Introduction to Machine Learning (IML)

LECTURE #13: OTHER LEARNING SETTINGS

236756 – 2024 SPRING – TECHNION

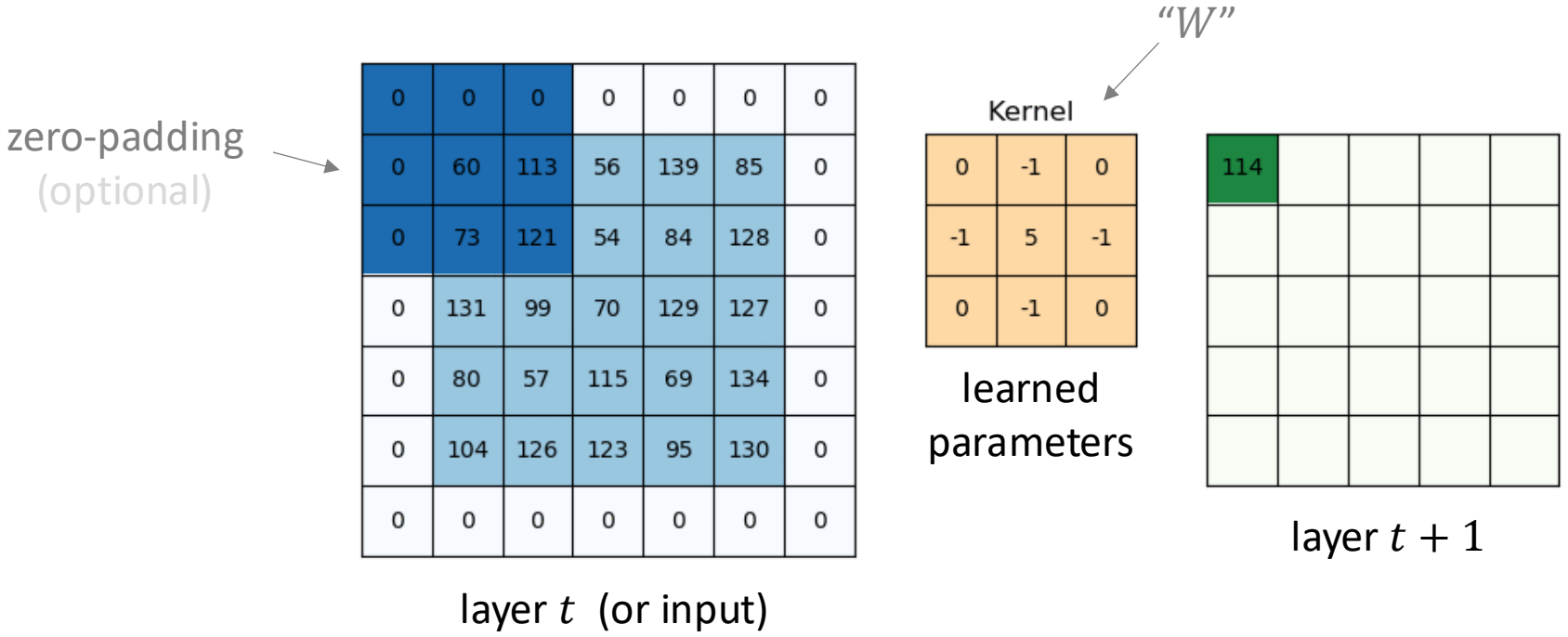
LECTURER: NIR ROSENFELD

Today

- **Finish up deep learning**
- **Part IV**: *beyond supervised learning*
  1. ~~Unsupervised learning: dimensionality reduction~~
  2. Other learning settings + course summary (today)

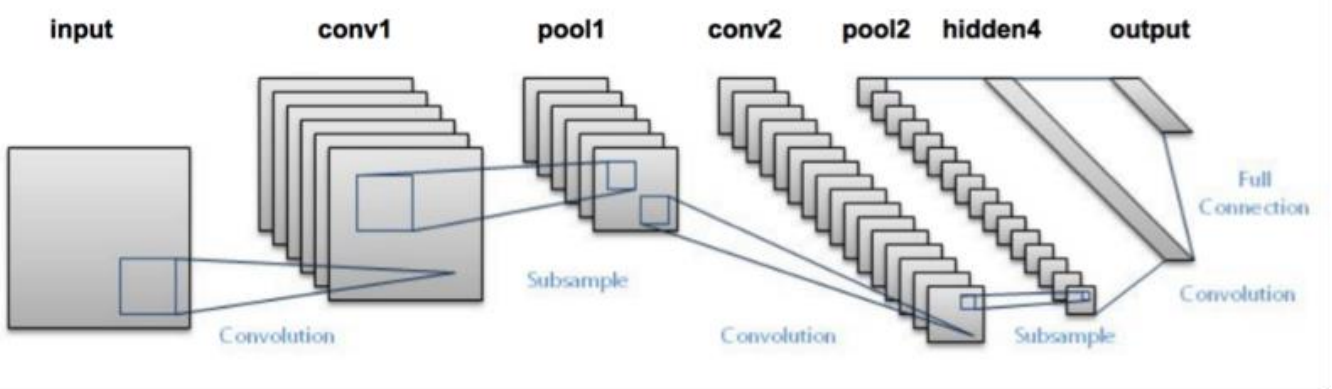
Convolutional Neural Networks

- **Idea**: apply same local "convolution" filter to all "patches"
- **Step 1**: discard connection – keep only local information
- **Step 2**: tie/share weights across neurons in same layer
- **Result**: layers act as convolution operator that is invariant to translations



Convolutional Neural Networks

- **Idea**: apply same local “convolution” filter to all “patches”
- **Step 1**: discard connection – keep only local information
- **Step 2**: tie/share weights across neurons in same layer
- **Result**: layers act as convolution operator that is invariant to translations
- Typically have multiple kernels (or “filters”) per layer; adds a “third” dimension:

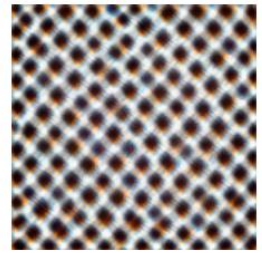
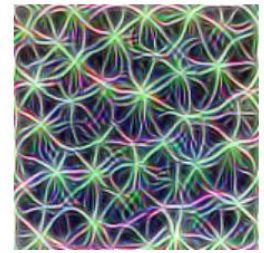
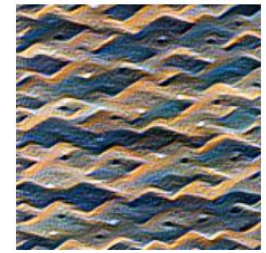


- (Usually, at increasing depth, height/width decreases, but num. filters increases)

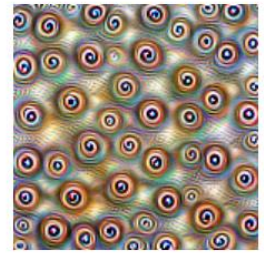
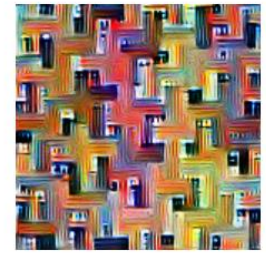
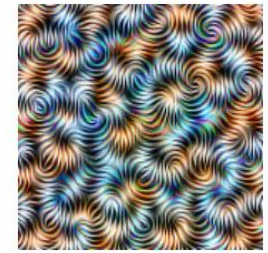
Convolutional Neural Networks

- **CNNs work well on natural images because they:**
  - ➢ use a fraction of possible connections (by applying local filters)
  - ➢ drastically reduce the number of learnable parameters (by weight sharing/tying)
  - ➢ do so in a way that preserves real-world input structure and invariance
- **Thus, CNNs can:**
  - enjoy the benefits of deep models (multiple layers, non-linearities, etc)
  - but using a manageable number of parameters
- Some studies suggest that different filters at different depths have different meanings!

Layer 3a

Layer 3b

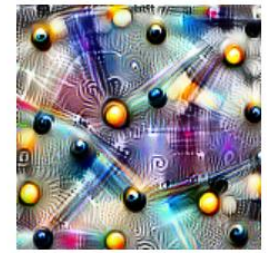





Layer 4b

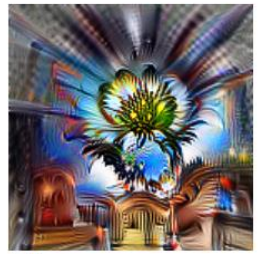

Architecture


Fluffy rope


Trees


Billiard balls

Layer 4c


Palm trees


Wheels


Dogs on leash


Houses

Layer 5a


Candles

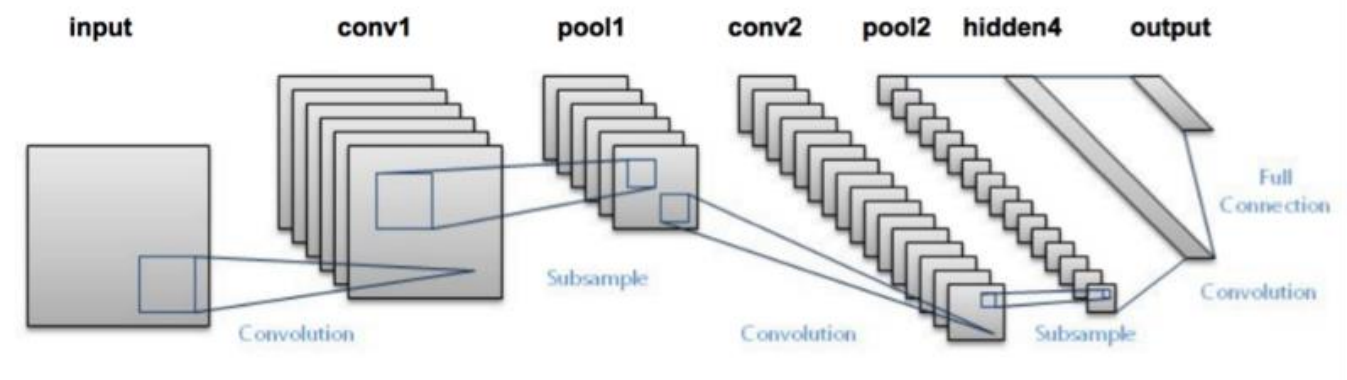

Balls


Brass instruments

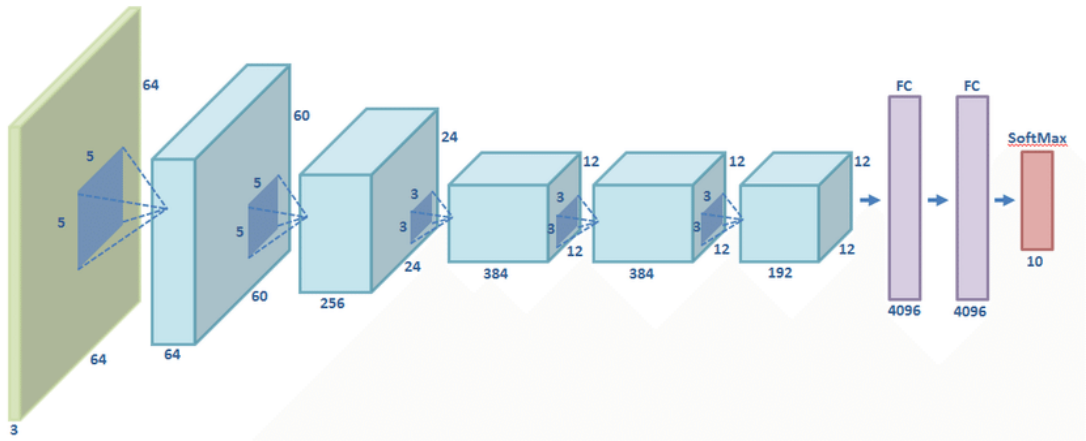

Traffic lights

https://distill.pub/2017/feature-visualization/appendix/

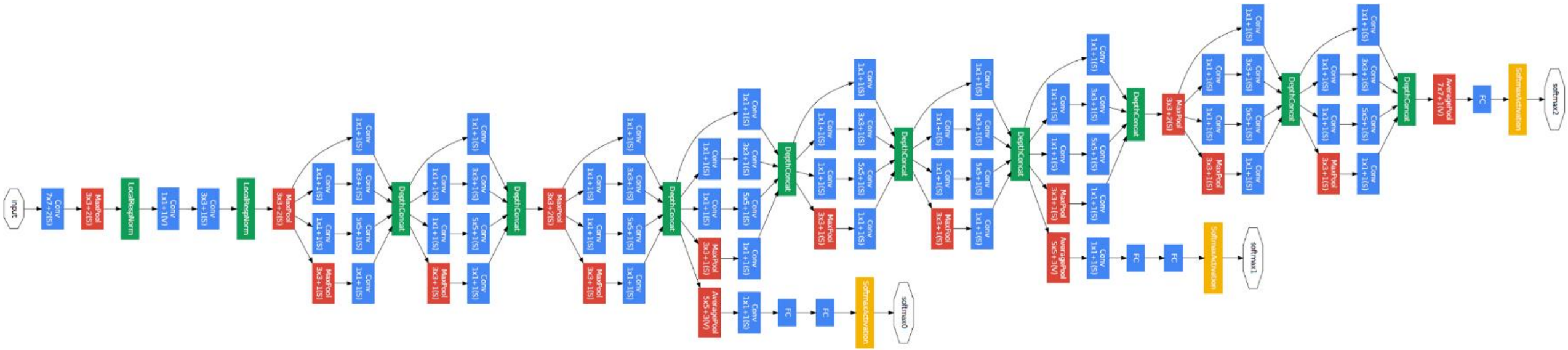
Size matters


input
conv1
pool1
conv2
pool2
hidden4
output
Full
Connection
Convolution
Subsample
Convolution
Convolution
Subsample

