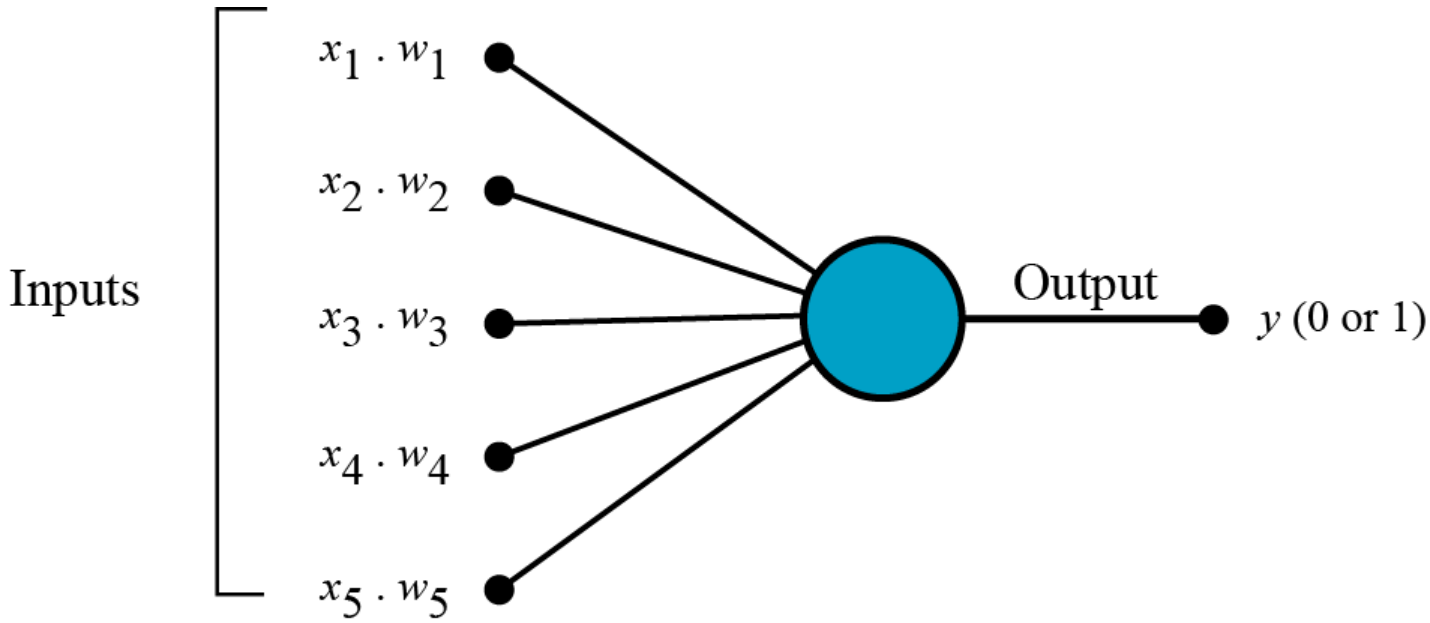
Lecture 8 : The Topology of Neural Nets and Deep Learning

May 16, 2019

Gunnar Carlsson

Stanford University

Perceptrons



Perceptrons

- Input variables are initially Boolean, as is Output

Perceptrons

- Input variables are initially Boolean, as is Output
- $$\text{Output} = \text{sign}(\sum w_i x_i)$$

Perceptrons

- Input variables are initially Boolean, as is Output
- $$\text{Output} = \text{sign}(\sum w_i x_i)$$
- Given an output function on data, "train" the weights w_i to best approximate a given Boolean function

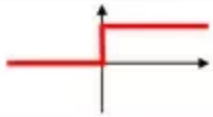
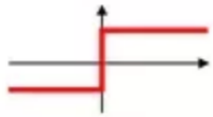
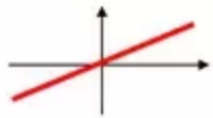
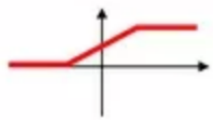
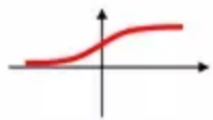
Perceptrons

- Input variables are initially Boolean, as is Output
- $$\text{Output} = \text{sign}(\sum w_i x_i)$$
- Given an output function on data, "train" the weights w_i to best approximate a given Boolean function
- Ultimately generalize to include continuous functions as well as Boolean ones

Perceptrons

- Input variables are initially Boolean, as is Output
- $$\text{Output} = \text{sign}(\sum w_i x_i)$$
- Given an output function on data, "train" the weights w_i to best approximate a given Boolean function
- Ultimately generalize to include continuous functions as well as Boolean ones
- Couple together to get networks of perceptrons

Activation Functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Neural Networks

- Idea is to hook several perceptrons together to create a more complicated and expressive computational unit

Neural Networks

- Idea is to hook several perceptrons together to create a more complicated and expressive computational unit
- Pattern can be encoded by a directed graph

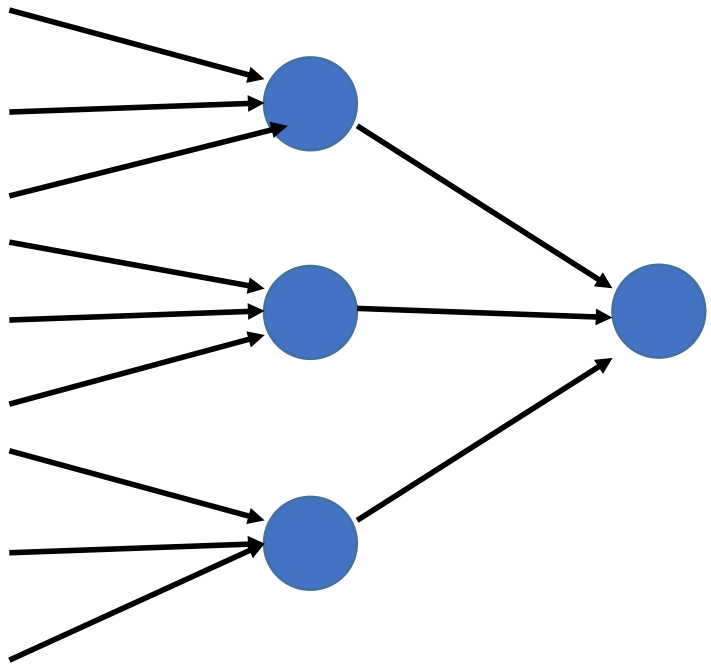
Neural Networks

- Idea is to hook several perceptrons together to create a more complicated and expressive computational unit
- Pattern can be encoded by a directed graph
- Each node is called a neuron. It has a state, which can be Boolean or continuous

Neural Networks

- Idea is to hook several perceptrons together to create a more complicated and expressive computational unit
- Pattern can be encoded by a directed graph
- Each node is called a neuron. It has a state, which can be Boolean or continuous
- State is updated based on graph connections

Neural Networks



Neural Networks

- An architecture is a choice of graph structure

Neural Networks

- An architecture is a choice of graph structure
- Assignment of weights to edges

Neural Networks

- An architecture is a choice of graph structure
- Assignment of weights to edges
- Different architectures for different types of data and different types of problems

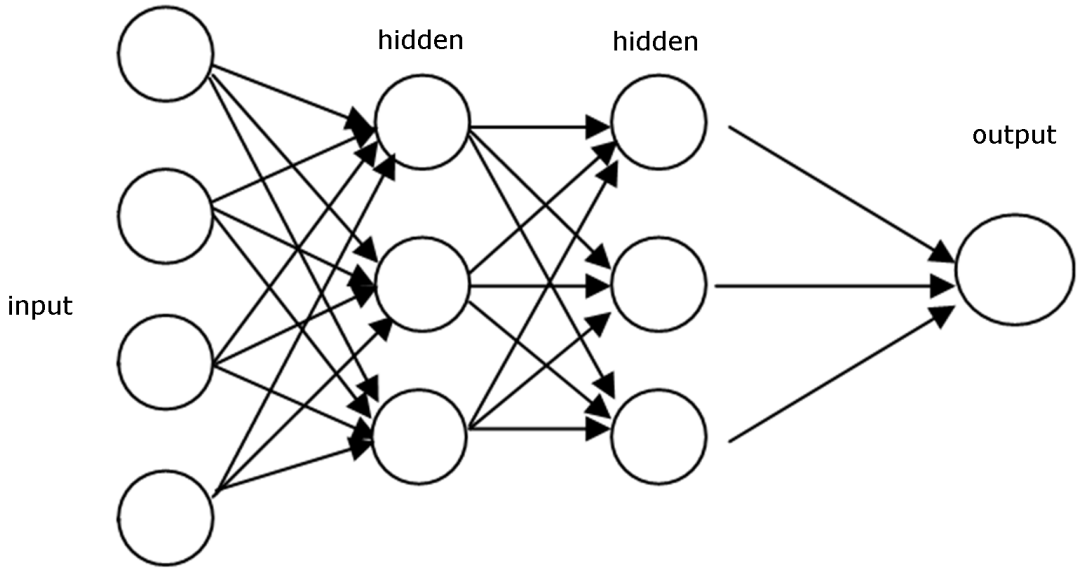
Neural Networks

- An architecture is a choice of graph structure
- Assignment of weights to edges
- Different architectures for different types of data and different types of problems
- Training is done by performing a stochastic version of gradient descent on the weights

