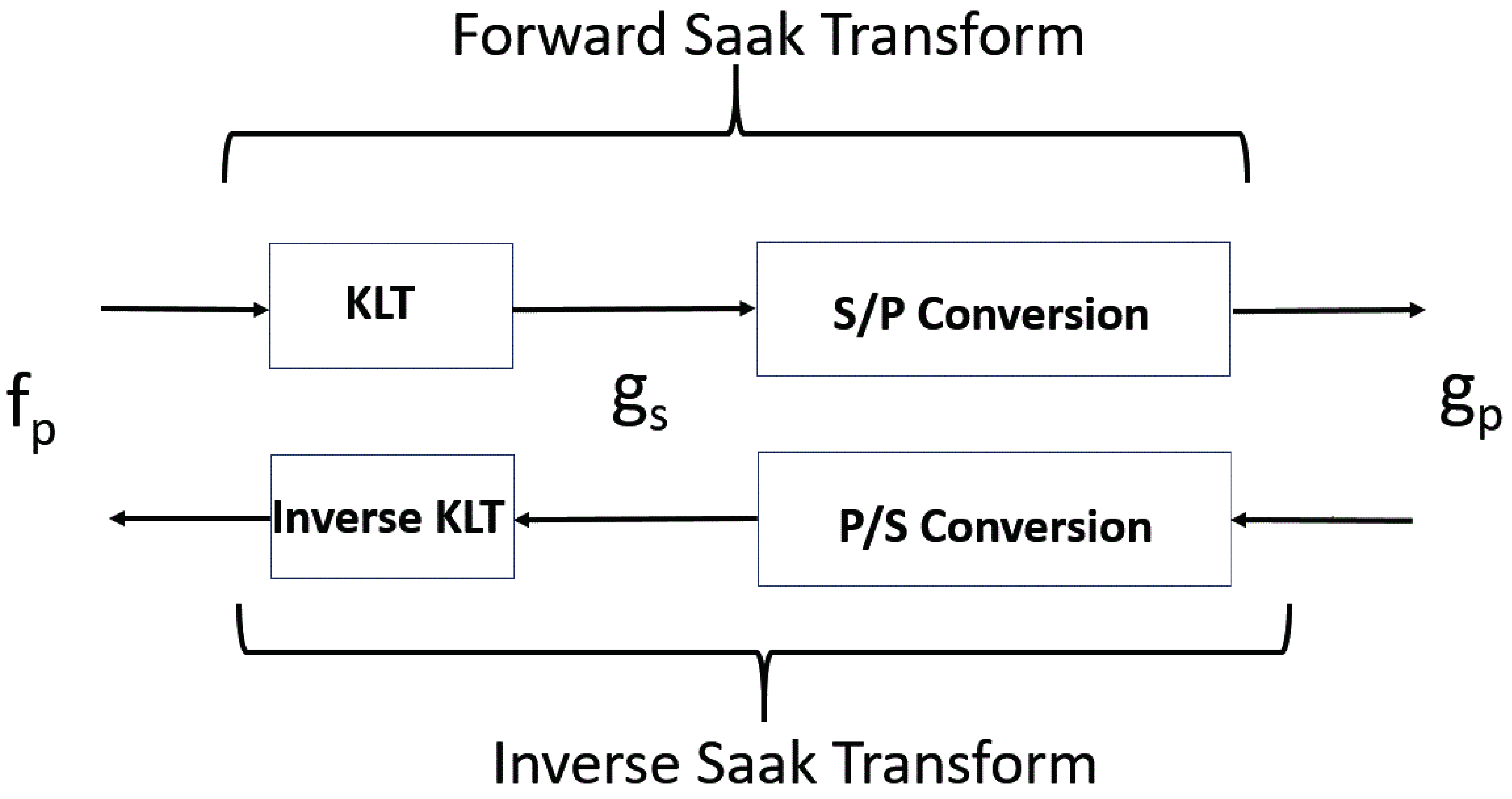
$$(\underline{5}, -3, \underline{-2}, 4)^T \longrightarrow (\underline{5,\ 0}, 0, 3, \underline{0,\ 2}, 4, 0)^T$$