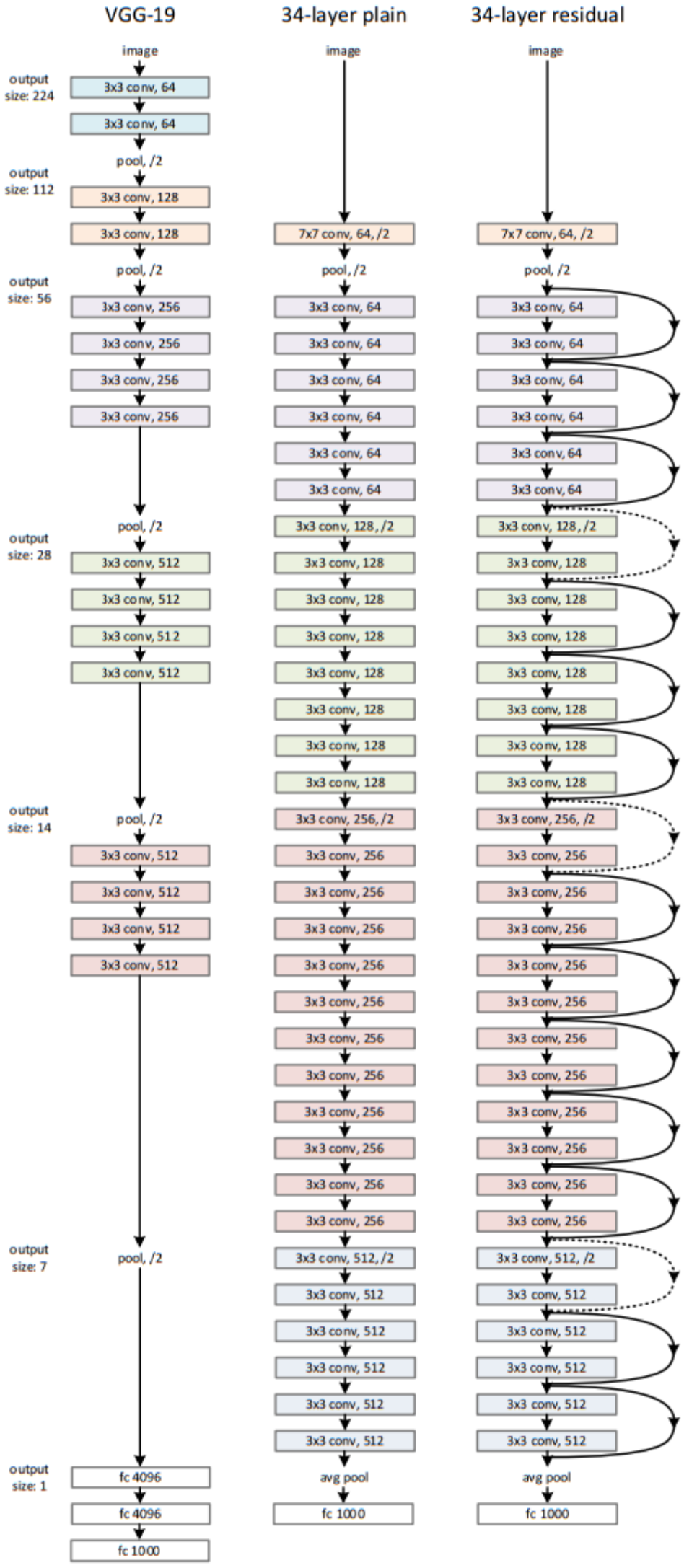
LeNet-5 (1989)

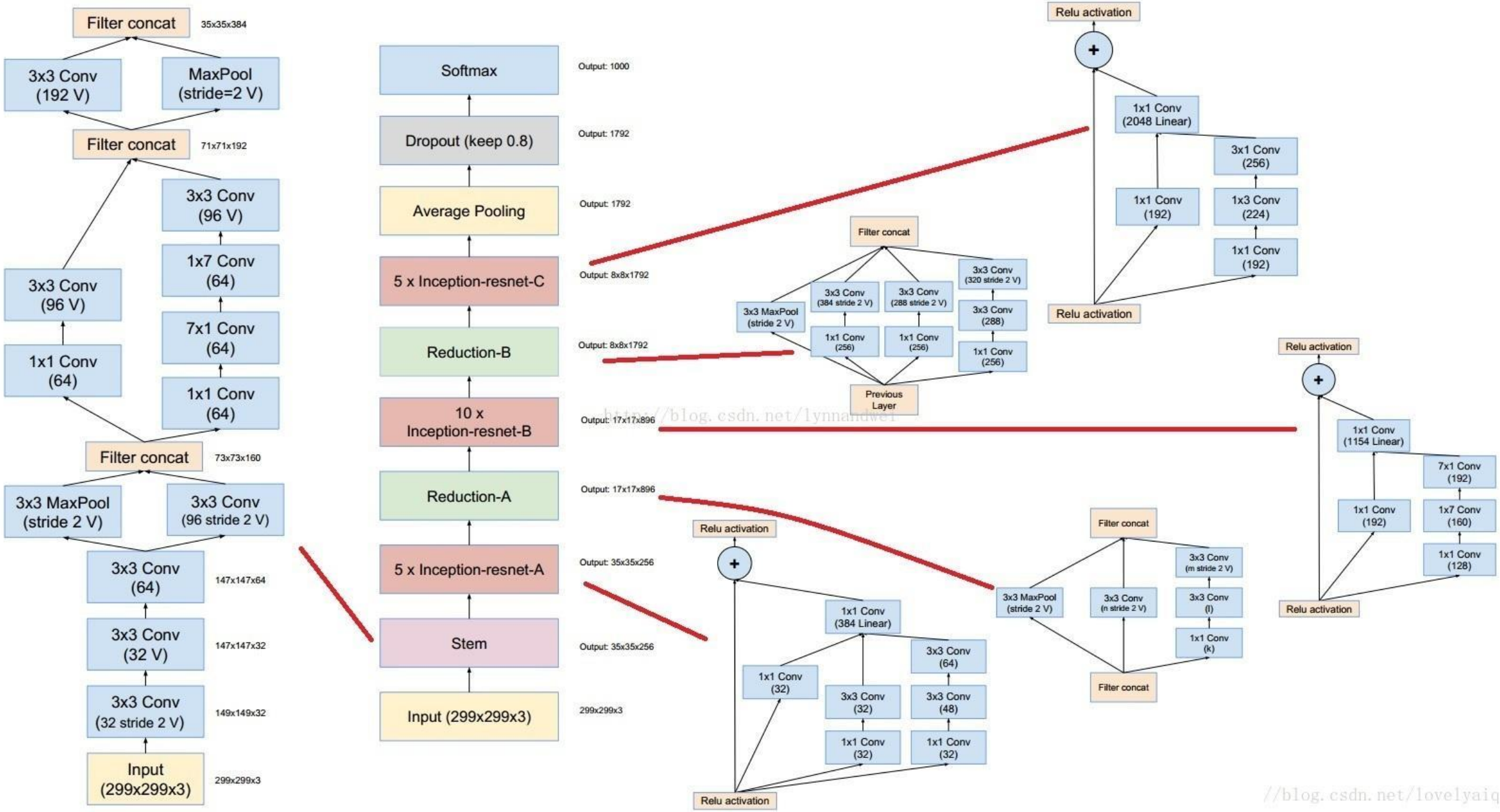


AlexNet (2012)



GoogLeNet (2012)


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
7x7 conv, 64, /2
pool, /2
3x3 conv, 128, /2
3x3 conv, 256, /2
3x3 conv, 512, /2
avg pool
fc 1000
VGG-19 / ResNet (2014)



Generalization (in theory)

- Modern neural networks are **big**. Should we expect them to generalize?
- **Theorem**: (won't prove)
  Let $\mathcal{H}_{NN}$ be class of sigmoidal-activation neural networks with N units and M connections (i.e., parameterized weights), then:

$$M^2 \leq VC(\mathcal{H}_{NN}) \leq N^2M^2$$

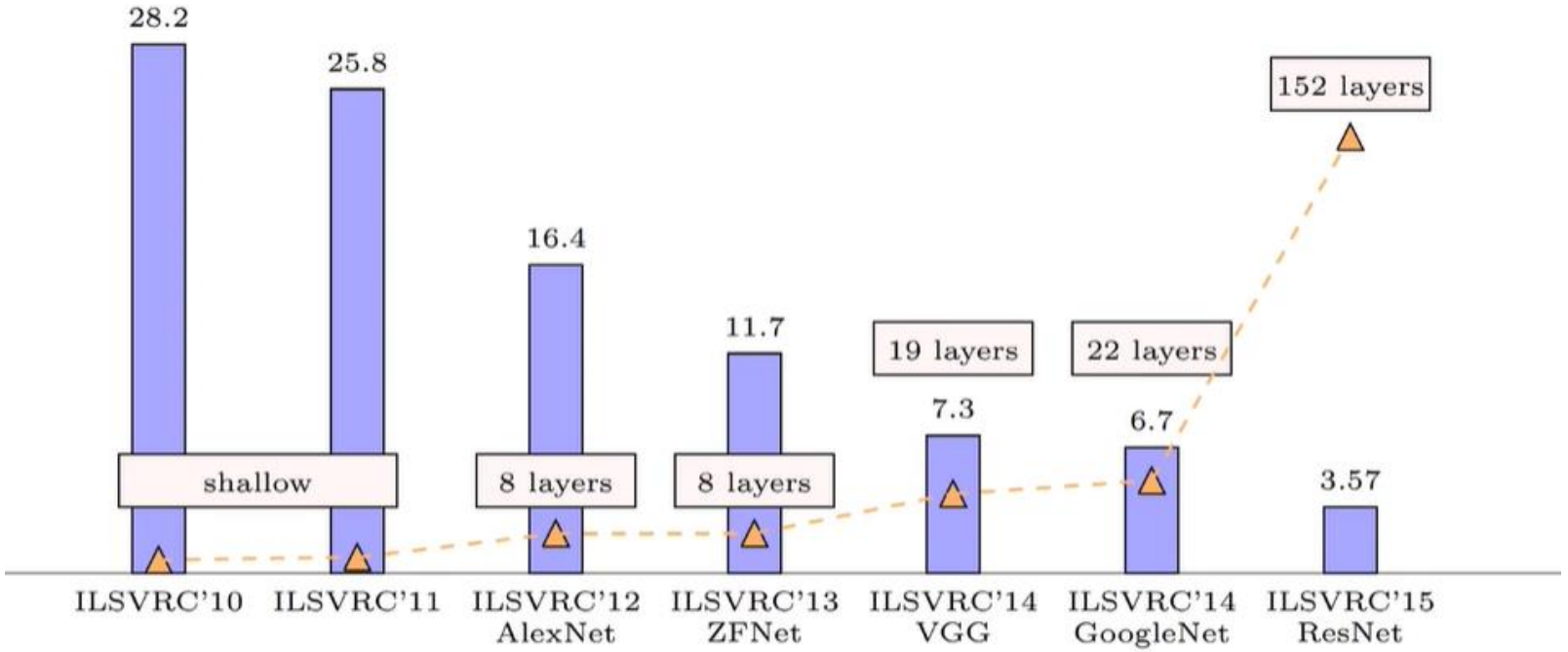
- VC is large, and grows quickly with network size
- (Interestingly, the bound is agnostic to width, depth, or connectivity structure; only depends on num. units and parameters!)
- Need to carefully control for overfitting

Controlling overfitting

- Can use explicit regularization (e.g., norm on weights), but not a very popular choice
- Common *implicit* regularization tricks:
  1. **Early stopping** (using held-out validation set; like we saw)
  2. **Drop-out:**
     at each **epoch** (=SGD step over all data points),
     temporarily **"remove"** a random subset of connections (=parameters)
     (typical drop-out rates (=% discarded connections) are 10%-50% - large!)
- **Intuition for drop-out**:
  - Prevents network from relying extensively on specific weights
  - Thus, prevents network from being tailored to training data (sort of…)
  - Can be thought of as **implicitly training an ensemble of network models**, each corresponding to a partial connectivity pattern (sub-graph)

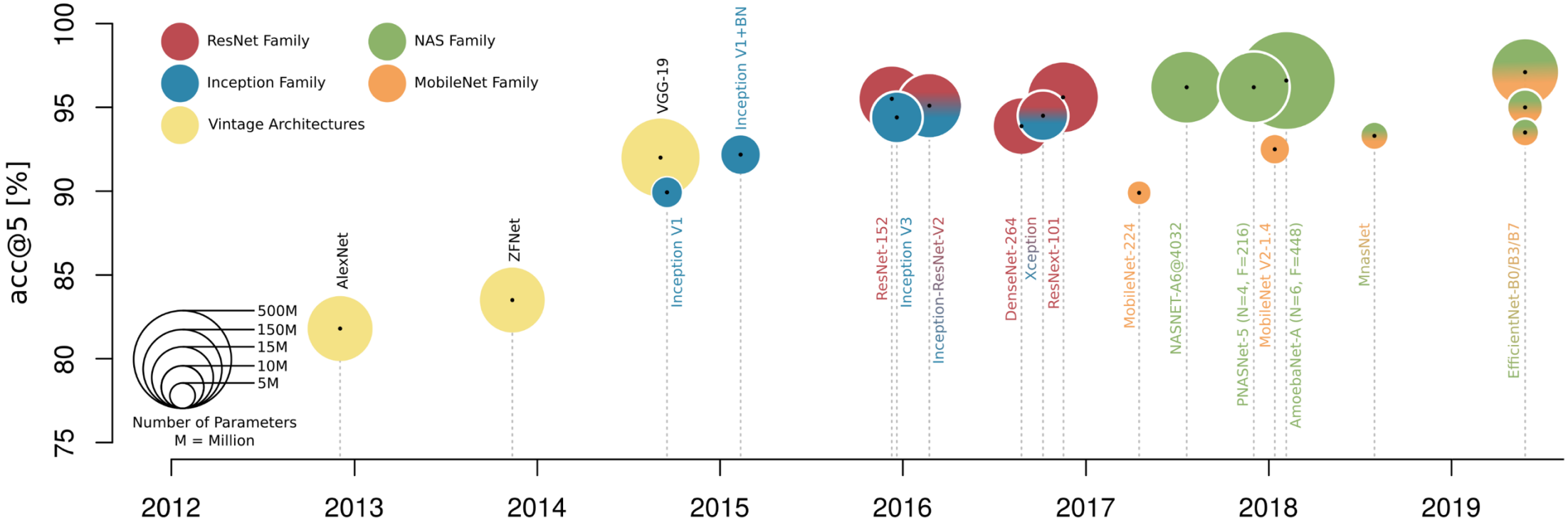
Generalization (in practice)

- VC says bad generalization, but empirically, even huge networks don't always overfit (this likely goes far beyond any effect attributed to explicit regularization)



Generalization (in practice)

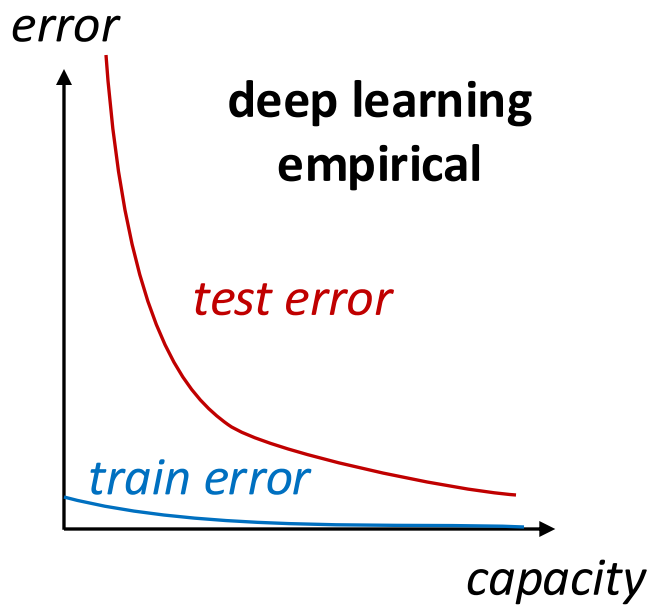
- VC says bad generalization, but empirically, even huge networks don't always overfit (this likely goes far beyond any effect attributed to explicit regularization)



- Surprising from the standpoint of classic statistical learning theory!