Neural Networks

- An architecture is a choice of graph structure
- Assignment of weights to edges
- Different architectures for different types of data and different types of problems
- Training is done by performing a stochastic version of gradient descent on the weights
- *Many* variable optimization problem

Feed Forward Networks



Feed Forward Networks

- Designed to compute composites of simple functions, i.e. of functions computed by individual perceptrons

Feed Forward Networks

- Designed to compute composites of simple functions, i.e. of functions computed by individual perceptrons
- Neurons are divided into layers

Feed Forward Networks

- Designed to compute composites of simple functions, i.e. of functions computed by individual perceptrons
- Neurons are divided into layers
- Collection of layers are given a total ordering

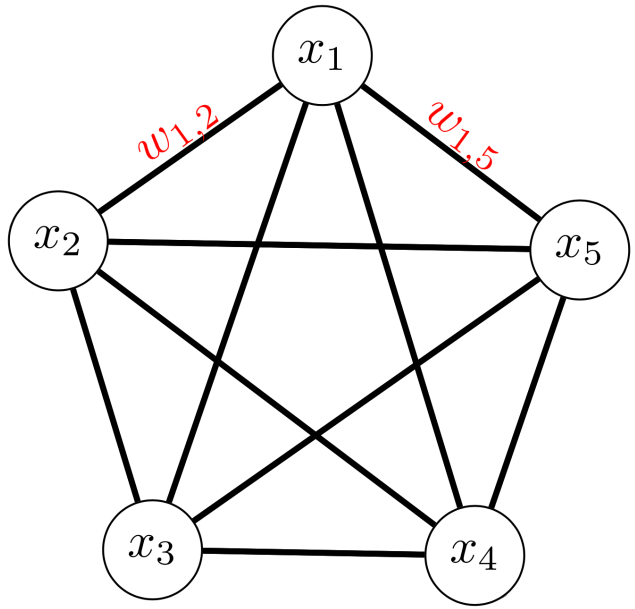
Feed Forward Networks

- Designed to compute composites of simple functions, i.e. of functions computed by individual perceptrons
- Neurons are divided into layers
- Collection of layers are given a total ordering
- There is an edge from neuron A to neuron B if and only if the layer containing B immediately follows the layer containing A.

Feed Forward Networks

- Designed to compute composites of simple functions, i.e. of functions computed by individual perceptrons
- Neurons are divided into layers
- Collection of layers are given a total ordering
- There is an edge from neuron A to neuron B if and only if the layer containing B immediately follows the layer containing A.
- One optimizes a fit of a function of this form to given function over the space of all possible weights

Hopfield Networks



Hopfield Networks and Boltzmann Machines

- Fully connected, although one may enforce the fact that some edges are given zero weight

Hopfield Networks and Boltzmann Machines

- Fully connected, although one may enforce the fact that some edges are given zero weight
- Idea is to learn to recognize examples, or a distribution on a set

Hopfield Networks and Boltzmann Machines

- Fully connected, although one may enforce the fact that some edges are given zero weight
- Idea is to learn to recognize examples, or a distribution on a set
- The examples are local minima of an energy function

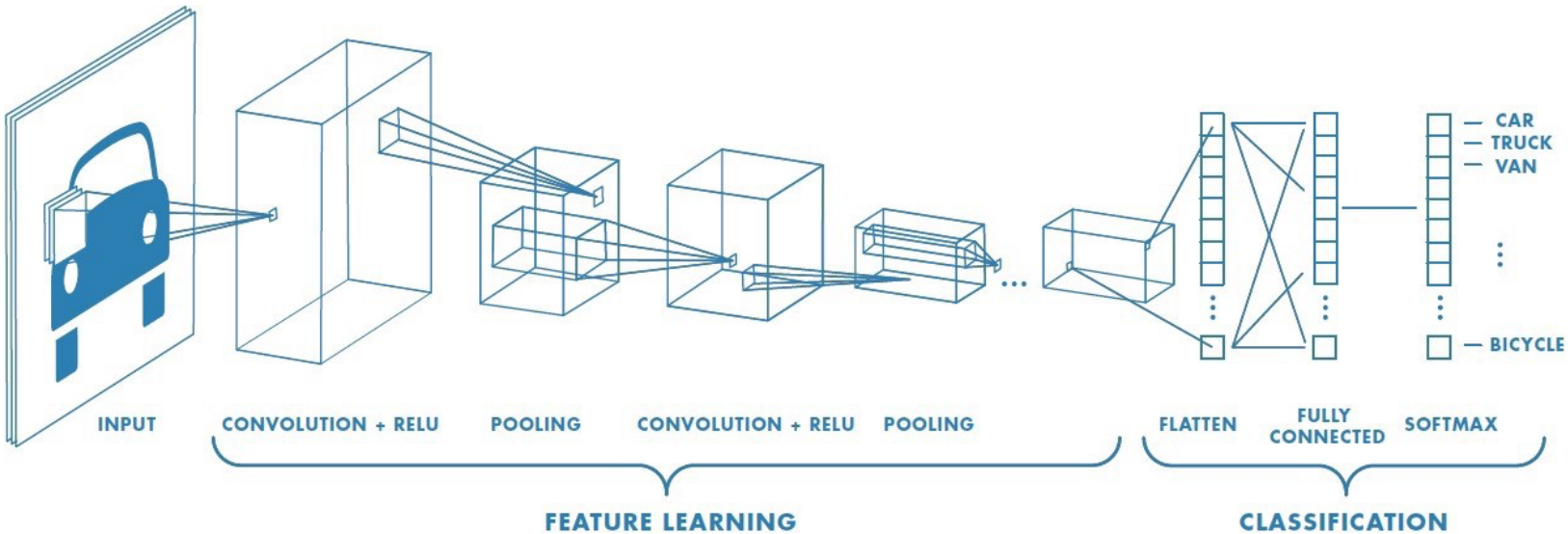
Hopfield Networks and Boltzmann Machines

- Fully connected, although one may enforce the fact that some edges are given zero weight
- Idea is to learn to recognize examples, or a distribution on a set
- The examples are local minima of an energy function
- A dynamical system is created which flows to the minima of the energy function

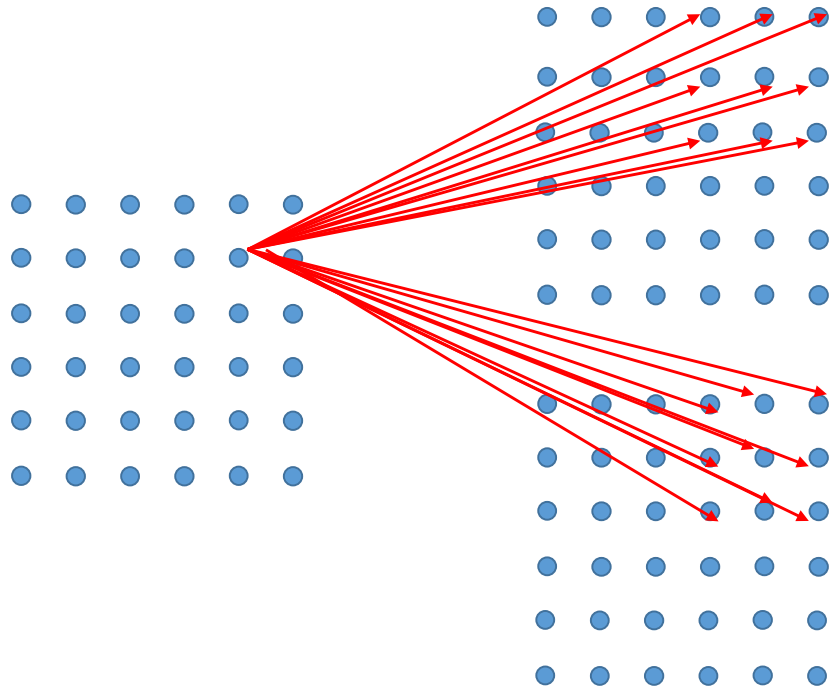
Hopfield Networks and Boltzmann Machines

- Fully connected, although one may enforce the fact that some edges are given zero weight
- Idea is to learn to recognize examples, or a distribution on a set
- The examples are local minima of an energy function
- A dynamical system is created which flows to the minima of the energy function
- An initial point which is close to one of the examples would naturally flow to the corresponding local minimum

Convolutional Neural Networks



Convolutional Neural Networks



Locality

Convolutional Neural Networks

- Locality is a restriction on the architecture

Convolutional Neural Networks

- Locality is a restriction on the architecture
- Uses geometry of feature space (grid geometry) to restrict the connections