- Physical meaning: They are two different patterns in biological systems



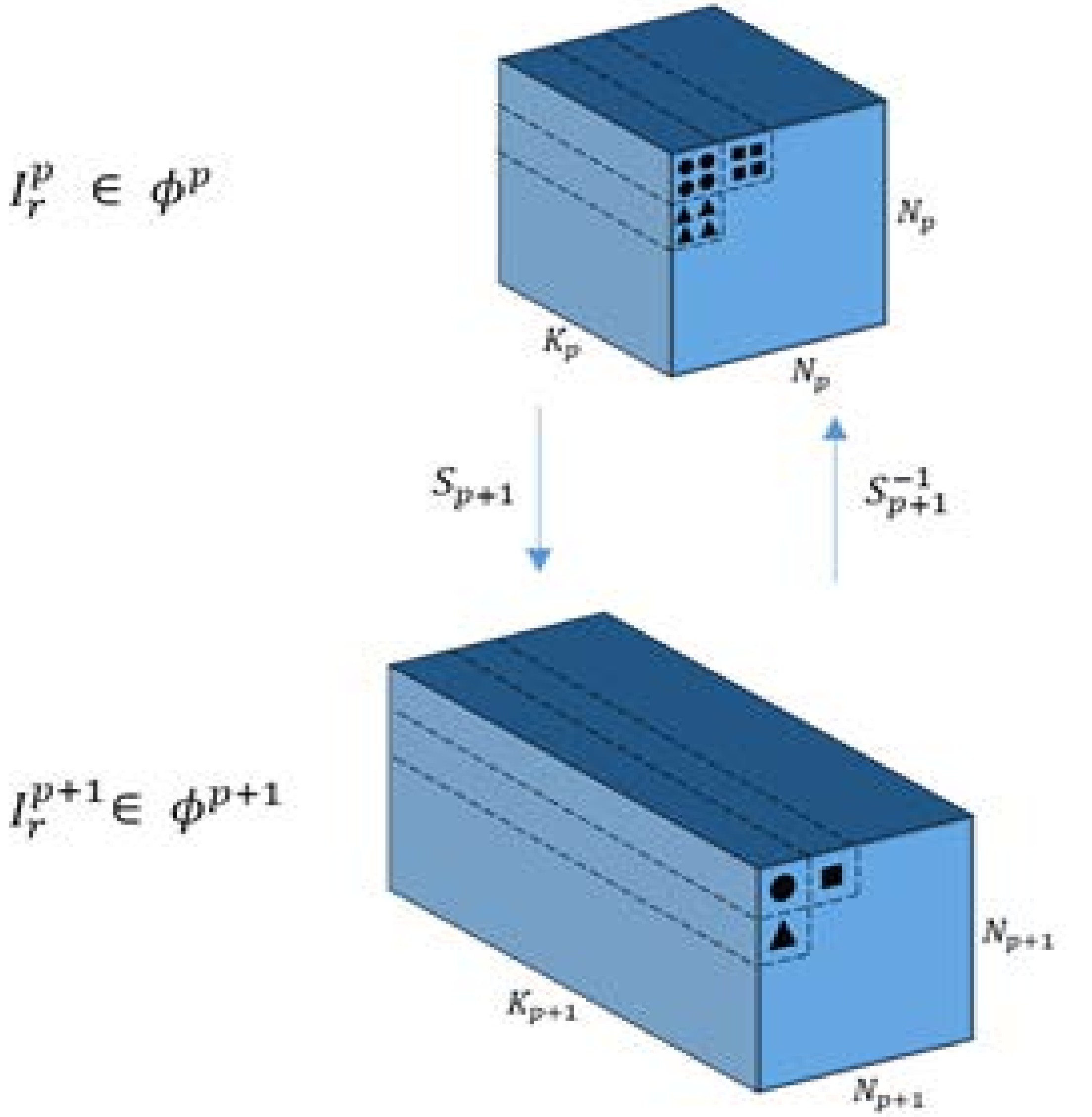


Saak Transform Computation



Semi-distance preserving property!