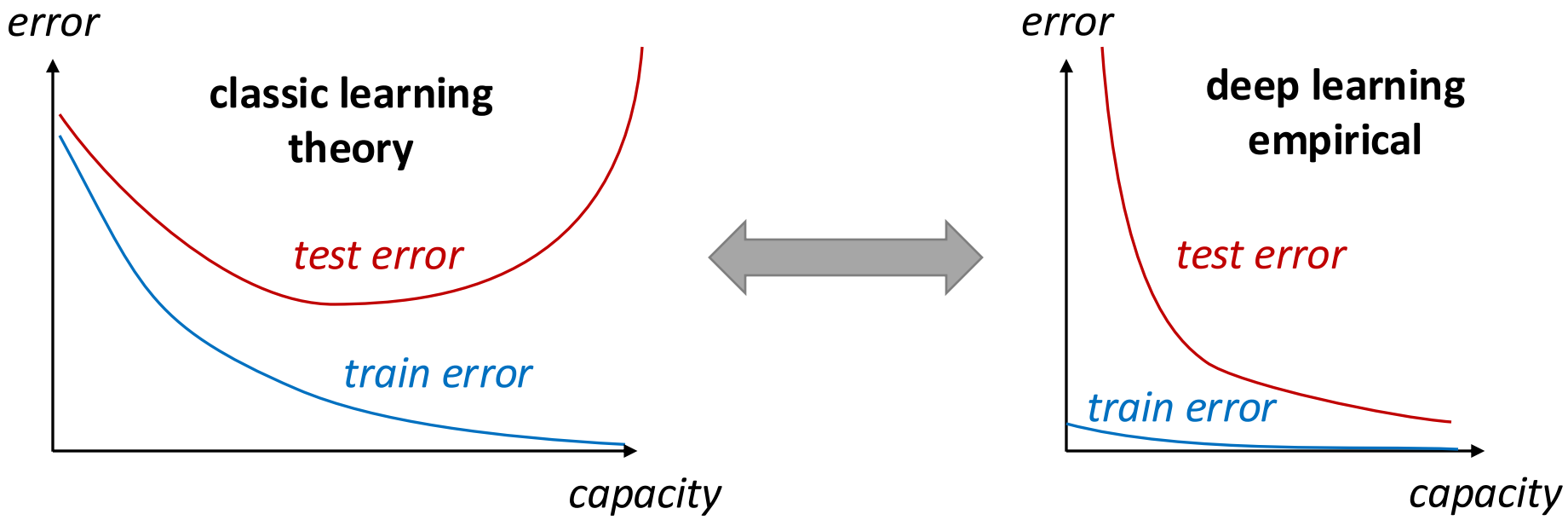
Theory! (?)

- Some research argues that the classic theory just doesn't apply (overstatement, but true)
- **Empirical observation:** lower train error entails lower test error – even for tiny train errors!



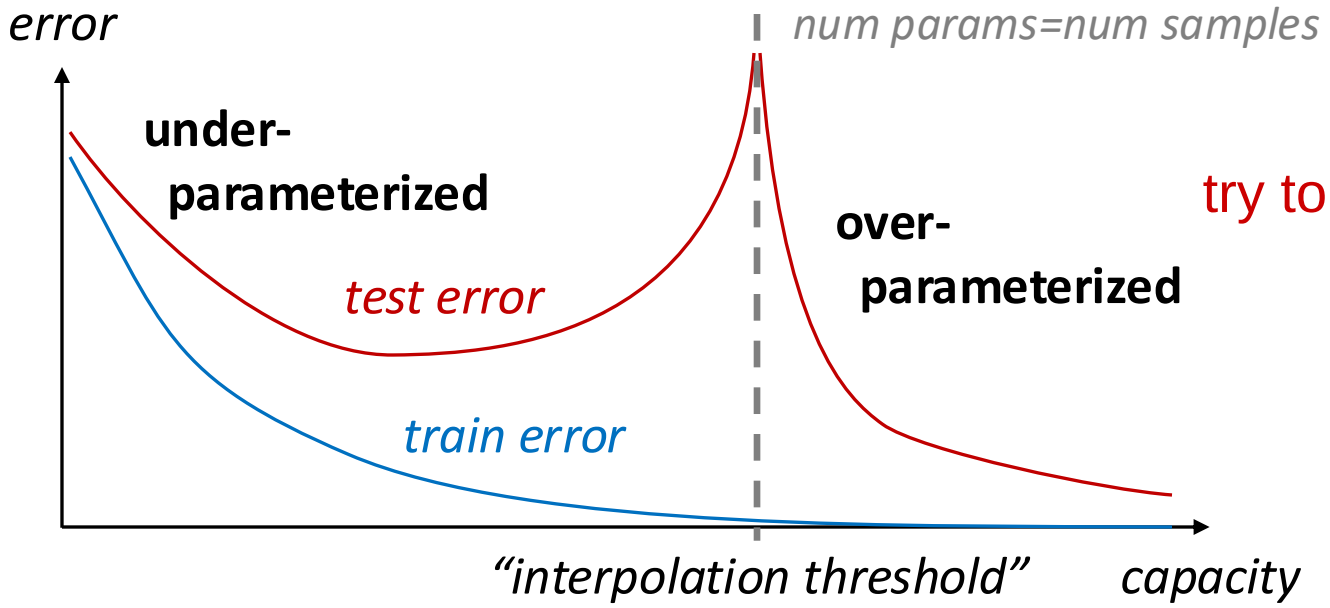
Theory! (?)

- Some research argues that the classic theory just doesn't apply (overstatement, but true)
- **Empirical observation:** lower train error entails lower test error – even for tiny train errors!
- Does not align with classic bias-variance perspective



Theory! (?)

- Some research argues that the classic theory just doesn't apply (overstatement, but true)
- **Empirical observation:** lower train error entails lower test error – even for tiny train errors!
- Does not align with classic bias-variance perspective
- Reconciled by "double descent" theory: (or is it?)

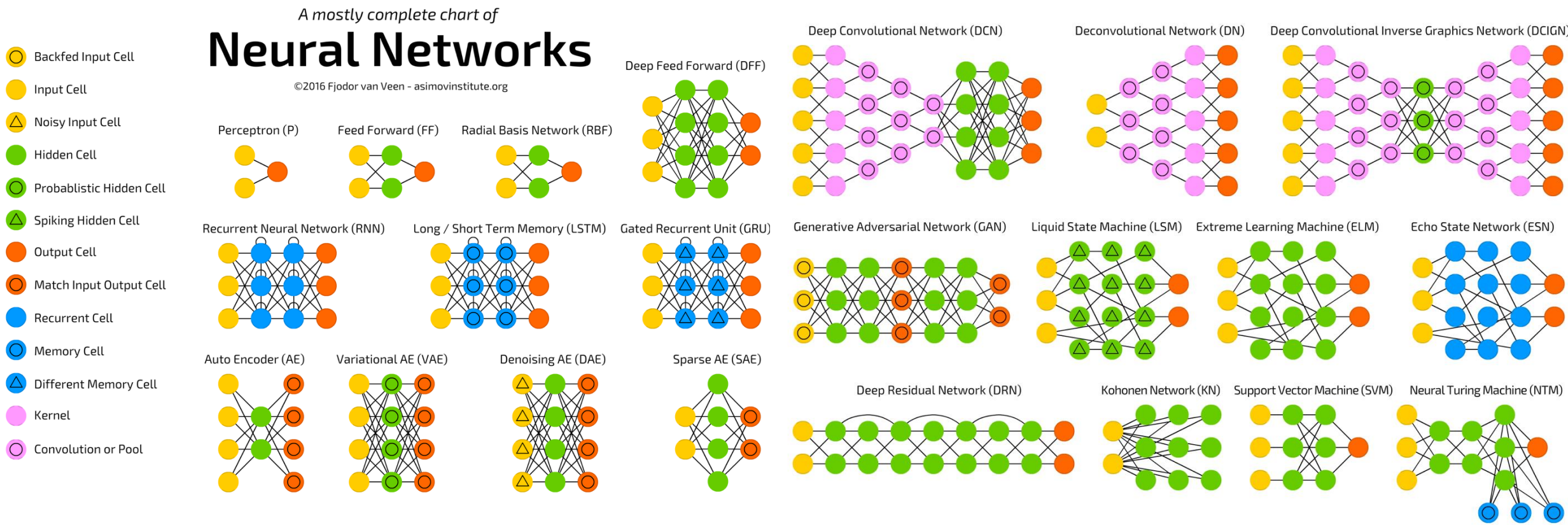


try to understand this better later

Discussion

Discussion

- Deep learning is an old idea that took time to become practically useful
- **Reasons why it works well now** (on some tasks):
  - compute power (GPU)
  - more machines (parallel)
  - tons of data (linear scaling)
  - algorithms (backprop)
  - autodiff
  - modeling flexibility (autodiff)
  - compositional flexibility (building blocks)
  - socially acceptable
  - **file-drawer effect**
- So “new electricity” or “fancy logistic regression”?
- **Suggestion**: just make the best of it!



and many more…
will see some in tirgul!

Beyond supervised learning

Other learning settings

What lies beyond?

- So far we've studied ML mostly from a very particular point of view
- But there is much more to it!

- Today we'll give a broad survey of some **extended learning settings**
- **Main theme** – extend by considering ***structure*:**
  1. **beyond vectors**: structure in *input space*
  2. **beyond binary/scalar**: structure in *output space*
  3. **beyond iid prediction**: structure in *learning*
- **Fair warning**: no longer "textbook material", so not representative (and biased towards what I know)

Structure in input space

Beyond vectors

Vectors and representations

- So far we've mostly focused on input *feature vectors,* $x \in \mathcal{X} = \mathbb{R}^d$
- These are abstract objects – useful for discussing algorithms, optimization, and theory
- But in real problems, inputs have *meaning* – a form of *prior knowledge*

Vectors and representations

- So far we've mostly focused on input *feature vectors,* $x \in \mathcal{X} = \mathbb{R}^d$
- These are abstract objects – useful for discussing algorithms, optimization, and theory
- But in real problems, inputs have *meaning* – a form of *prior knowledge*
- We will use the term "**structure**" to describe knowledge which we can utilize
- Incorporating structure (into model, loss, regularization, etc) is called **inductive bias**

- Most approaches we saw utilize structure via *representation mappings* $\phi(x)$
- Examples of structure-encoding representations we've seen:
  - **domain structure**: hand-coded features
  - **similarity structure**: kernels (increase model capacity)
  - ~~**geometric structure**: manifolds~~
  - **invariance structure**: eg CNNs for vision (reduce model capacity)

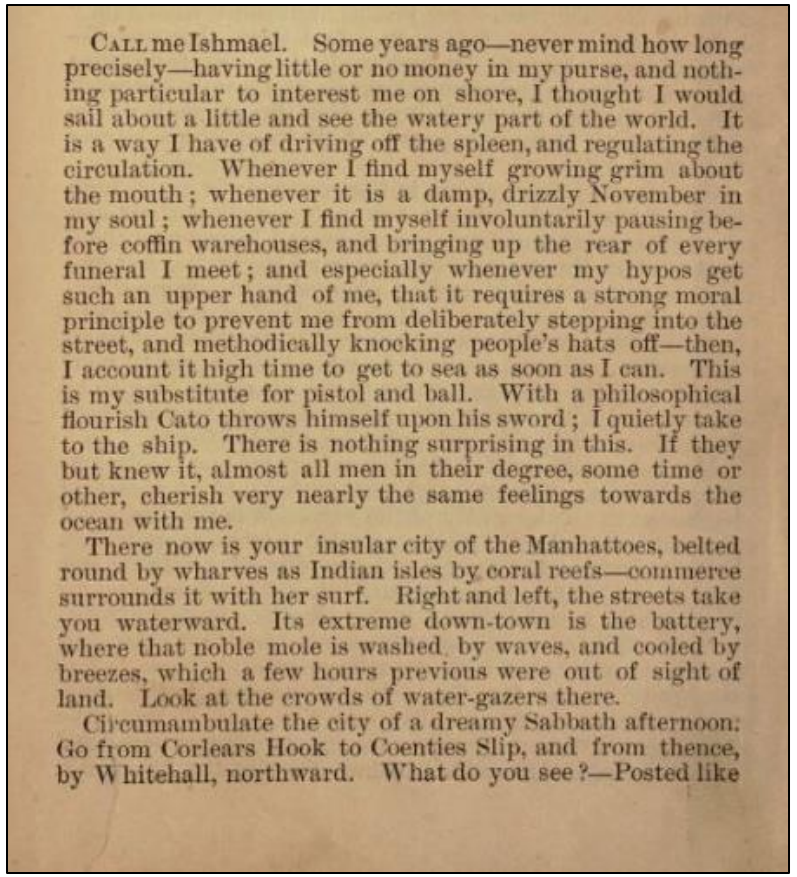
Beyond vectors

- In all the examples we've discussed, ϕ are vector representations of… *vectors*
- But not all objects can naturally be described by vectors
- For general objects, we need general models: $h(\text{object})$
- **Q**: How should we design such models?
- **A**: Common approach – linear-in-representation models: $h_w(\text{object}) = w^\top \phi(\text{object})$
- Only difference: ϕ is now a mapping from arbitrary objects, rather than from vectors
- Allows us to use most of the tools we've seen (kernels, boosting, feed-forward networks, etc.)
- Modeling challenge lies in designing ϕ that utilize structure efficiently
- Can use prior knowledge to either **engineer** a fixed ϕ or **learn** a parameterized ϕ_θ
- **Let's see some examples**

Beyond vectors

- As an example, let's consider **text**

$x =$



CALL me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

There now is your insular city of the Manhattoes, belted round by wharves as Indian isles by coral reefs—commerce surrounds it with her surf. Right and left, the streets take you waterward. Its extreme down-town is the battery, where that noble mole is washed by waves, and cooled by breezes, which a few hours previous were out of sight of land. Look at the crowds of water-gazers there.