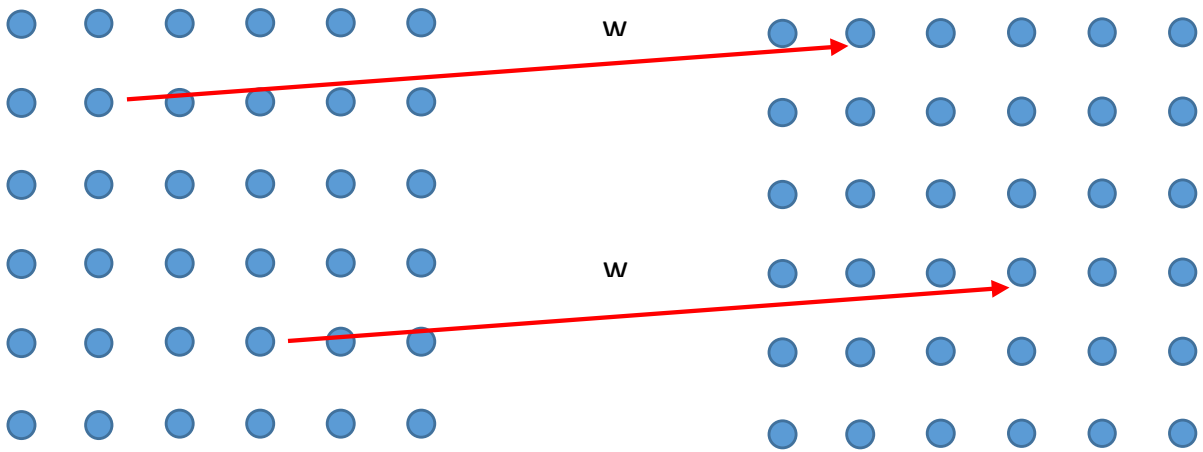
Convolutional Neural Networks

- Locality is a restriction on the architecture
- Uses geometry of feature space (grid geometry) to restrict the connections
- Means that the features "learned" will be local, i.e. will involve only the neighborhoods in the picture

Convolutional Neural Networks

- Locality is a restriction on the architecture
- Uses geometry of feature space (grid geometry) to restrict the connections
- Means that the features "learned" will be local, i.e. will involve only the neighborhoods in the picture
- Restriction means models are smaller and learning is faster than fully connected situation

Convolutional Neural Networks



Homogeneity or "Convolutionality"

Convolutional Neural Networks

- Restriction on assignment of weights, not the architecture

Convolutional Neural Networks

- Restriction on assignment of weights, not the architecture
- Means training will be a constrained optimizaton problem

Convolutional Neural Networks

- Restriction on assignment of weights, not the architecture
- Means training will be a constrained optimizaton problem
- Means that recognizing a cat is done the same way regardless of the position of the cat

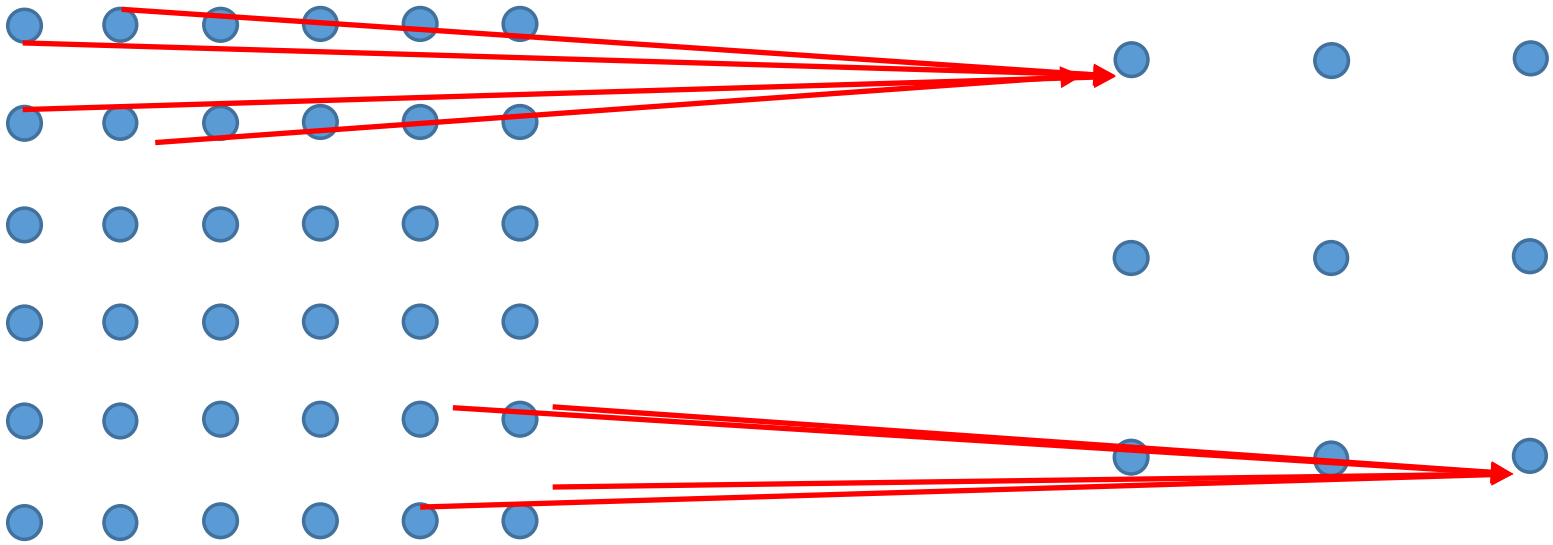
Convolutional Neural Networks

- Restriction on assignment of weights, not the architecture
- Means training will be a constrained optimizaton problem
- Means that recognizing a cat is done the same way regardless of the position of the cat
- Exploits symmetries in space of features

Convolutional Neural Networks

- Restriction on assignment of weights, not the architecture
- Means training will be a constrained optimizaton problem
- Means that recognizing a cat is done the same way regardless of the position of the cat
- Exploits symmetries in space of features
- Can exploit this idea for other feature geometries

Convolutional Neural Networks



Pooling

Convolutional Neural Networks

- Pooling creates a lower resolution and higher abstract version

Convolutional Neural Networks

- Pooling creates a lower resolution and higher abstract version
- Such layers are interspersed between convolutional layers

Convolutional Neural Networks

- Pooling creates a lower resolution and higher abstract version
- Such layers are interspersed between convolutional layers
- Mimics our understanding of how human visual pathway works

Problems with Deep Learning

- Long training time

Problems with Deep Learning

- Long training time
- Requires a *lot* of data

Problems with Deep Learning

- Long training time
- Requires a *lot* of data
- Models are often very large

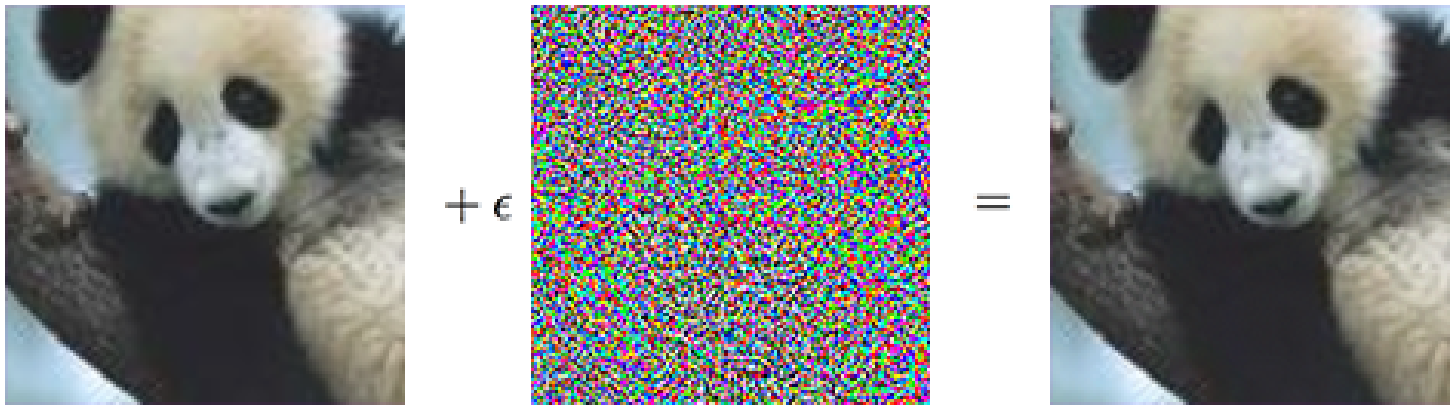
Problems with Deep Learning

- Long training time
- Requires a *lot* of data
- Models are often very large
- Don't generalize

Problems with Deep Learning

- Long training time
- Requires a *lot* of data
- Models are often very large
- Don't generalize
- Adversarial Examples

Adversarial Examples



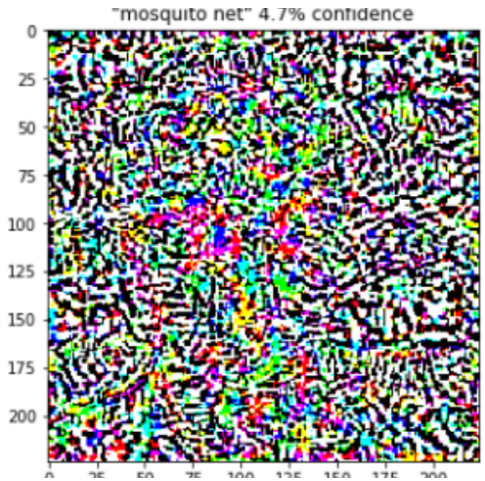
"panda"