One-stage Saak Transform



Multistage Saak Transform

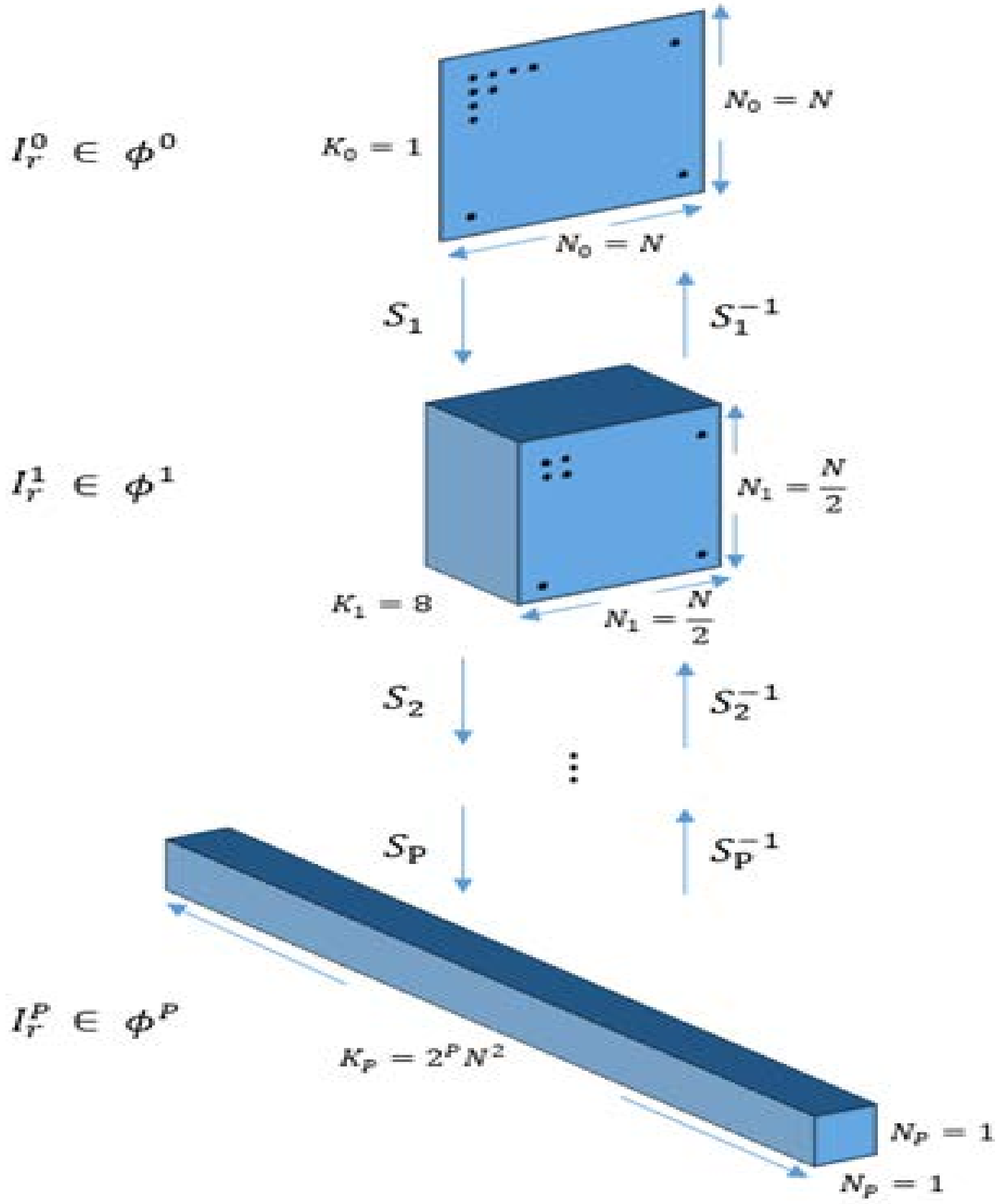


Image Synthesis via Inverse Saak Transform

Original Input

100 Saak Coefficients

500 Saak Coefficients

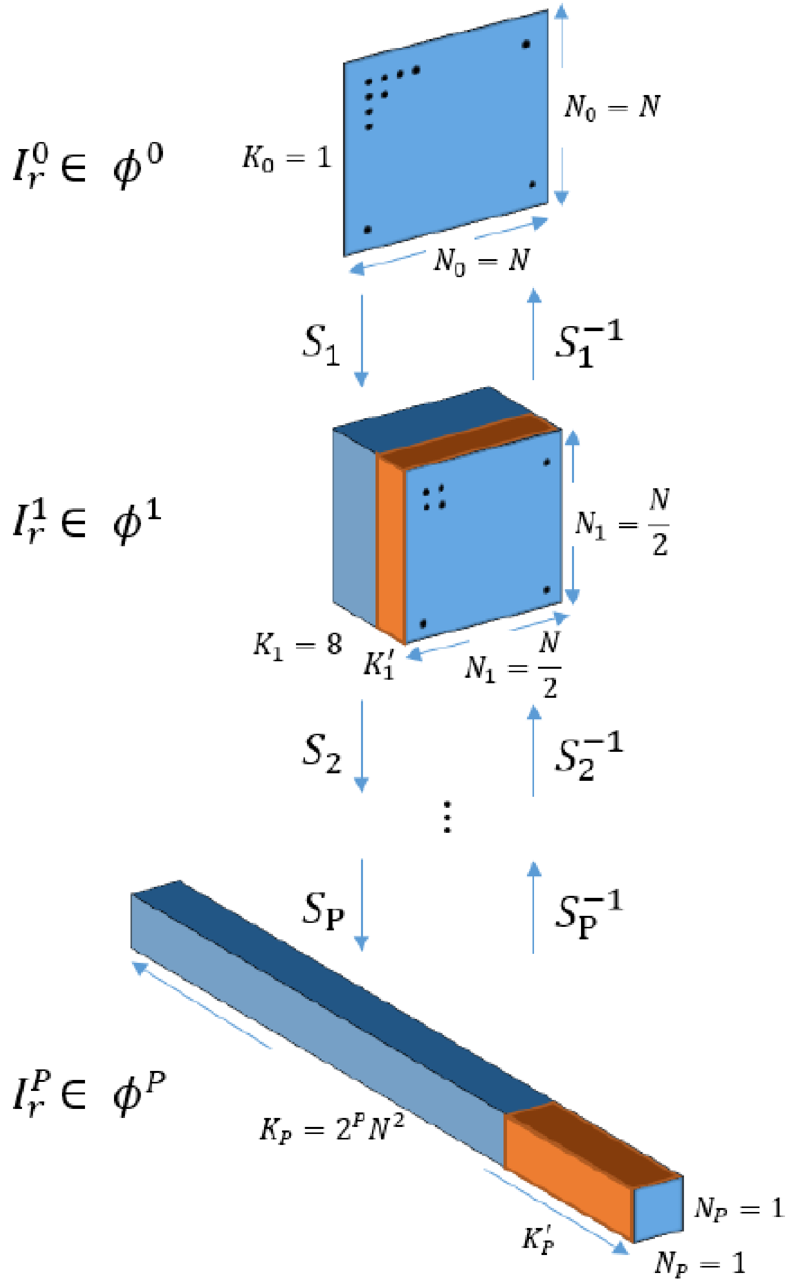
1000 Saak Coefficients