Circumambulate the city of a dreamy Sabbath afternoon: Go from Corlears Hook to Coenties Slip, and from thence, by Whitehall, northward. What do you see?—Posted like

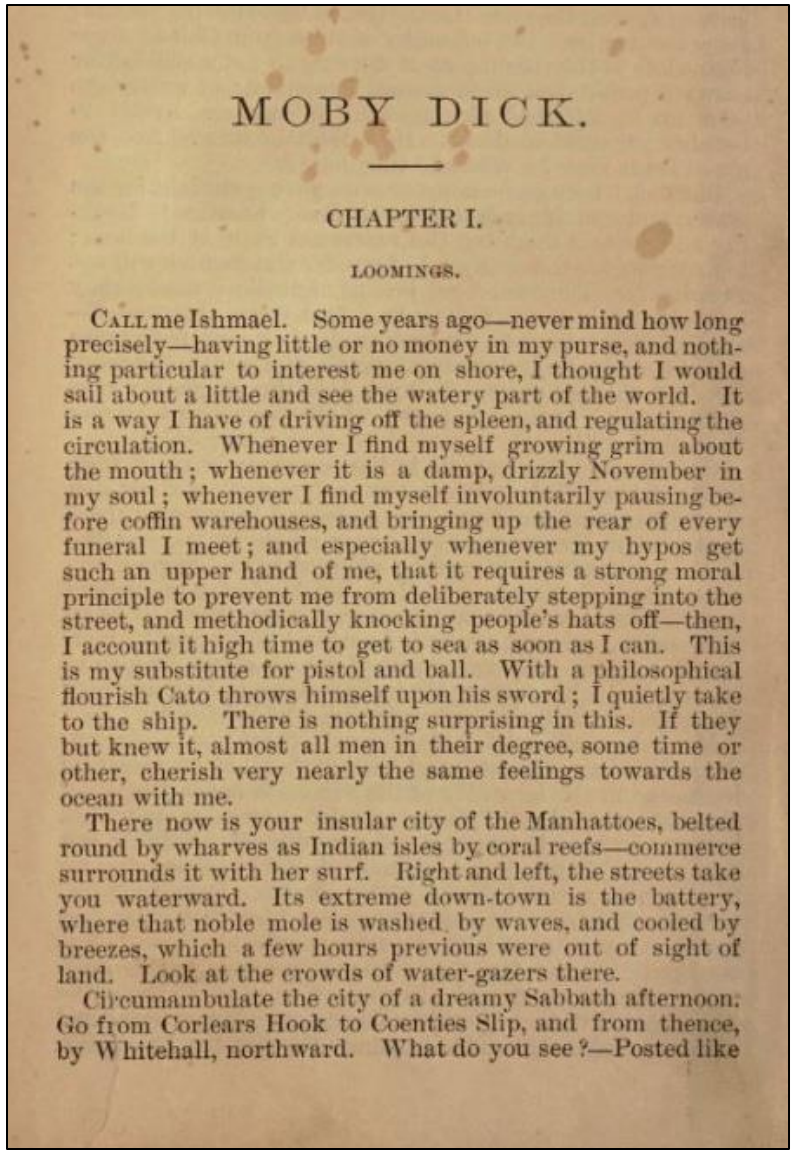
Beyond vectors

- As an example, let's consider **text** →
- **Native representation**: sequence of characters $x =$

$$x \in \mathcal{X} = \bigcup_{k=1}^{\infty} \Sigma^k$$

where Σ is some finite alphabet (e.g., $\Sigma = \{A, B, \ldots, Z\}$)

- But natural text has **rich linguistic structure**:
  - morphologic (words)
  - syntactic (sentences)
  - grammatical (clauses)
  - semantic (meaning)
  - pragmatic (context)
- The native representation misses out on these!



MOBY DICK.

CHAPTER I.

LOOMINGS.

Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

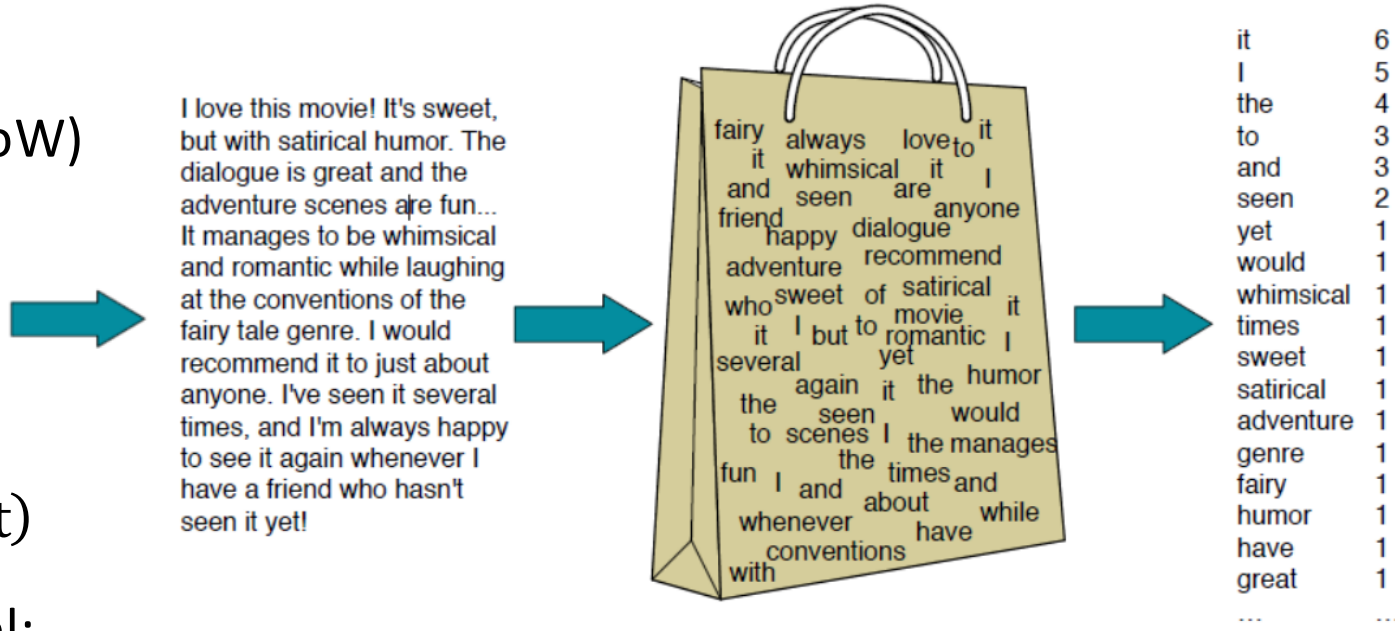
There now is your insular city of the Manhattoes, belted round by wharves as Indian isles by coral reefs—commerce surrounds it with her surf. Right and left, the streets take you waterward. Its extreme down-town is the battery, where that noble mole is washed by waves, and cooled by breezes, which a few hours previous were out of sight of land. Look at the crowds of water-gazers there.

Circumambulate the city of a dreamy Sabbath afternoon. Go from Corlears Hook to Coenties Slip, and from thence, by Whitehall, northward. What do you see?—Posted like

Representing text

- **Simplest approach**: Bag of Words (BoW)
- Text representation:

  histogram of word frequencies



- BoW effectively *vectorizes* text:

$$\big(\phi(\text{text})\big)_i = \text{count}(\text{word}_i; \text{text})$$

- On top, apply a standard linear model:

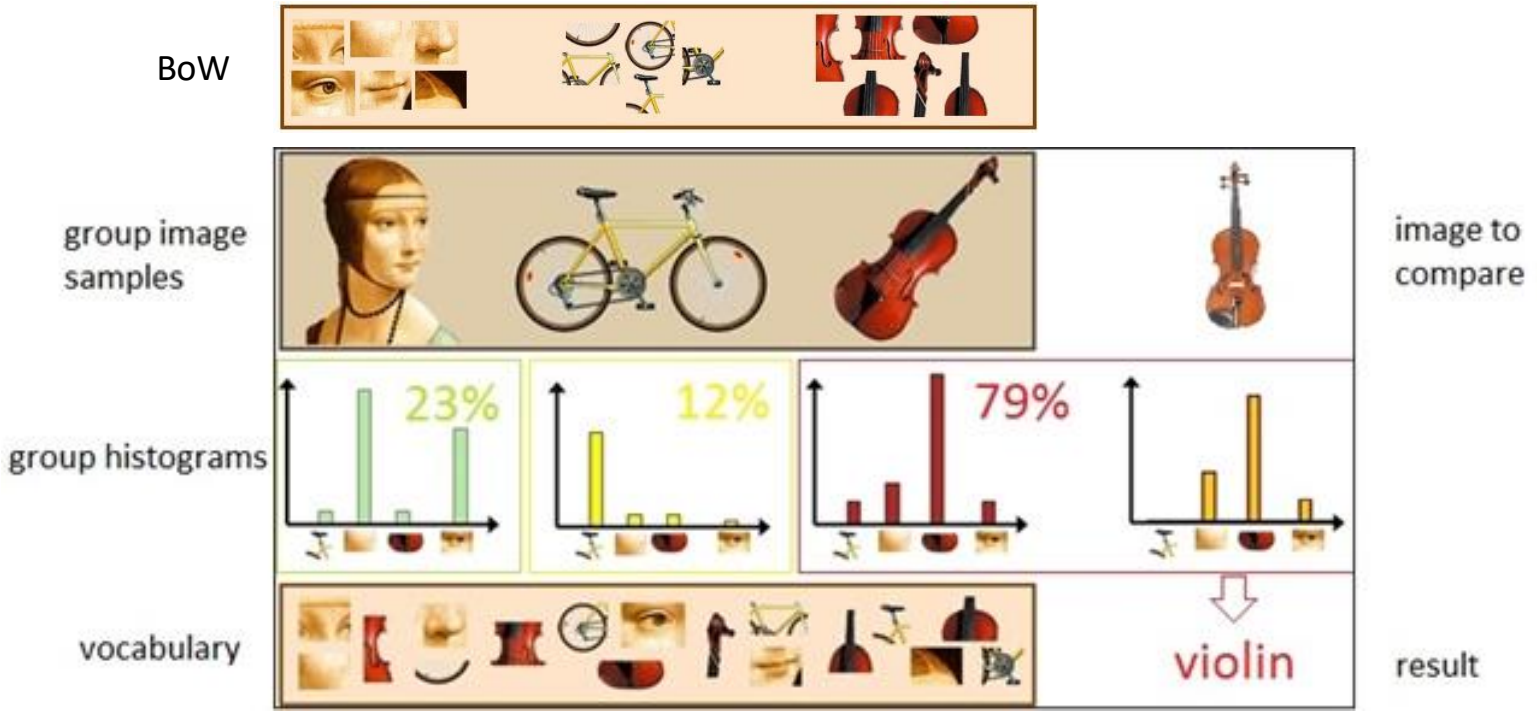
$$h_w(\text{text}) = w^\top \phi(\text{text}),\ \ w \in \mathbb{R}^{|\text{corpus}|}$$

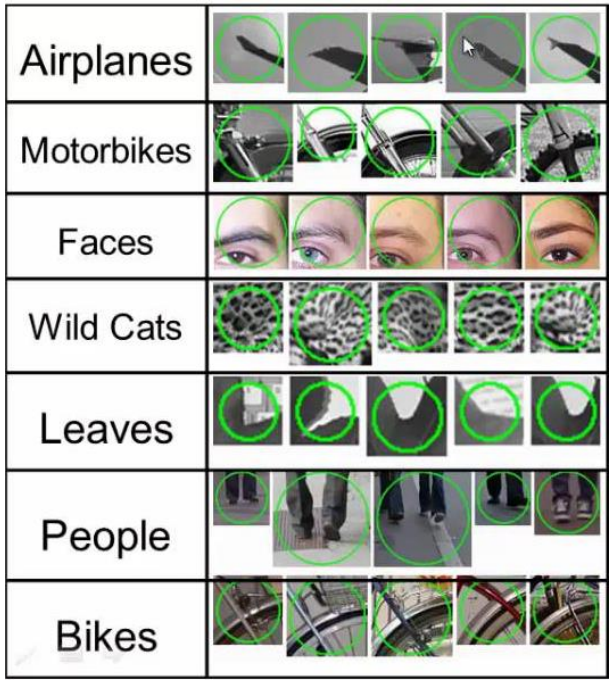
- Utilizes basic statistical linguistic properties (so uses some structure, but not all)
- Despite its simplicity, often works very well
- **Algorithmic challenge**: *huge* dimension
- Fortunately**,** natural text is **sparse**, which can be exploited

https://web.cs.hacettepe.edu.tr/~ilyas/Courses/CMP711/lec05-TextClassificationNaiveBayes.pdf

Representing objects

- The BoW approach applies broadly, and can be used for various input types
- For example, for natural images, can use **bag of *visual* words**:

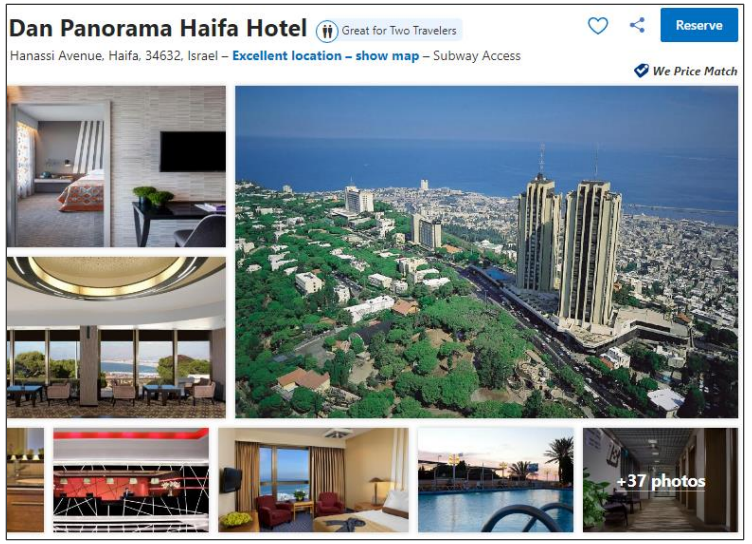




https://www.youtube.com/watch?v=iGZpJZhqEME, https://vgg.fiit.stuba.sk/2015-02/bag-of-visual-words-in-opencv/

Set models

- many domains have inputs that are sets:



- BoW represents objects as a set, but requires a fixed, finite corpus
- Not always the case! (e.g., elements described by arbitrary feature vectors)

Set models

- Consider simple case:
  - scalar elements $e \in \mathcal{E} = \mathbb{R}$
  - set inputs $x = \{e_1, \dots, e_n\} \subseteq \mathcal{E}$
- Let's try our approach:
  - $\phi(x) = (e_1, \dots, e_n)$
  - $h(x) = w^\top \phi\ (x)$
- **Won't work**! because:
  - ➢ set size may vary (vector size is constant)
  - ➢ require permutation invariance: $h(x) = h\big(\pi(x)\big)$ for any permutation π

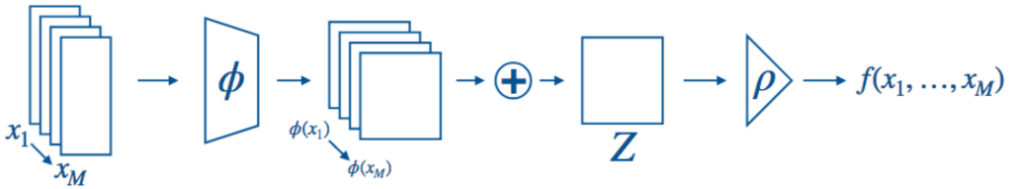
Set models

- Sets ≠ vectors, because in sets, **order does not matter** (and neither does size)
- Implementing a set function using a vector function requires **permutation invariance**:

$$\text{if } z = \text{vec}(x), \text{ then require } h(z) = h\big(\pi(z)\big) \text{ for any permutation } \pi$$

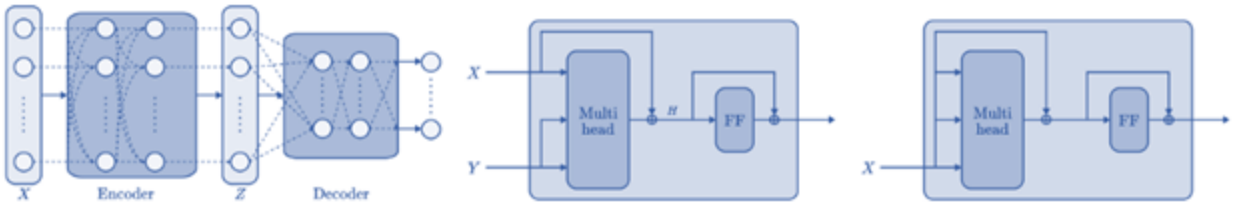
- We've already seen invariance as a form of prior knowledge (in CNNs)
- **Challenge**: for set inputs, design expressive permutation-invariant models
- **Result**: any permutation-invariant function has to be of the form $h(x) = \rho(\underbrace{\sum_i \phi(e_i)}_{\text{vector}})$ *same ϕ applied to all elements*
- **Bonus**: permutation-invariant neural architectures:

deep sets



https://arxiv.org/pdf/1703.06114.pdf

set transformer

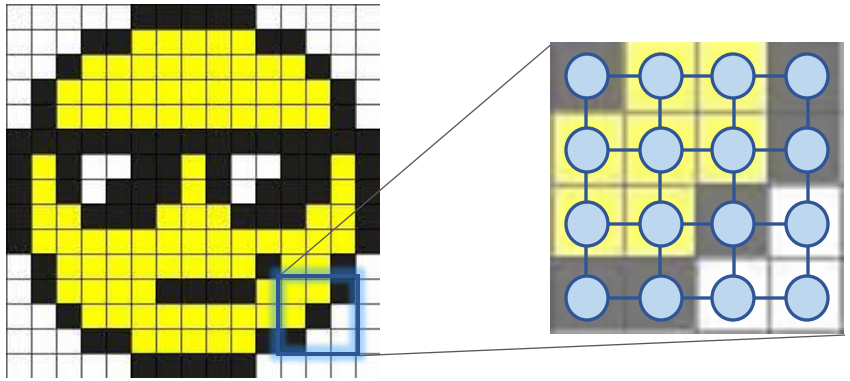


http://proceedings.mlr.press/v97/lee19d/lee19d.pdf

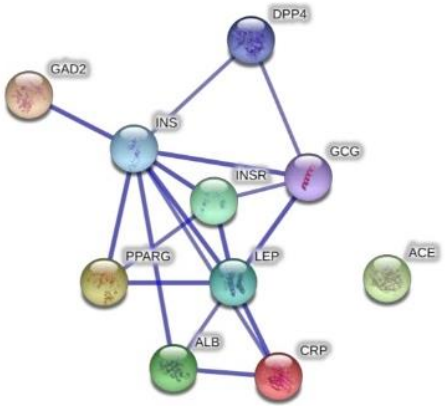
Beyond sets

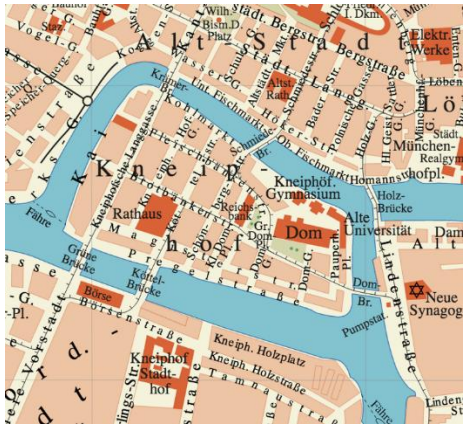
- Sets are combinatorial objects without any additional structure
- But other combinatorial objects do have structure
- We will briefly review:
  - **Graphs** having *spatial* structure
  - **Sequences** having *temporal* structure

Spatial structure



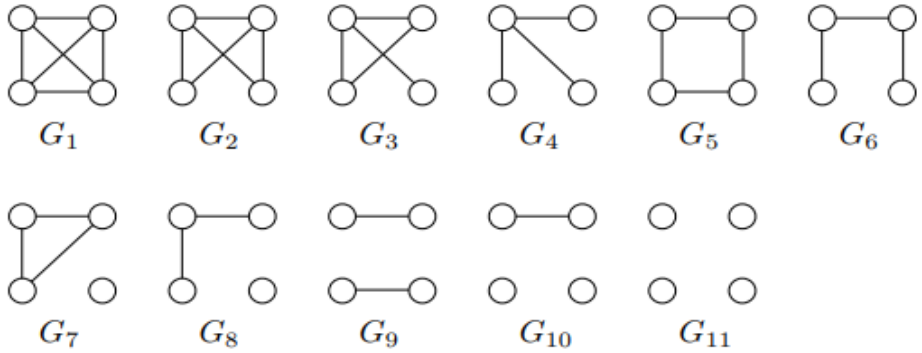
- Images have spatial “grid” structure
- **Model images using graphs**:
  - nodes = pixels
  - edges = spatial proximity
- Inputs are now graphs: $x = G = (V, E)$
  (often nodes and labels have “labels” (aka tags), such as type, strength, features, etc.)
- Can now think of CNNs as learning certain vector representations of grid-graph objects
- Many problem domains have graph-based inputs:
- **Q**: What are good representations for arbitrary graphs $\phi(\text{graph})$?

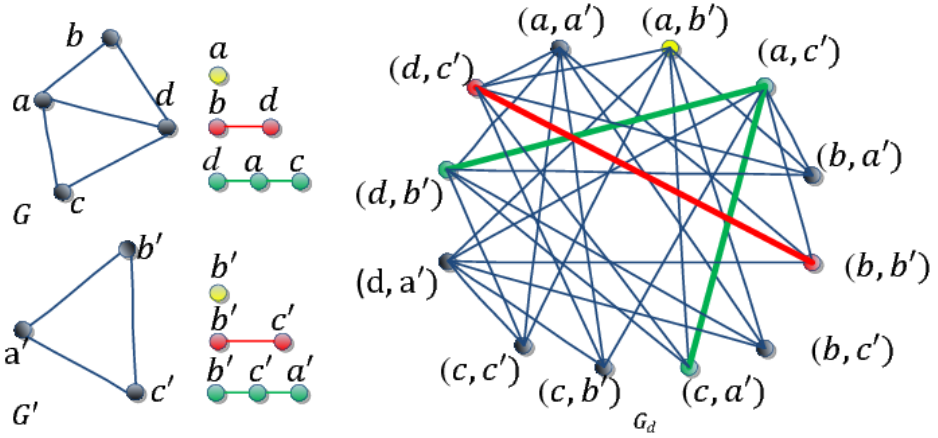




Graphs kernels

- **A1**: kernels
- Need $\phi: G \to \mathbb{R}^k$ s.t. $K(G, G') = \langle \phi(G), \phi(G') \rangle$:
  - is a valid inner product
  - captures meaningful similarities between graphs
  - can be computed efficiently
- **Examples**:
  1. graphlet kernel (aka “bag of subgraphs”)
  2. shortest-paths graph kernel
  3. random walk graph kernel
  4. WL graph kernel (akak “isomorphism” kernel)