57.7% confidence

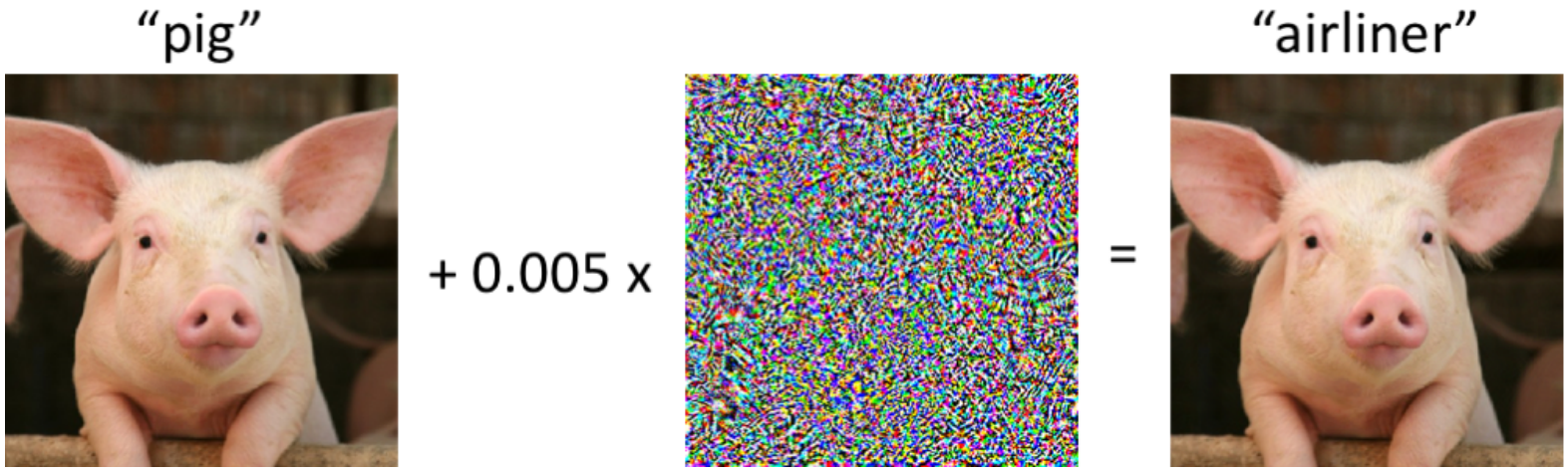
"gibbon"

99.3% confidence

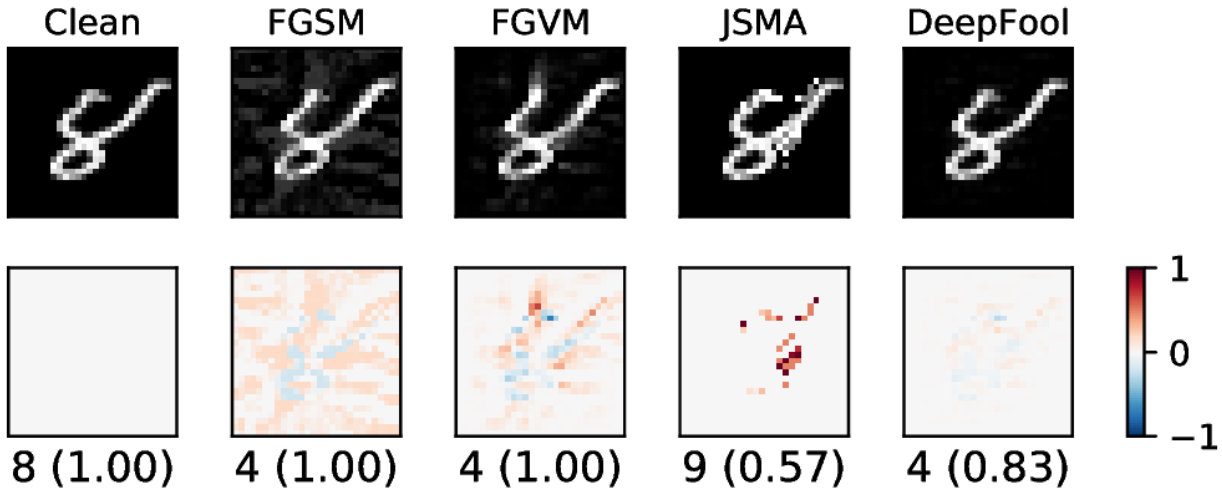
Adversarial Examples



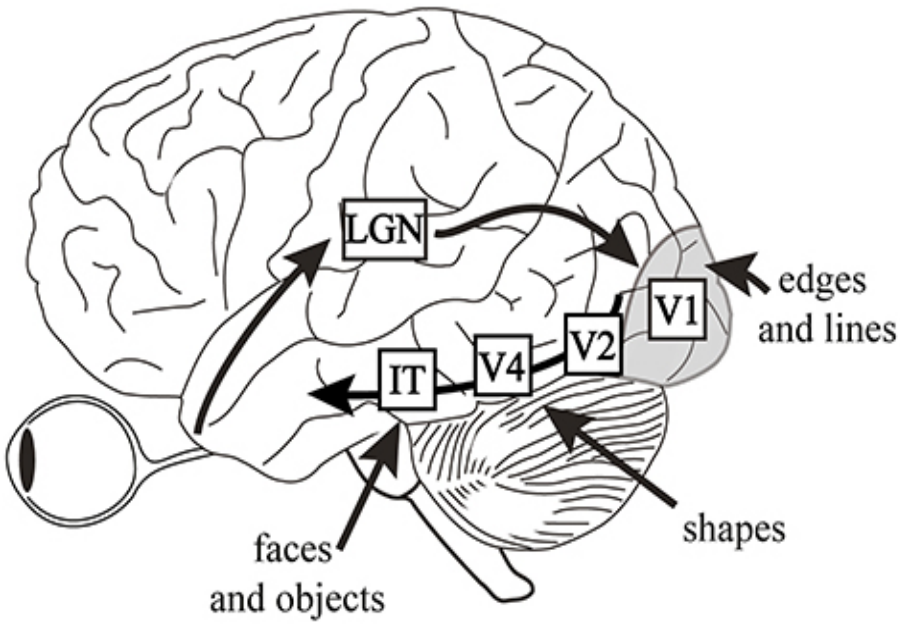
Adversarial Examples

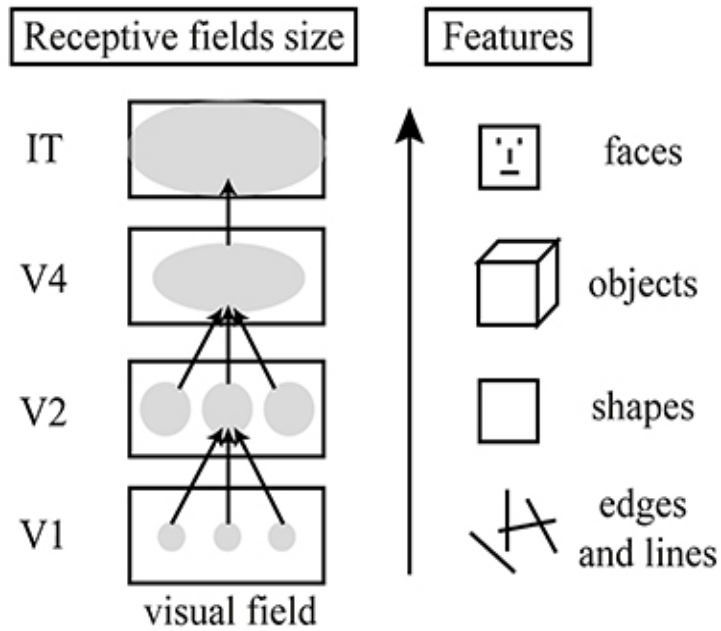


Adversarial Examples

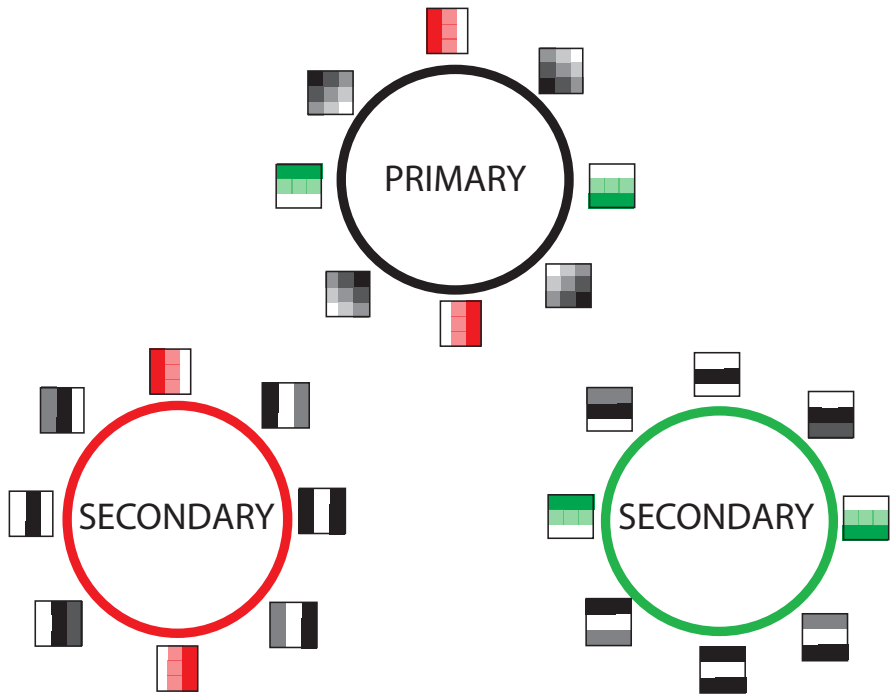


Do CNN's Work Like Mammalian Visual Pathway?