2000 Saak Coefficients

16,000 Saak Coefficients

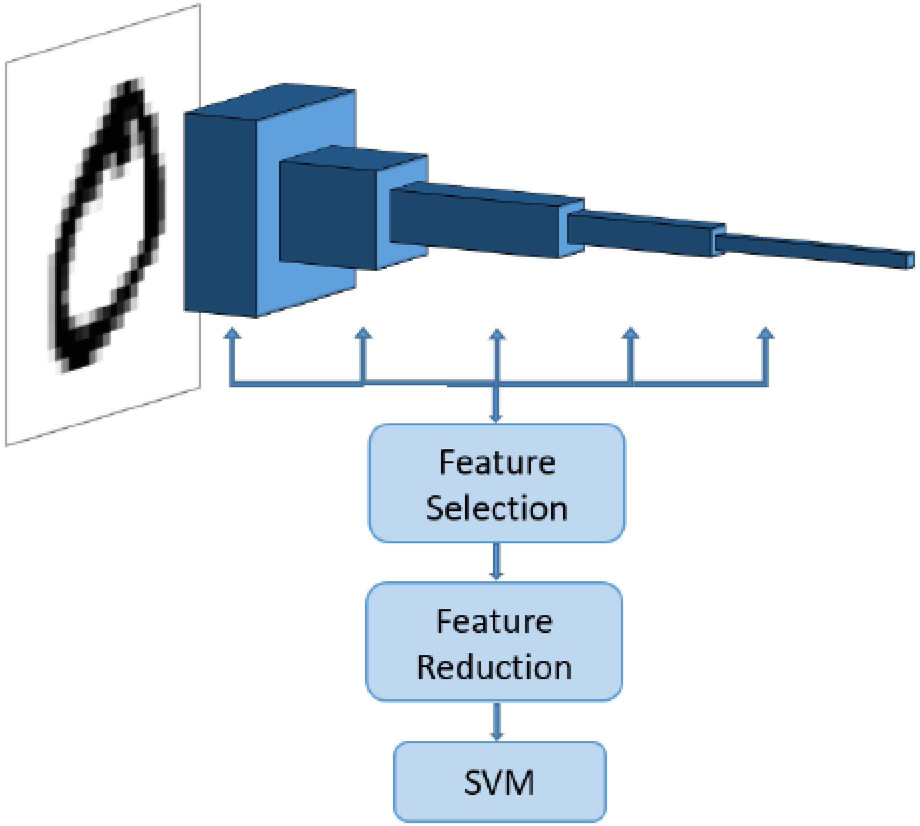
Lossy Saak Transform

- Thresholding
  - Based on the eigenvalue magnitude
    - Energy of individual Saak component
  - Based on the cumulative energy