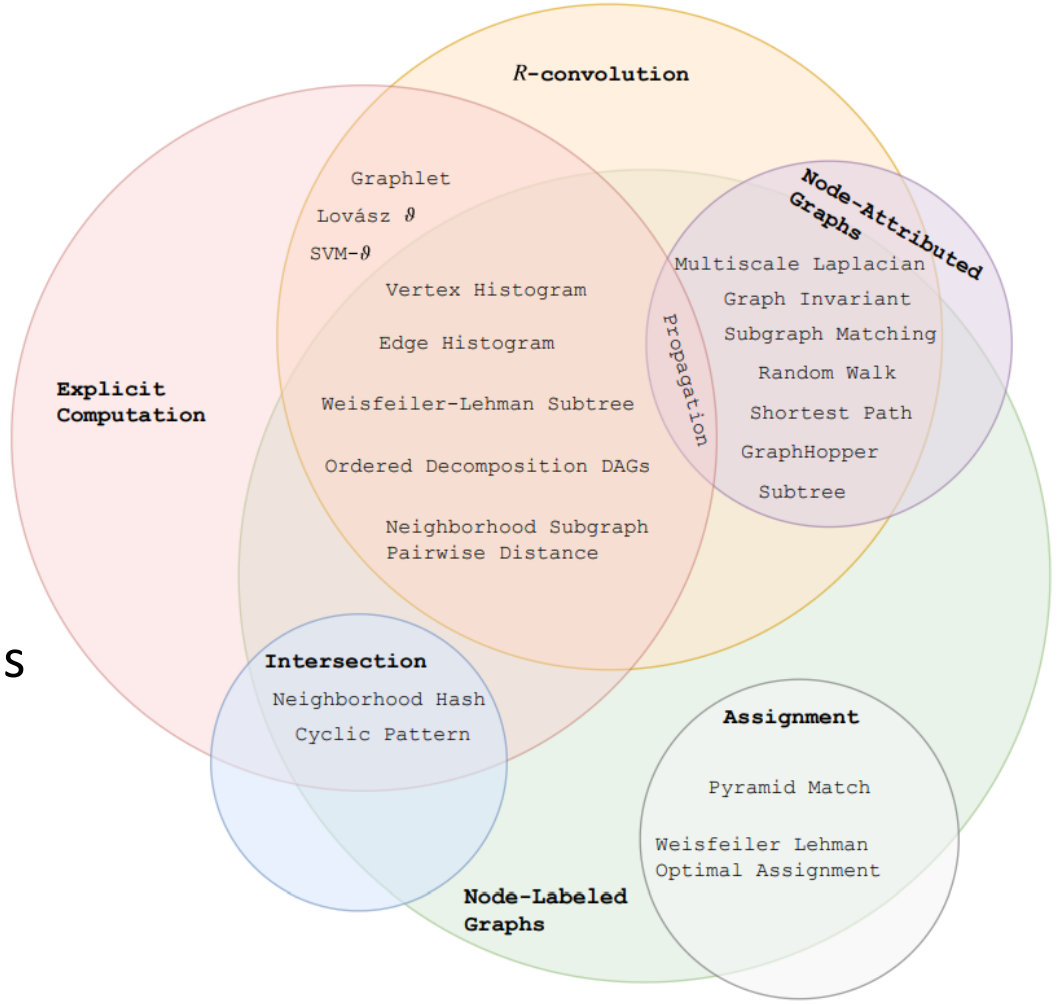




https://arxiv.org/pdf/1904.12218.pdf

Graphs kernels

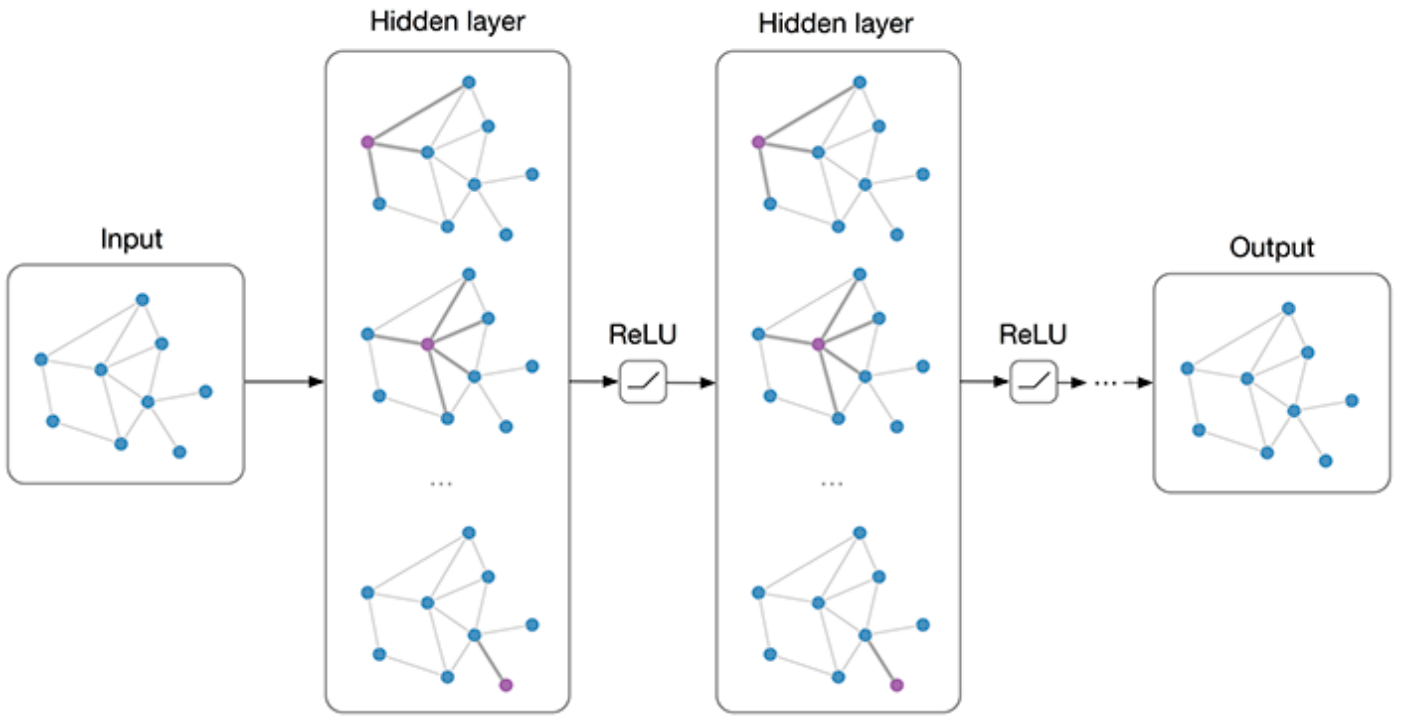
- **A1**: kernels
- Need $\phi: G \to \mathbb{R}^k$ s.t. $K(G, G') = \langle \phi(G), \phi(G') \rangle$:
  - is a valid inner product
  - captures meaningful similarities between graphs
  - can be computed efficiently
- **Examples**:
  1. graphlet kernel (aka “bag of subgraphs”)
  2. shortest-paths graph kernel
  3. random walk graph kernel
  4. WL graph kernel (akak “isomorphism” kernel)
  5. …



https://arxiv.org/pdf/1904.12218.pdf

Graphs Neural Networks (GNNs)

- **A2**: Graph Neural Networks* (GNNs)
- Mostly aim to generalize "convolution" from grids to arbitrary graphs
- **Challenging**! Graphs vary in size, nodes, edges, labels/tags, etc.
- Many, many variations on this idea



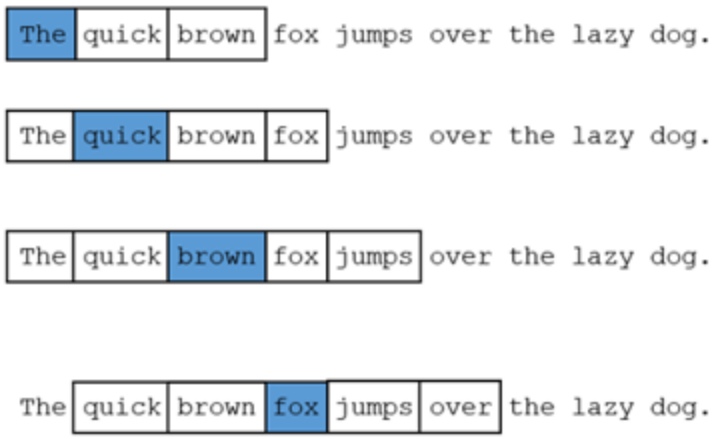
* Confusing! some GNNs operate on graph inputs; others operate on vector inputs tied by graph structure

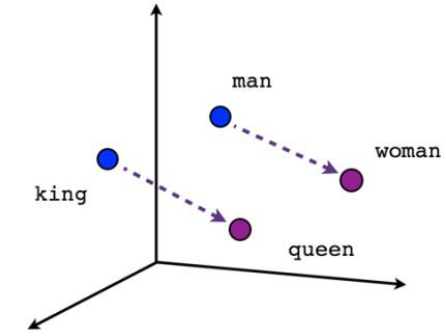
https://tkipf.github.io/graph-convolutional-networks/

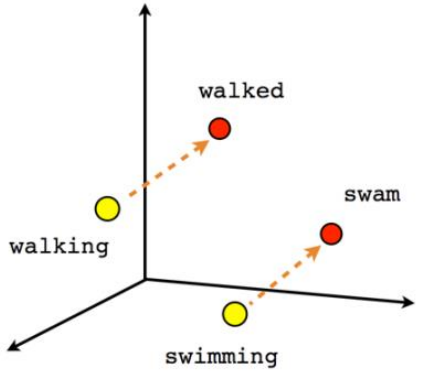
Temporal structure

- **Observation**: text is a *sequence* – order is important!
- **Approaches**: (high-level)

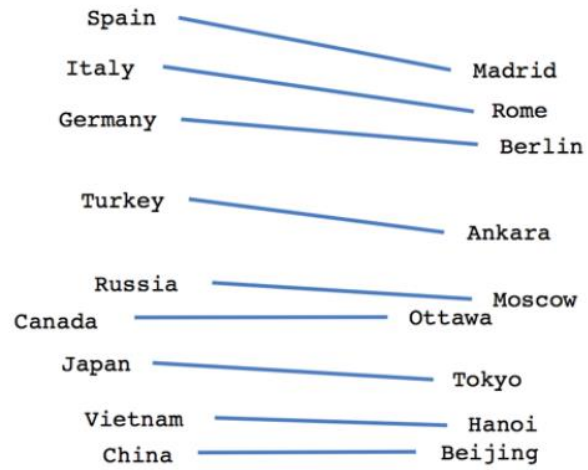
wrod2vec







RNN **LSTM**



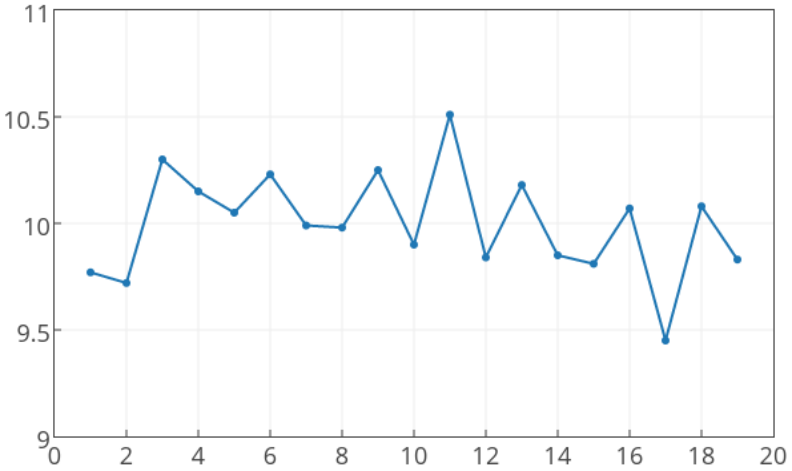
- Apply to any sequence-based inputs

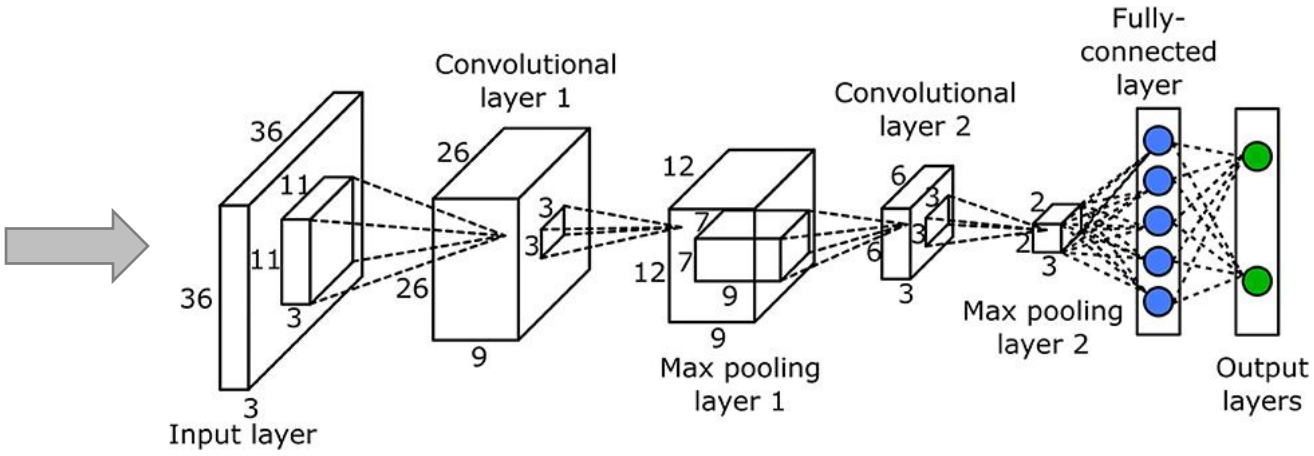
Choosing structure

- **Q**: How should we choose which structure to utilize, and how?
- **Modeling example**: recommendation systems

  x = collection (aka "slate") of n recommended items $\{\boldsymbol{i}_1, \dots, \boldsymbol{i}_n\}$, each $\boldsymbol{i} \in \mathbb{R}^d$
- Labels derived from user interaction with items – click, watch, buy, share, etc.
- Can represent slate x as:
  - individual items (assume user chooses best item)
  - set (alternatives can influence choice – even irrelevant ones!)
  - graph (inter-item dependencies – e.g., similarity)
  - sequence (order matters! higher-ranked items more important?)
- **A**: Ask yourself:
  - ❑ What kind of structure exists?
  - ❑ Can it be utilized? Should it be utilized?
  - ❑ How? At what cost, and is it worthwhile? (e.g., vs. simple approach)

Bad modeling





Structure in output space

Beyond binary and scalar

Beyond binary/scalar

- The world is not binary, nor is it scalar
- Learning setting that extend to other output types:
  - **multiclass:** one object out of many, $y \in \mathcal{Y} = [K]$
  - **multilabel**: multiple objects, $y \in \mathcal{Y} = 2^{[K]}$ (equivalently, $y \subseteq [K]$)
  - **structured**: e.g., relations between objects, e.g., $y \in \mathcal{Y} = 2^{[K] \otimes [K]}$

increasingly complex combinatorial spaces

binary	**multiclass**	**multilabel**	**structured**
			
dog / ~~not dog~~	dog $\in$ Animals	$\{\text{cat}, \text{dog}\} \subseteq$ Animals	chasing(cat, dog)

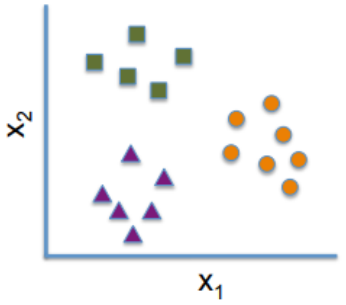
Beyond binary/scalar

- To support complex combinatorial output spaces, need to change:
  - **Model**: $h: \mathcal{X} \to \mathcal{Y}$
  - **Loss**: $\ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

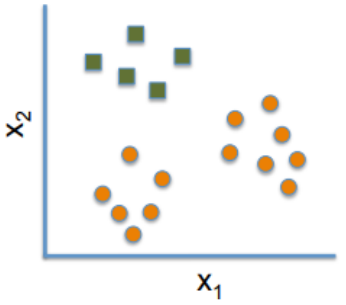
(where $\mathcal{Y} = [K]$ for multiclass, $\mathcal{Y} = 2^{[K]}$ for multilabel, etc.)