DoCNN's Reflect Natural Image Statistics?



Creating Data Sets from CNN's

- Consider convolutional neural nets with 9 connections for each pixel

Creating Data Sets from CNN's

- Consider convolutional neural nets with 9 connections for each pixel
- Construct 9-vectors for each pixel

Creating Data Sets from CNN's

- Consider convolutional neural nets with 9 connections for each pixel
- Construct 9-vectors for each pixel
- Many grids in first and second layers

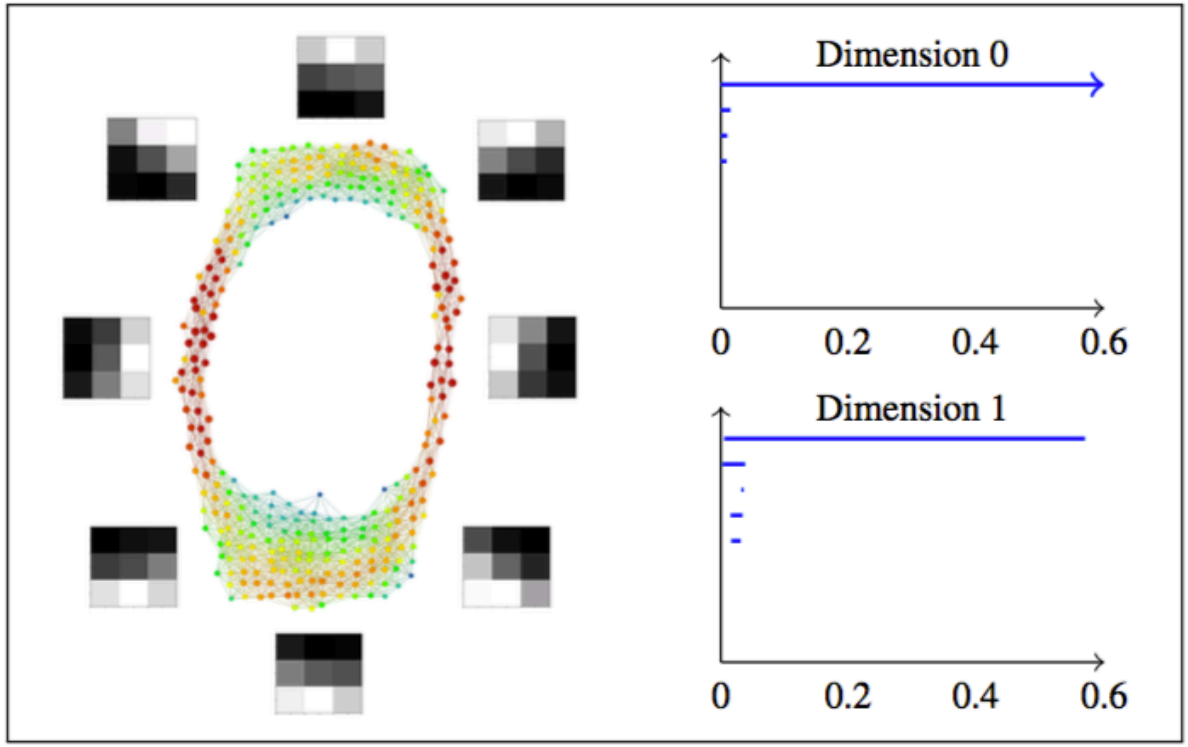
Creating Data Sets from CNN's

- Consider convolutional neural nets with 9 connections for each pixel
- Construct 9-vectors for each pixel
- Many grids in first and second layers
- Many experiments

Creating Data Sets from CNN's

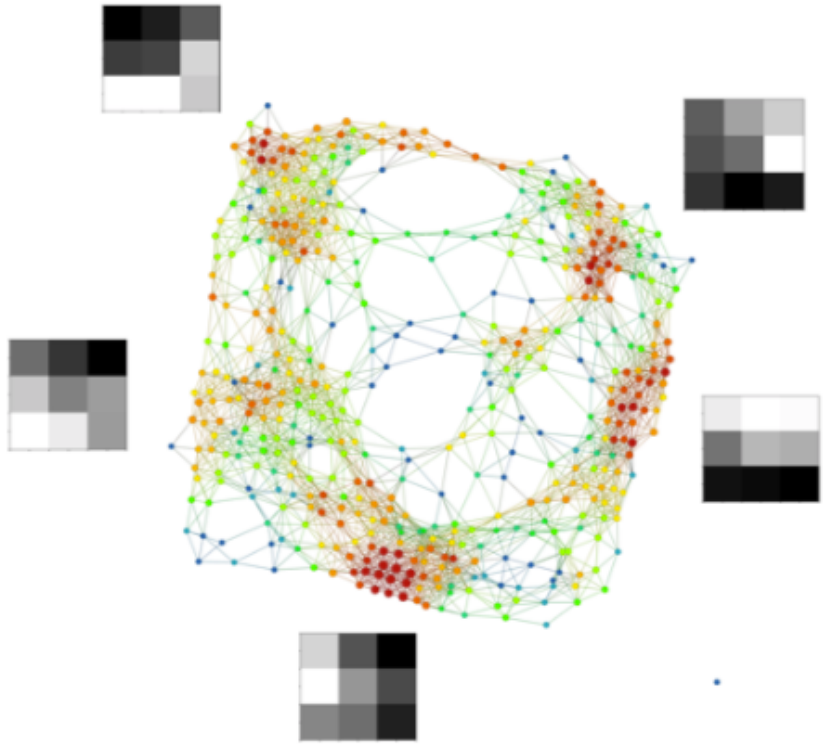
- Consider convolutional neural nets with 9 connections for each pixel
- Construct 9-vectors for each pixel
- Many grids in first and second layers
- Many experiments
- Apply same methodology as in Lecture 4

Findings: MNIST



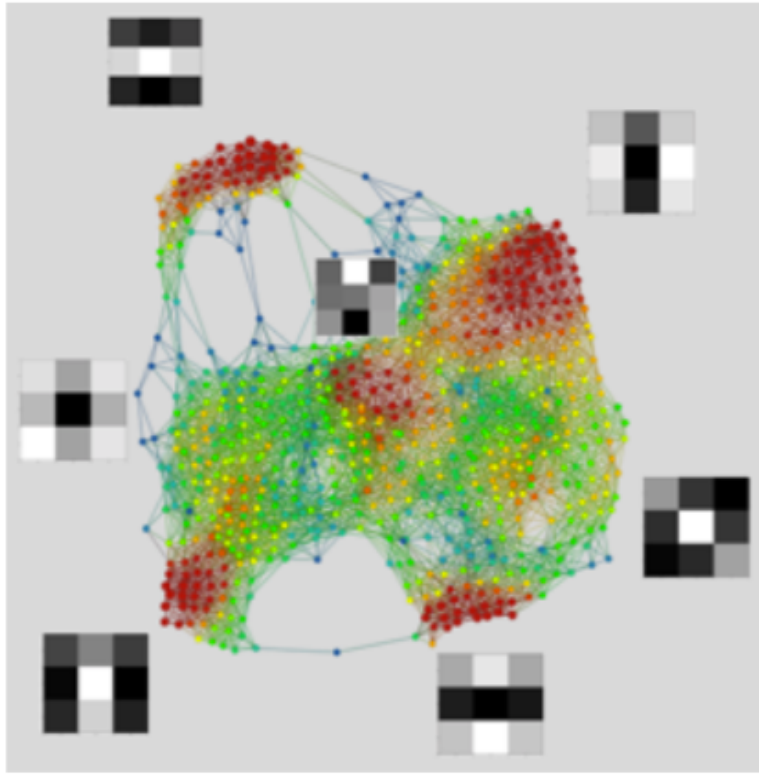
MNIST for layer 1 of a depth two net, non-localized estimator

Findings: MNIST



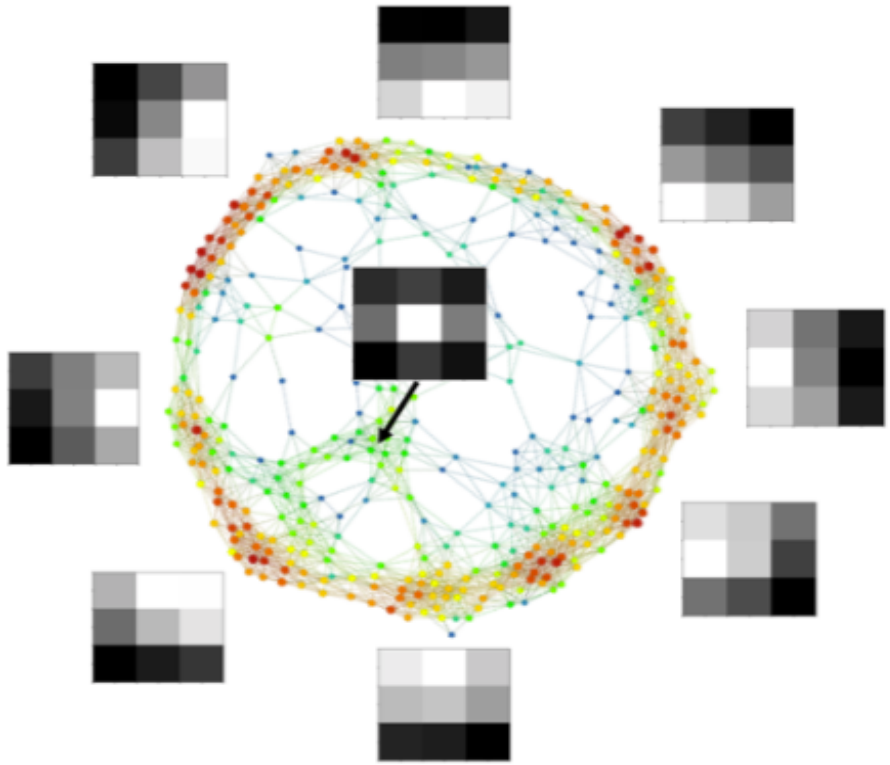
Mnist for layer 1 of a depth two net, localized estimator