Handwritten Digits Recognition

MNIST Dataset



Recognition Accuracy

	#Kernels for each stage	32	64	128	256
	All kernels	98.19	98.58	98.53	98.14
> 1%	(4, 11, 16, 20, 17)	98.24	98.54	98.33	97.84
> 3%	(4, 5, 8, 7, 9)	98.30	98.54	98.26	97.68
> 5%	(4, 5, 5, 6, 7)	98.28	98.52	98.21	97.70
> 7%	(4, 4, 4, 5, 5)	98.22	98.42	98.08	97.58

Weak Supervision

- Recognition Accuracy with setting (4, 5, 8, 7, 9)

Size	60000	5000	40000	30000	20000	10000
Accuracy	98.54	98.53	98.53	98.53	98.52	98.52

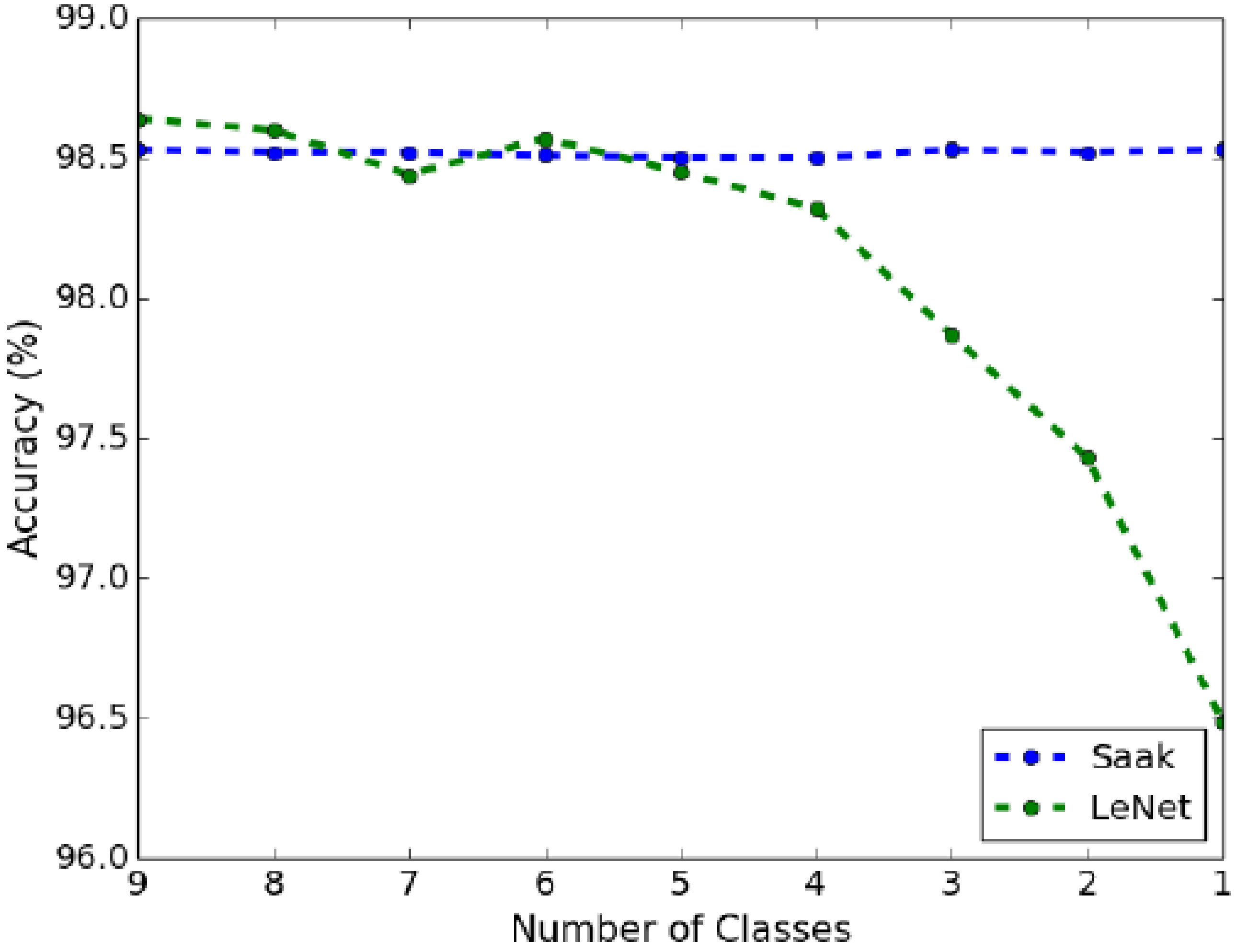
Little performance degradation in recognition accuracy