- Let's start with **multiclass**
- Common approaches
  1. **reductions** (1v1, 1vA)
  2. **relaxations** (softmax)
  3. ~~**label-dependent representations**~~

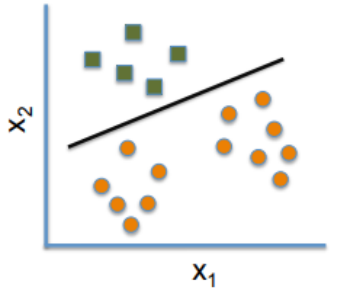
Reductions



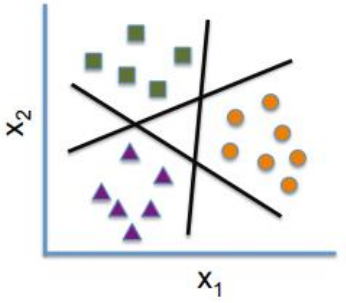
- **Simple idea**: combine multiple learned binary predictors to form single multiclass predictor
- Called a reduction because method "wraps" binary learner $A(S)$



- **<u>One vs. All (1vA):</u>**
  - for $i = 1, \ldots, K$
    - construct sample set $S_i = \left\{\left(x_j, y_j^{(i)}\right)\right\}_{j=1}^{m}$ where $y_j^{(i)} = \mathbb{1}\{y_j = i\}$
    - train binary predictor $h_i = A(S_i)$
  - return multiclass predictor $h(x) = \underset{i \in [K]}{\operatorname{argmax}}\, h_i(x)$

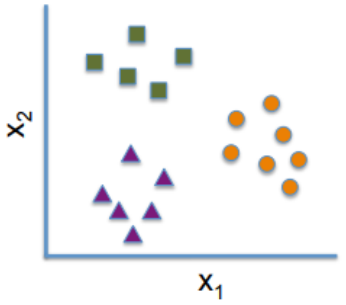


- **Problem**: what if there are multiple $h_i(x) = 1$?
- **Solution**: use "confidence"

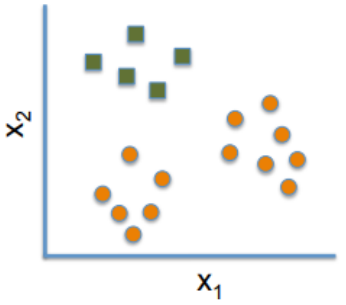


https://www.andrew.cmu.edu/course/18-661/lectures/multiclass.pdf (p65)

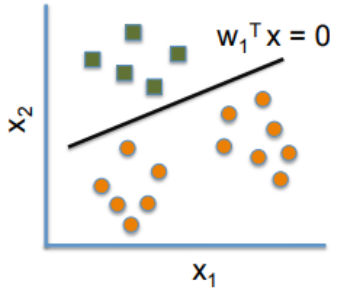
Reductions



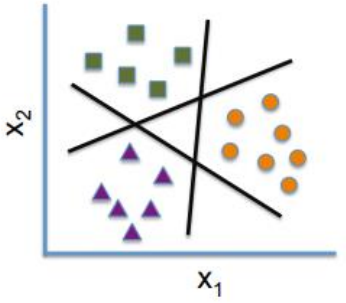
- **Simple idea**: combine multiple learned binary predictors to form single multiclass predictor
- Called a reduction because method "wraps" binary learner $A(S)$



- **<u>One vs. All (1vA):</u>**
  - for $i = 1, \ldots, K$
    - construct sample set $S_i = \left\{\left(x_j, y_j^{(i)}\right)\right\}_{j=1}^{m}$ where $y_j^{(i)} = \mathbb{1}\{y_j = i\}$
    - train scalar score function $f_i = A(S_i)$
  - return multiclass predictor $h(x) = \underset{i \in [K]}{\operatorname{argmax}}\, f_i(x)$



- **Hinge loss**: predict class by largest margin
- **Cross-entropy loss**: predict class with highest predictive probability

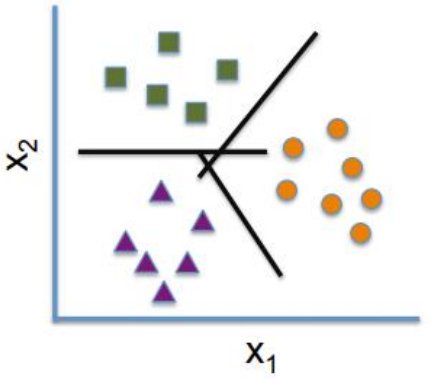


https://www.andrew.cmu.edu/course/18-661/lectures/multiclass.pdf (p65)

Reductions

- **Con**: 1vA tends to create highly imbalanced sub-problems, which can be problematic
- Alternative:

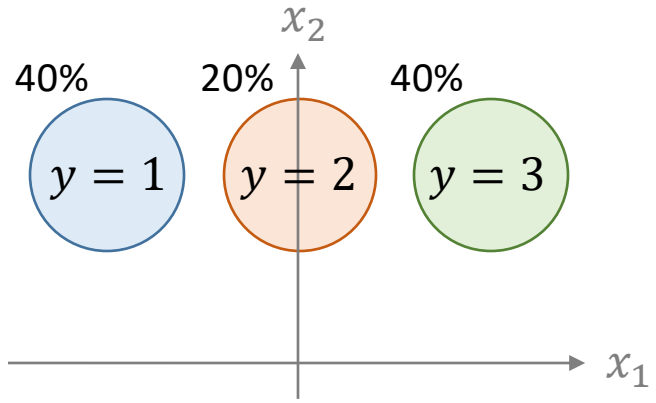
- **<u>One vs. One (1v1):</u>**
  - for $i = 1, \dots, K$
    - for $j = i + 1, \dots, K$
      - construct sample set $S_{ij} = \{(x_\ell, y_\ell) : y_\ell \in \{i, j\}\}$
      - train binary predictor $h_{ij} = A(S_{ij})$, set $h_{ji} = -h_{ij}$
  - return multiclass predictor $h(x) = \underset{i \in [K]}{\operatorname{argmax}} \sum_{j \neq i} h_{ij}(x)$



- **Con**: computationally demanding when K is large

Reductions can fail

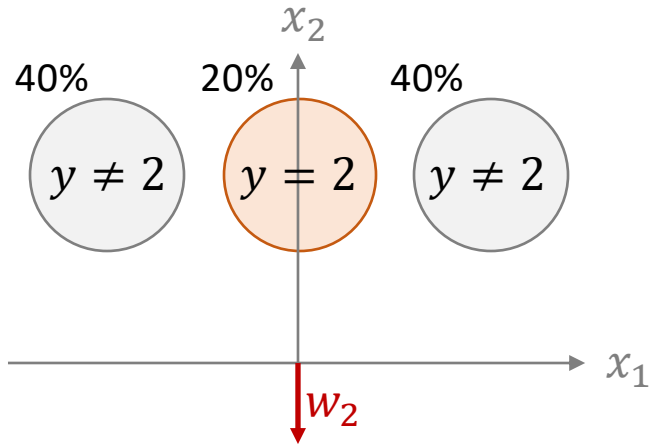
- Reductions are appealing due to their simplicity, but this simplicity has a price
- Simple example of 1vA failure:



$$h(x) = \underset{i \in [k]}{\operatorname{argmax}} f_i(x)$$

Reductions can fail

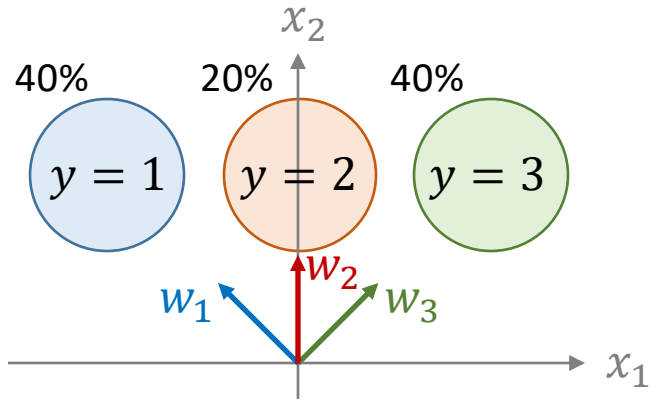
- Reductions are appealing due to their simplicity, but this simplicity has a price
- Simple example of 1vA failure:



$$h(x) = \underset{i \in [k]}{\operatorname{argmax}} f_i(x)$$

Reductions can fail

- Reductions are appealing due to their simplicity, but this simplicity has a price
- Simple example of 1vA failure:
- **Compare** – possible classes:
  - cat, dog, sheep, pig, snake
  - cat, poodle, labrador, colli, hound
  - mammals, reptiles, dog, cat, snake, lizzard
  - …
- **Conclusion**:
  - Reductions do not properly utilize structure
  - Should learn multiclass model directly and using an appropriate loss



$$h(x) = \underset{i \in [k]}{\operatorname{argmax}} f_i(x)$$

Softmax

- **Goal**: learn $h(x) = \underset{i \in [K]}{\operatorname{argmax}} f_i(x) = \operatorname{argmax}(f_1(x), \dots, f_K(x))$ directly
- **Learning objective**: $\underset{h \in H}{\operatorname{argmin}} \mathbb{E}\left[\mathbb{1}\left(y \neq h(x)\right)\right]$ where $y \in [K]$ ← *multiclass 0/1 loss*
- **Problem**: $h(x)$ includes an argmax operator, which is non-continuous (and non-differentiable)
- **Solution**:
  - Relax "hard" argmax operator to differentiable softmax operator:

$$f(x) = \operatorname{softmax}\left(f_1(x), \dots, f_K(x)\right) \coloneqq \left(\frac{e^{f_1(x)}}{\sum_{i \in [K]} e^{f_i(x)}}, \dots, \frac{e^{f_K(x)}}{\sum_{i \in [K]} e^{f_i(x)}}\right) = \hat{y} \in [0,1]^K$$

  - Use cross entropy as proxy loss: $\ell(y, \hat{y}) = -\sum_{i \in [K]} y_i \log \hat{y}_i$
- **Example (*K=3*)**: suppose $f_1(x) = -0.5,\ f_2(x) = 0,\ f_3(x) = 3$

$\operatorname{argmax}(-0.5, 0, 3) = 3$, one-hot $\mapsto (0,0,1)$

$$\operatorname{softmax}(-0.5, 0, 3) = \left(\frac{\mathrm{e}^{-0.5}}{\mathrm{e}^{-0.5} + \mathrm{e}^{0} + \mathrm{e}^{3}}, \frac{\mathrm{e}^{0}}{\mathrm{e}^{-0.5} + \mathrm{e}^{0} + \mathrm{e}^{3}}, \frac{\mathrm{e}^{3}}{\mathrm{e}^{-0.5} + \mathrm{e}^{0} + \mathrm{e}^{3}}\right) \approx (0.028, 0.046, 0.926)$$

Softmax

- **Properties**:
  1. When f_i are scaled with "temperature" β, softmax approaches argmax (as 1-hot vector)

$$\text{softmax}(f_1(x), \dots, f_K(x); \beta) = \begin{pmatrix} \frac{e^{\beta f_1(x)}}{\sum_{i\in[K]} e^{\beta f_i(x)}} \\ \frac{e^{\beta f_2(x)}}{\sum_{i\in[K]} e^{\beta f_i(x)}} \\ \vdots \\ \frac{e^{\beta f_K(x)}}{\sum_{i\in[K]} e^{\beta f_i(x)}} \end{pmatrix} \xrightarrow[\beta\to\infty]{} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

argmax

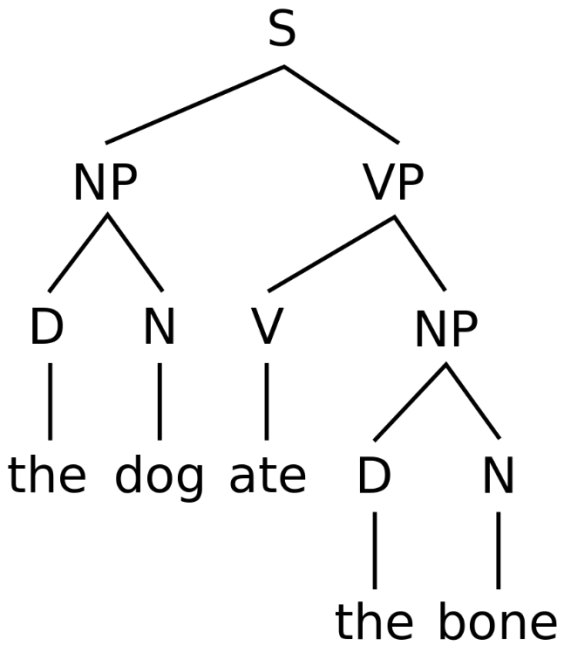
	β= **0.1**	**0.5**	**1**	**2**	**5**
$f_1(x)$= **0.3**	0.25	0.23	0.17	0.08	0.00
1.4	0.28	0.39	0.53	0.73	0.97
0.7	0.26	0.28	0.26	0.18	0.03
-1.2	0.21	0.11	0.04	0.00	0.00
					=softmax

- ➢ large β – tighter approximation
- ➢ small β – smoother approximation

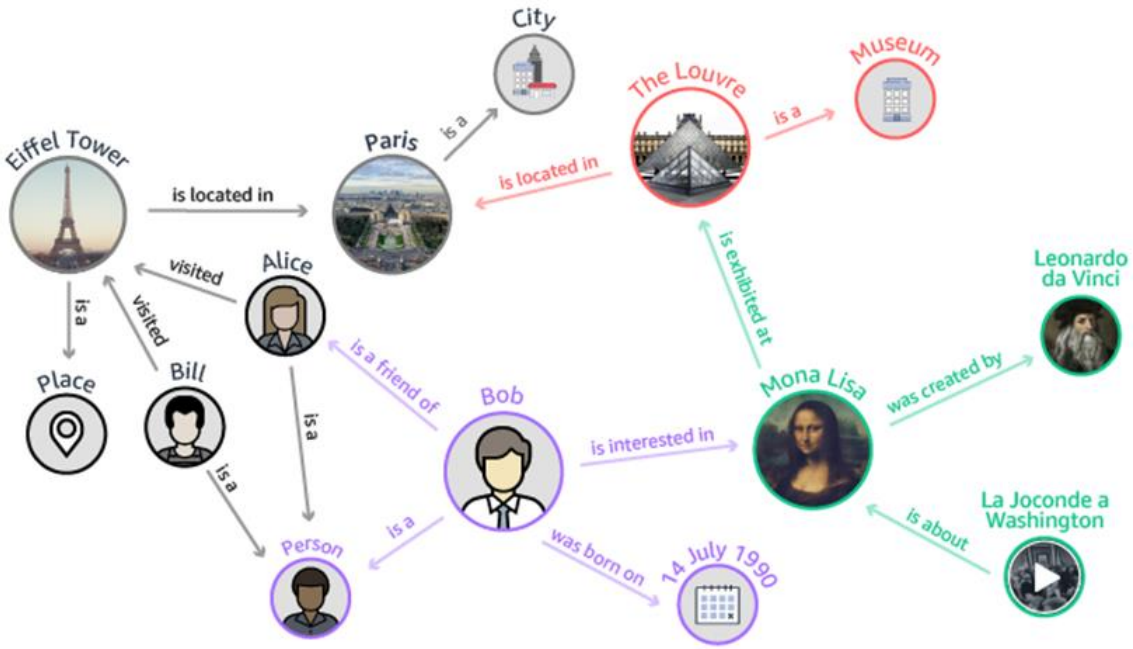
  2. Differentiable; convex for linear f_i and cross-entropy (aka log-loss)
  3. Normalization means entries sum to 1; gives probability distribution over classes

- **We've seen this!** when $K = 2$, get (binary) logistic regression as special case (details in tirgul)
- Softmax + cross-entropy is the predominant approach in modern deep learning