Findings: Cifar10



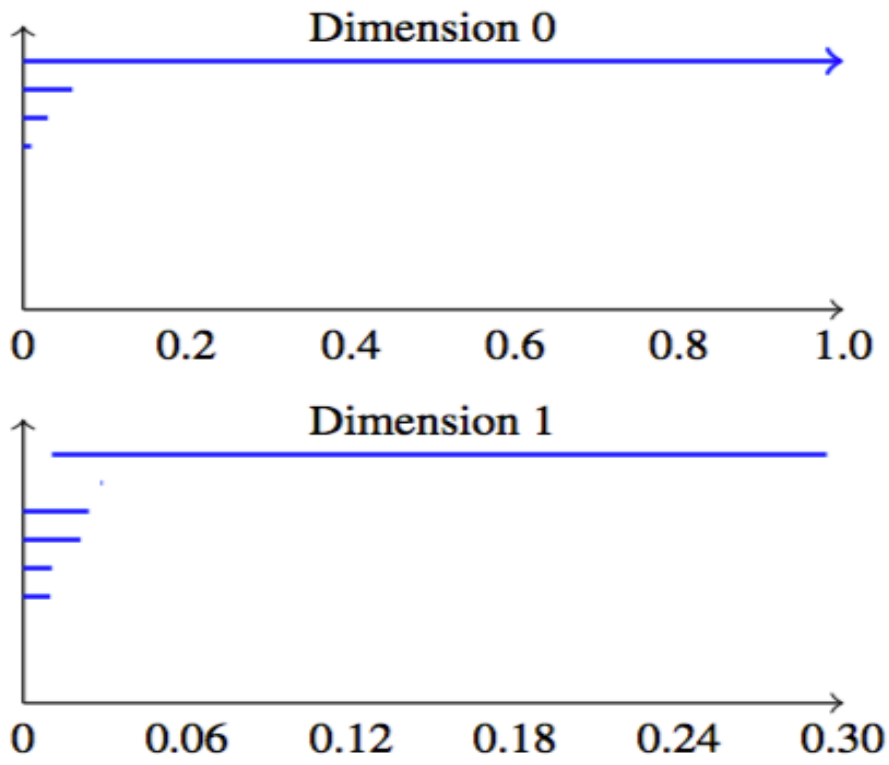
Layer 1 of depth two net, reduced to gray scale

Findings: Cifar10



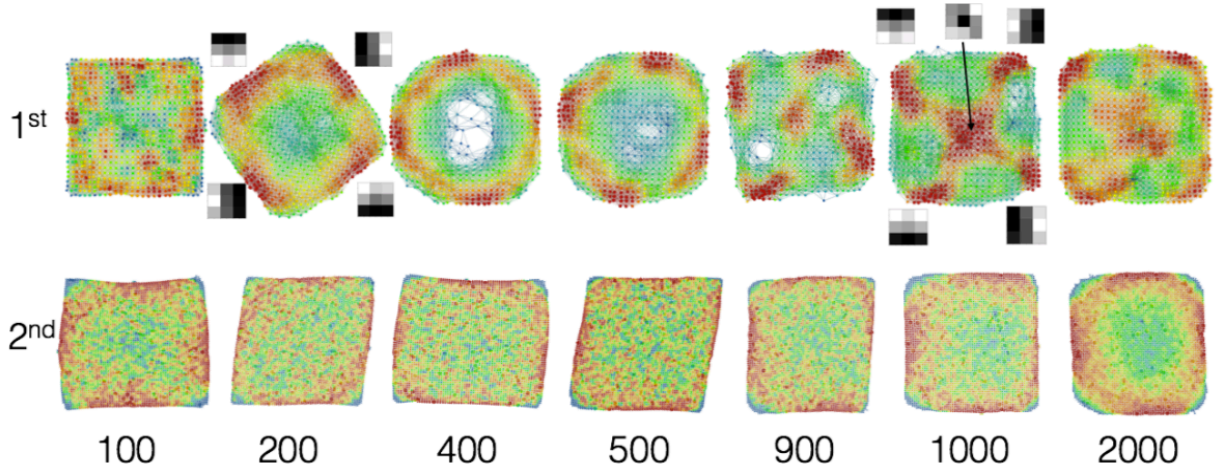
Second layer, reduced to gray scale

Findings: Cifar10



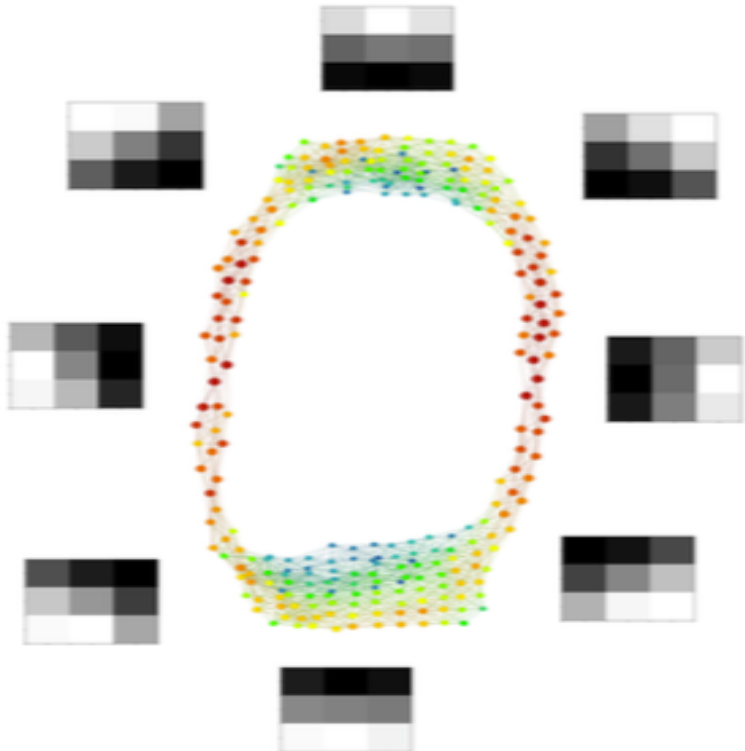
1D barcode for tightly thresholded data set of for 2nd layers , reduced to gray scale

Findings: Cifar10



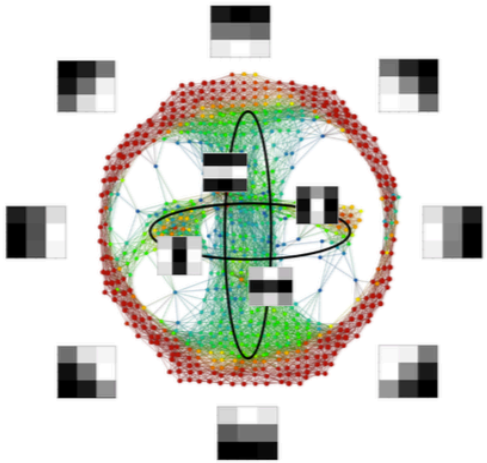
Mapper representations over the number of iterations, tightly density thresholded, gray scale reduced