Scalability Against Object Classes

- Recognition Accuracy

LeNet-5: The convolutional layers are trained with fewer classes while the FC layers (i.e. MLP) are trained with all 10 object classes

Saak transform based approach: The Saak transform is obtained with fewer classes while the SVM is trained with all 10 object classes



Robustness

Method	S&P 1	S&P 2	S&P 3	S&P 4	Speckle	Gaussian	random_bg	texture_bg
LeNet-5	89.13	86.12	74.62	67.68	**84.10**	81.75	94.11	85.59
Saak	**95.71**	**95.31**	**91.16**	**87.49**	83.06	**94.08**	**94.67**	**87.78**

Comparison between CNN and Saak Transform (1)

End-to-end Optimization versus Modular Design

- CNN
  - Network architectures
  - Cost functions
  - Labeled data
- Saak transform
  - Feature extraction does not need data labels (only classifier training need data labels)
  - Return to traditional PR paradigm – “feature extraction” followed by “classifiers”
  - Better properties (robustness against perturbation, scalability against object classes, etc.)

Comparison between CNN and Saak Transform (2)

Generative Network versus Inverse Transform

- CNN: GAN
  - Training generative networks for image synthesis
  - Training is very tricky while performance is un-predictable
- Saak transform
  - Use the inverse Saak transform for image generation
  - Cross-domain image generation
    - Build the Saak transform for source images
    - Build the Saak transform for target images
    - Build a bridge between Saak coefficients of source/target domain
    - Source image -> forward Saak -> Saak coefficients of the source -> Saak coefficients of the target -> inverse Saak -> target image

Comparison between CNN and Saak Transform (3)

Theoretical Foundation

- CNN: Mathematically intractable
- Saak transform: Mathematically transparent
    - Probability and statistics (covariance estimation & sampling)
    - Linear algebra (KLT/PCA)
    - Transform theory (signal representation, forward/inverse transform)

Comparison between CNN and Saak Transform (4)

Other Considerations

- Computational complexity in Saak kernel determination
  - Significantly faster than CNN filter weight determination
  - No GPU is needed
- Weakly supervised learning
  - Less dependent on labeled data
    - ImageNet is actually an "outlier" in big data analytics! Difficult to find another one

Take Home Lessons

- Filter weights as a matched filter
  - K-means clustering to determine the filter weights
- Filter weights as vectors to span a signal subspace
  - PCA to determine the filter weights
- Nonlinear activation
  - Needed to resolve the sign confusion problem in cascaded networks
- Saak transform
  - Being motivated by CNNs
  - A signal transform for automatic feature extraction (neither being handcrafted nor being learned by BP)

Conclusion & Future Work

- Conclusion
  - The Saak transform was inspired by CNNs
    - An unsupervised feature extraction tool
    - Need to address the segmentation problem explicitly

- Future work
  - Extensive evaluation between Saak-based and CNN-based methodologies
  - Joint compression and understanding
  - What is the mechanism behind RNNs?

Main References

- C.-C. Jay Kuo, "Understanding convolutional neural networks with a mathematical model," the Journal of Visual Communications and Image Representation, Vol. 41, pp. 406-413, November 2016.
- C.-C. Jay Kuo, "The CNN as guided multi-layer RECOS transform," the IEEE Signal Processing Magazine, Vol. 34, No. 3, pp. 81-89, May 2017.
- C.-C. Jay Kuo and Yueru Chen, "On data-driven Saak transform," arXiv preprint arXiv:1710.04176 (2017). Also to appear in the Journal of Visual Communications and Image Representation
- Yueru Chen, Zhuwei Xu, Shanshan Cai, Yujian Lang and C.-C. Jay Kuo, "The Saak transform approach to efficient, scalable and robust handwritten digits recognition," arXiv preprint arXiv:1710.10714 (2017).

Codes Available at Github

- https://github.com/davidsonic/Saak-Transform