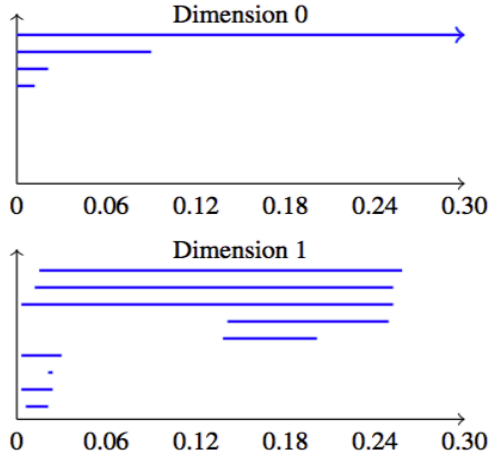
Findings: Cifar10



1st layer, Coarse density thresholding, color retained

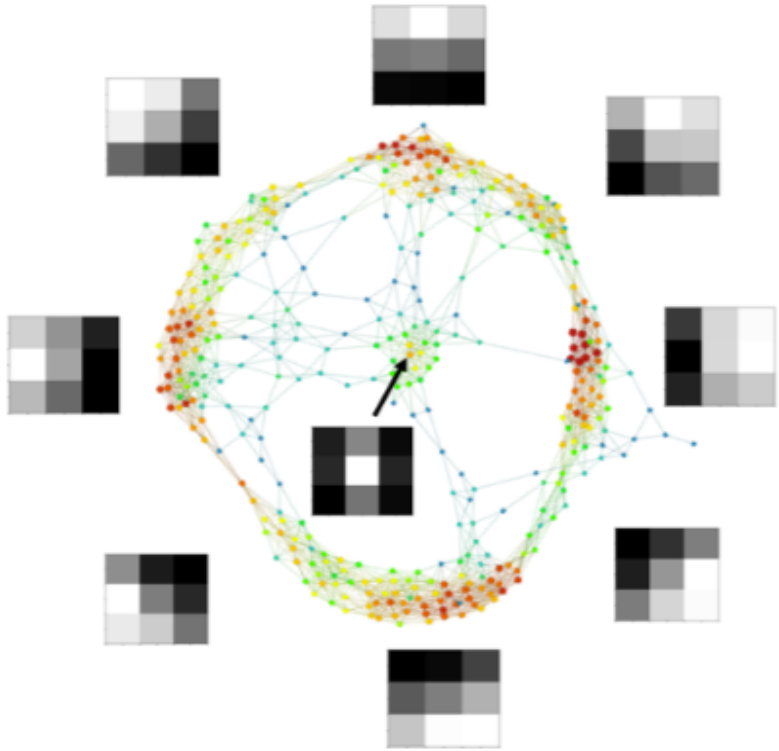
Findings: Cifar10





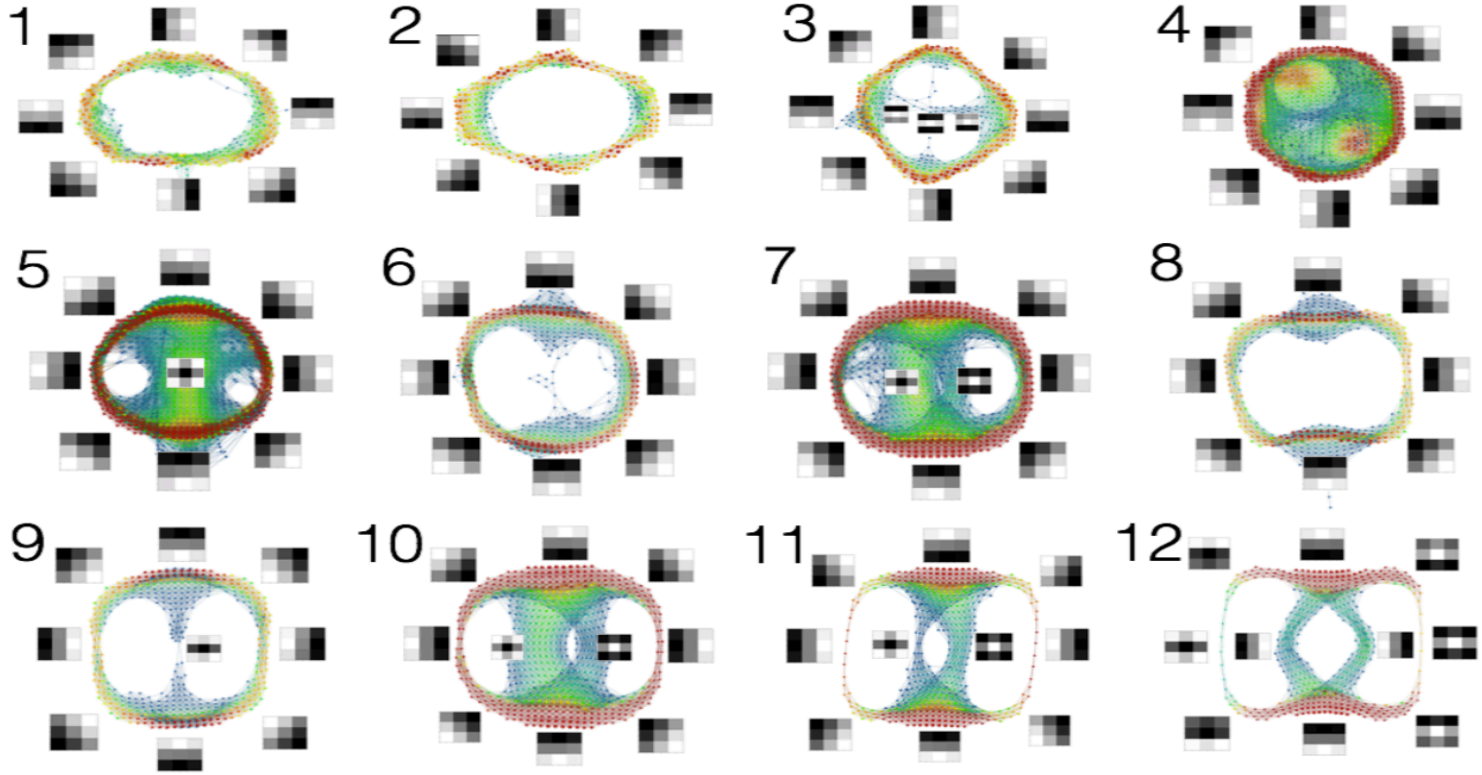
1st layer, looser density thresholding, more localized density estimator, color retained

Findings: Cifar10



2nd layer, fine density thresholding, color retained

Findings: VGG16



Mapper findings from each of 13 layers, same density thresholding, relatively local estimator

Experiments

- Create new features by forming dot products with patches modeled on primary circle

Experiments

- Create new features by forming dot products with patches modeled on primary circle
- Add these to existing “raw” features

Experiments

- Create new features by forming dot products with patches modeled on primary circle
- Add these to existing "raw" features
- Obtain a 3.5x speedup in training for SVHN data set

Experiments

- Create new features by forming dot products with patches modeled on primary circle
- Add these to existing "raw" features
- Obtain a 3.5x speedup in training for SVHN data set
- Obtain improved generalization for model computed on MNIST applied to SVHN

Building Architectures

- Key ingredient in constructing convolutional neural nets is the geometry of the feature space, i.e. the grid of pixels

Building Architectures

- Key ingredient in constructing convolutional neural nets is the geometry of the feature space, i.e. the grid of pixels
- Are there other geometries that might be useful?

Building Architectures

- Key ingredient in constructing convolutional neural nets is the geometry of the feature space, i.e. the grid of pixels
- Are there other geometries that might be useful?
- We can build geometries on feature spaces using the primary circle or Klein bottle

Building Architectures

- Key ingredient in constructing convolutional neural nets is the geometry of the feature space, i.e. the grid of pixels
- Are there other geometries that might be useful?
- We can build geometries on feature spaces using the primary circle or Klein bottle
- Mapper construction gives compressed geometric structures on the set of features of a general data set

Building Architectures: Feed Forward Systems

- Given two sets X and Y, a relation between X and Y is a subset $\mathcal{R} \subseteq X \times Y$

Building Architectures: Feed Forward Systems

- Given two sets X and Y, a relation between X and Y is a subset $\mathcal{R} \subseteq X \times Y$
- Can compose relations just like functions

Building Architectures: Feed Forward Systems

- Given two sets X and Y, a relation between X and Y is a subset $\mathcal{R} \subseteq X \times Y$
- Can compose relations just like functions
- Graph of a function is an example

Building Architectures: Feed Forward Systems

- Given two sets X and Y, a relation between X and Y is a subset $\mathcal{R} \subseteq X \times Y$
- Can compose relations just like functions
- Graph of a function is an example
- There is a category $C^{\mathcal{R}}$ whose objects are sets and where the morphisms from X to Y are the relations in $X \times Y$.

Building Architectures: Feed Forward Systems

- Given two sets X and Y, a relation between X and Y is a subset $\mathcal{R} \subseteq X \times Y$
- Can compose relations just like functions
- Graph of a function is an example
- There is a category $C^{\mathcal{R}}$ whose objects are sets and where the morphisms from X to Y are the relations in $X \times Y$.
- An *abstract feed forward system of depth n* is a functor from the category

$$\underline{n} = 1 \longrightarrow 2 \longrightarrow \cdots \qquad \cdots \longrightarrow n$$

to $c^{\mathcal{R}}$

Building Architectures: Construct Computation Graph from Feed Forward Systems

- For an abstract feed forward system $F : \underline{n} \to C^{\mathcal{R}}$, we form a directed graph

Building Architectures: Construct Computation Graph from Feed Forward Systems

- For an abstract feed forward system $F : \underline{n} \to C^{\mathcal{R}}$, we form a directed graph
- The vertex set is $\coprod_{i=1}^{n} F(i)$

Building Architectures: Construct Computation Graph from Feed Forward Systems

- For an abstract feed forward system $F : \underline{n} \to C^{\mathcal{R}}$, we form a directed graph
- The vertex set is $\coprod_{i=1}^{n} F(i)$
- For vertices $v \in F(i)$ and $w \in F(i+1)$, there is an edge from v to w if and only if (v, w) is in the relation $F(i \to i+1)$.

Building Architectures: Construct Computation Graph from Feed Forward Systems

- For an abstract feed forward system $F : \underline{n} \to C^{\mathcal{R}}$, we form a directed graph
- The vertex set is $\coprod_{i=1}^{n} F(i)$
- For vertices $v \in F(i)$ and $w \in F(i+1)$, there is an edge from v to w if and only if (v, w) is in the relation $F(i \to i+1)$.
- No edges from v to w unless there is an i so that $v \in F(i)$ and $w \in F(i+1)$.

Building Architectures: Important Relations

- If Γ is a graph, then we have a relation $R_\Gamma : V_\Gamma \to V_\Gamma$ defined by $(v, w) \in R_\Gamma$ if and only if (v, w) is an edge in Γ.

Building Architectures: Important Relations

- If Γ is a graph, then we have a relation $R_\Gamma : V_\Gamma \to V_\Gamma$ defined by $(v, w) \in R_\Gamma$ if and only if (v, w) is an edge in Γ.
- If (X, d) is a metric space and $r > 0$ is a threshold, then $R_d(r) : X \to X$ is the relation given by $(x, x') \in R_d(r)$ if and only if $d(x, x') \leq r$.

Building Architectures: Important Relations

- If Γ is a graph, then we have a relation $R_\Gamma : V_\Gamma \to V_\Gamma$ defined by $(v, w) \in R_\Gamma$ if and only if (v, w) is an edge in Γ.
- If (X, d) is a metric space and $r > 0$ is a threshold, then $R_d(r) : X \to X$ is the relation given by $(x, x') \in R_d(r)$ if and only if $d(x, x') \leq r$.
- If $f : X \to Y$ is a function, then the graph $\{(x, f(x)|x \in X$ is a relation from X to Y

Building Architectures: Important Relations

- If Γ is a graph, then we have a relation $R_\Gamma : V_\Gamma \to V_\Gamma$ defined by $(v, w) \in R_\Gamma$ if and only if (v, w) is an edge in Γ.
- If (X, d) is a metric space and $r > 0$ is a threshold, then $R_d(r) : X \to X$ is the relation given by $(x, x') \in R_d(r)$ if and only if $d(x, x') \leq r$.
- If $f : X \to Y$ is a function, then the graph $\{(x, f(x)|x \in X$ is a relation from X to Y
- If Γ_1 and Γ_2 are two mapper models of the same data set, then there is a naturally defined relation $R(\Gamma_1, \Gamma_2)$ from V_{Γ_1} to V_{Γ_2} given by $(v, w) \in R(\Gamma_1, \Gamma_2)$ if and only if he collections corresponding to v and w have a data point in common

Buillding Architectures

- The idea is to use “geometry” on the feature space to construct neural network architectures

Buillding Architectures

- The idea is to use "geometry" on the feature space to construct neural network architectures
- Geometry can arise in numerous ways

Buillding Architectures

- The idea is to use "geometry" on the feature space to construct neural network architectures
- Geometry can arise in numerous ways
- A priori geometry: grid in case of image convolutional networks, linear arrays for time series and text

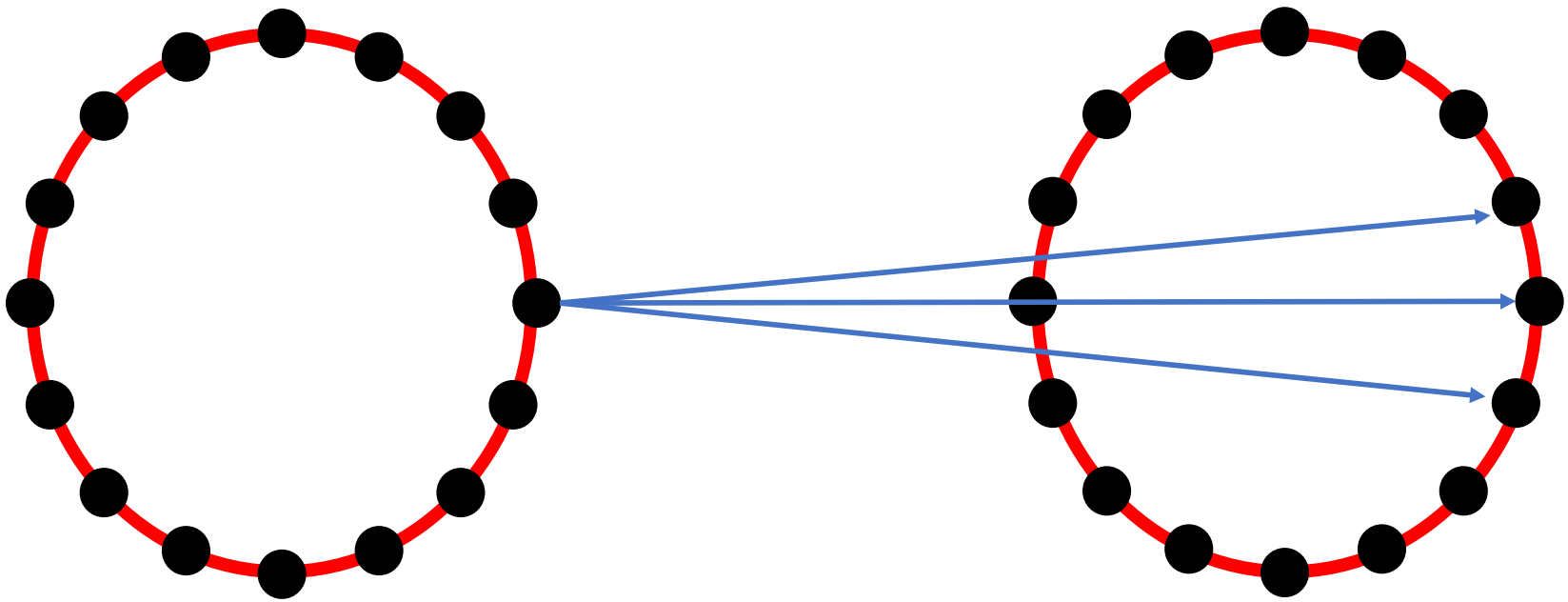
Buillding Architectures

- The idea is to use "geometry" on the feature space to construct neural network architectures
- Geometry can arise in numerous ways
- A priori geometry: grid in case of image convolutional networks, linear arrays for time series and text
- Geometries found by research. E.g. primary circle or Klein bottle in the case of natural images

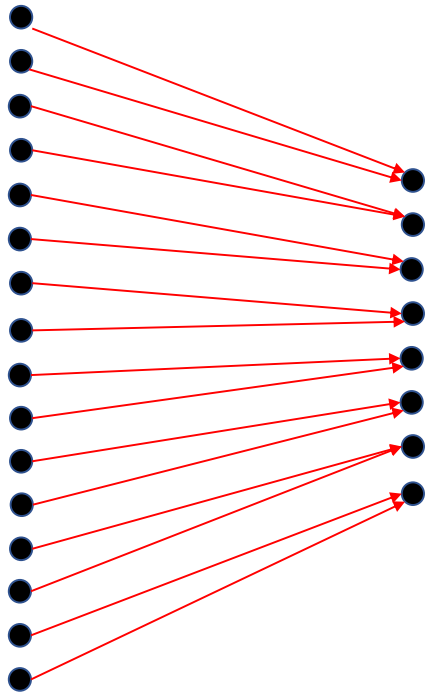
Buillding Architectures

- The idea is to use "geometry" on the feature space to construct neural network architectures
- Geometry can arise in numerous ways
- A priori geometry: grid in case of image convolutional networks, linear arrays for time series and text
- Geometries found by research. E.g. primary circle or Klein bottle in the case of natural images
- Bespoke geometries obtained by taking mapper models for the column spaces of data matrices

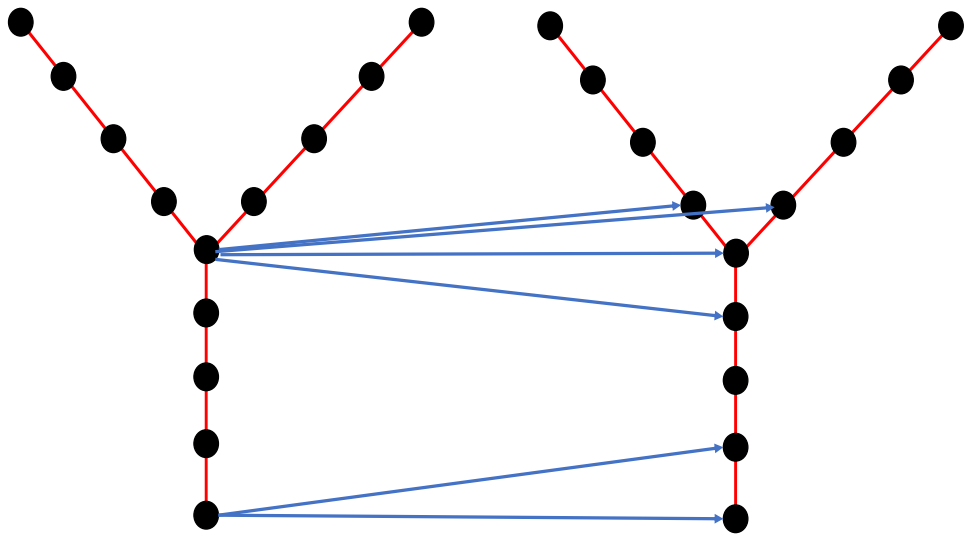
Building Architectures



Building Architectures



Building Architectures



Building Architectures

- Can use symmetries to construct other convolutional structures

Building Architectures

- Can use symmetries to construct other convolutional structures
- Example: rotations by multiplication by roots of unity on discretization on circle

Building Architectures

- Can use symmetries to construct other convolutional structures
- Example: rotations by multiplication by roots of unity on discretization on circle
- Also exist analogues of pooling constructions for circles and Klein bottles

Building Architectures

- Can use symmetries to construct other convolutional structures
- Example: rotations by multiplication by roots of unity on discretization on circle
- Also exist analogues of pooling constructions for circles and Klein bottles
- Sphere occurs in 3D voxelized images. Icosahedron is a symmetric discretization