Structured outputs



dependency parsing



knowledge graph

Structure in learning

Beyond iid prediction

Beyond iid

- The iid assumption – single largest assumption made ever
- Gives lots of power, but also the **Achilles' heal** of learning
- **Tip**: when something goes wrong – look for violations of iid
- **Special case:**
  distribution shift: $p = p_{\text{observed}} \neq p_{\text{deploy}} = p'$, but iid within each
- **Goal**: low expected error on $p' = p_{\text{deploy}}$
- **Problem**: have labeled data only from p, not from p'
- **Solution**: re-weight examples in loss to "mimic" p'
- **Special-special case:** (for which we can solve the above, sometimes)
  1. have labeled data from observed distribution, $(x, y) \sim p(x, y) = p(x)p(y|x)$
  2. have some unlabeled data from target distribution, $x \sim p'(x)$
  3. assumption – *covariate shift*: $p'(x, y) = p'(x)p(y|x)$ (only $p'(x)$ changed; $p(y|x)$ stayed the same)

$$p(x,y) = p(x)q(y|x)$$
$$p'(x,y) = p'(x)q(y|x)$$

- $\mathbb{E}_{p'(x,y)}[\ell(x,y)] = \mathbb{E}_{p'(x)}\left[\mathbb{E}_{p'(y|x)}[\ell(x,y)]\right] = \mathbb{E}_{p'(x)}\left[\underbrace{\mathbb{E}_{q(y|x)}[\ell(x,y)]}_{:= g(x)}\right] = \cdots$

- $\mathbb{E}_{p'}[g(x)] = \mathbb{E}_{p'}\left[\frac{p(x)}{p(x)}g(x)\right] = \int p'(x)\frac{p(x)}{p(x)}g(x)dx = \mathbb{E}_{p}\left[\underbrace{\frac{p'(x)}{p(x)}}_{:= w(x)}g(x)\right] = \mathbb{E}_{p}[w(x)g(x)]$

- $\ldots = \mathbb{E}_{p(x)}\left[w(x)\mathbb{E}_{q(y|x)}[\ell(x,y)]\right] = \mathbb{E}_{p(x,y)}[w(x)\ell(x,y)] \approx \frac{1}{m}\sum_i w(x_i)\ell(x_i,y_i)$

propensity weights

Covariate shift

- **Propensity weights:** $w(x) = \frac{p'(x)}{p(x)}$, many methods for estimation
- Great solution for non-iid data, as long as
  1. covariate shift holds, and
  2. unlabeled data from p' is available
- But we don't always have data from p'
- Sometimes p' does not even exist at train time
- Worse yet – sometimes learning itself is what determines p'

From predictions to decisions

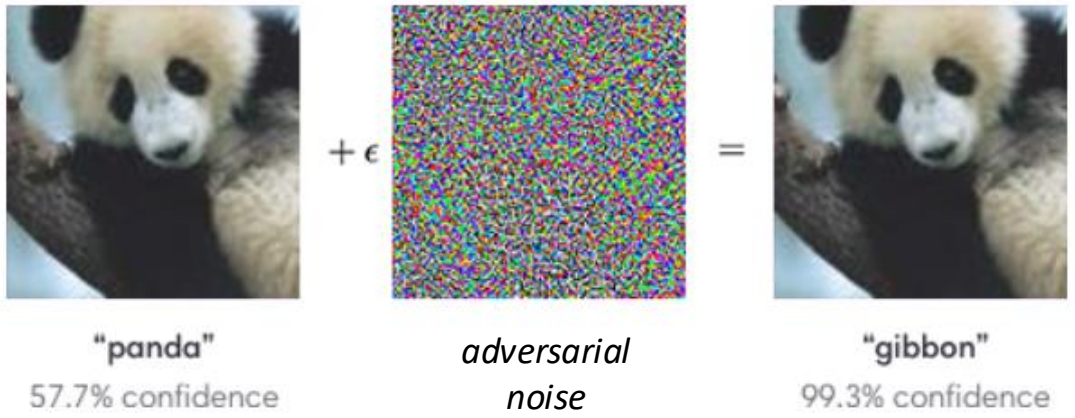
- To see how, consider some examples of (good?) uses for predictive ML:
  - recommendation, news, and media
  - hiring and admissions
  - finance, loans, and insurance
  - traffic and routing
  - healthcare
  - crime prevention
- In all the above, **predictions** are used to support **decisions**
- **Problem**:
  learning only considers *predictive accuracy* – not the *quality* of the induced *decisions*
- Let's see why and how this can cause trouble

Strategic behavior

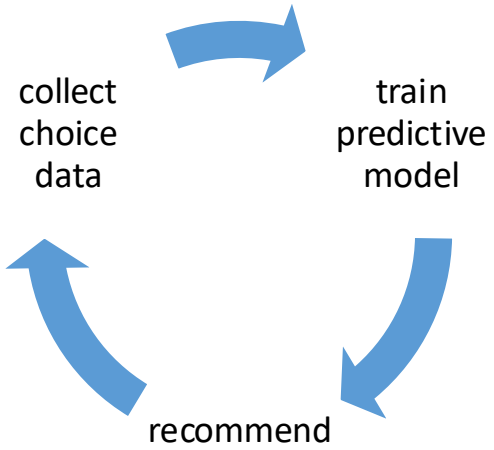
- Consider a bank interested in giving loans
- **Idea**:
  1. train $h(x)$ to predict if user x returned the loan ($y = 1$) or not ($y = 0$)
  2. for new x, use prediction $\hat{y} = h(x)$ to decide if to approve or reject loan
- **Problem**:
  - users want to get the loan ($\hat{y} = 1$), irrespective of true y!
  - incentivized to strategically modify x (at a cost) to some x' with $h(x') = 1$
  - examples: transfer between accounts, get additional credit card, ask for salary advance, …
- h is trained on "true" x, but tested on modified inputs x'
- In aggregate, modifications $x \mapsto x'$ amount to distribution shift
- But how this shift occurs depends on the choice of h!
- **Root cause**: users are self-interested (לא פראייריים)

Adversarial behavior

- Consider some traffic and routing app (not to name any)
- **Idea**: recommend routes based on learned predictive model, $f(x) = \mathrm{ETA}(\mathrm{GPS\ data}\ x)$
- **Problem**: malicious entities can *cause* traffic by reporting fake locations
- Actually done by students at the Technion:
  https://www.haaretz.com/.premium-israeli-students-fake-traffic-jam-on-waze-1.5338670
- Adversarial attacks on ML models are considered a major concern
  (especially in deep learning, which is believed to be particularly frail)
- **Disproportionately famous example**:
- **Root cause**:
  models are sensitive to tiny changes,
  which an adversary can exploit



Feedback



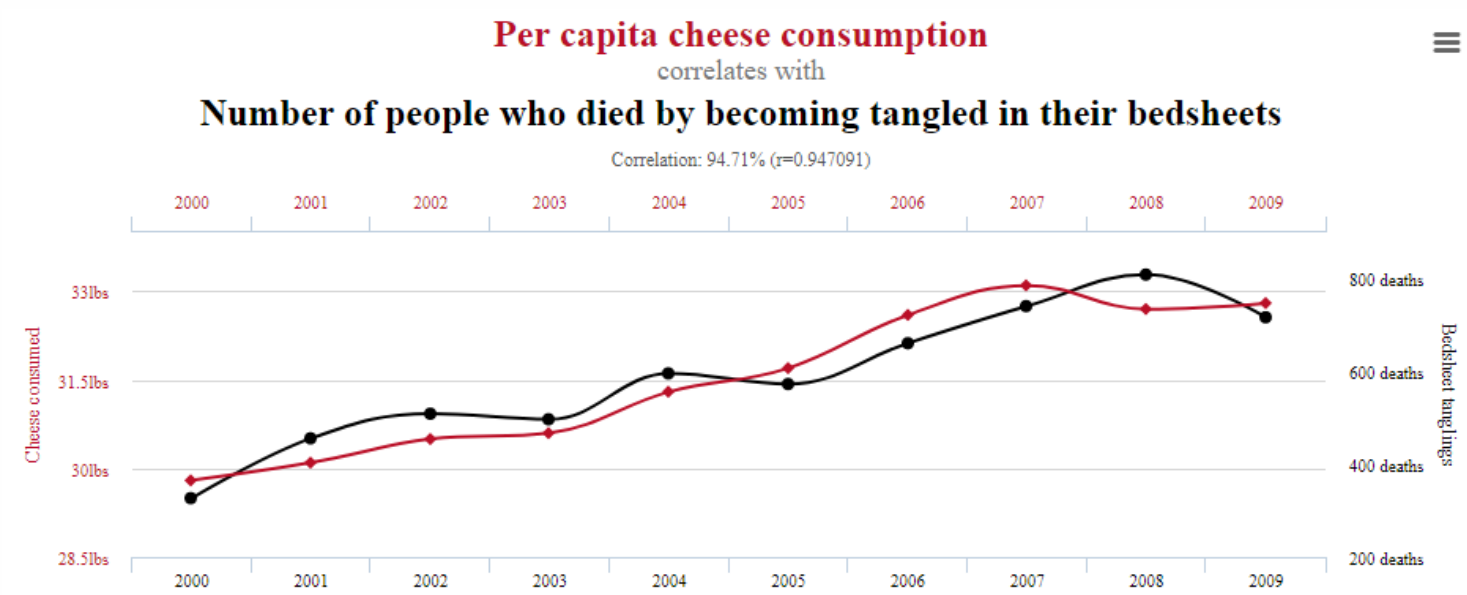
- Consider recommendation engines (news, media, ecommerce, social connections, etc.)
- **Typical process**:
  - collect labeled user data (x, y), where $x =$ recommedned items and $y =$ choices
  - train predictive model $f(x) \approx P(\text{user chose } x)$
  - use f to recommend items that users would likely choose (i.e., having large $f(x)$)
  - repeat!
- **Problem**:
  - the input distribution consists of user ratings on items recommended by the system
  - using f to recommend at the next time setp introduces feedback that changes the distribution
  - f has no guarantees on this new distribution (because it was trained on a different distribution)
  - In this sense, deploying a predictive model causes the data distribution to change over time
- **Root cause**: prediction is myopic, does not account for long-term interaction with active users

Feedback

- So the distribution in recommendation changes over time – is this good or bad?
- On the one hand, change means new content – novel, diverse, surprising
- On the other hand, a large body of research argues that these dynamics lead to:
  - filter bubbles
  - echo chambers
  - hyper-popularization
  - polarization

Predictions vs. decisions

- In all of our examples, predictions were used as the basis for decisions
- This is conceptually flawed:
  - **predictions** are based on **correlation**
  - **decisions** are based on **causation**



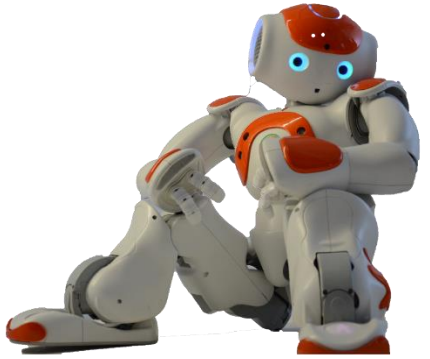
Predictions vs. decisions

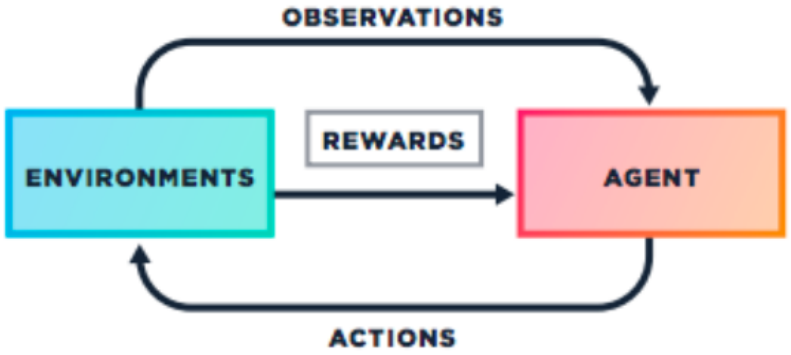
- In all of our examples, predictions were used as the basis for decisions
- This is conceptually flawed:
  - **predictions** are based on **correlation**
  - **decisions** are based on **causation**

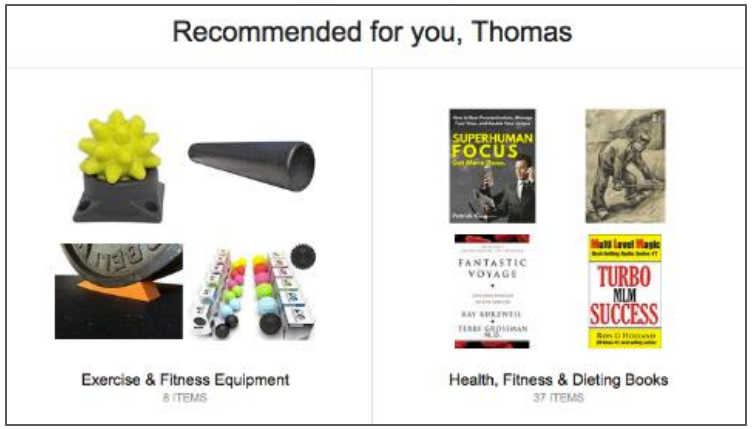
- If decisions cause distribution shift, this **breaks the iid assumption**
- In principle, performance guarantees (e.g., held-out validation) are no longer valid
- Sometimes the effects aren't too bad, but at times they can be catastrophic
- But most times – we can't really know
- **Take-away**: think carefully on what you plan to do with your predictions

Ending on a positive note

- Thankfully, lots of research on how to learn properly in these challenging settings
- But more generally, and as we've noted, *learning is not all about prediction*
- In fact, certain branches of machine learning are *all about decisions*
- **Key example**:
  reinforcement learning (way outside of our scope)







Course summary

Our course in retrospect

- **Part I:** *supervised binary classification*
  - Introduction
  - Methods
  - SVM in depth
- **Part II:** *aspects of learning*
  - Statistical: generalization
  - Modeling: model selection, validation
  - Optimization: gradient descent
  - Practical aspects of learning
- **Part III:** *more supervised learning*
  - Regression
  - Bagging and boosting
  - Deep learning
- **Part IV**: *beyond supervised learning*
  - ~~Unsupervised learning: dimensionality reduction~~
  - Other learning settings
    + course summary

types of learning

input:
feature vectors
images
text
speech
behavior
graphs
time-series
…

output:
binary
multiclass
multilabel
structured
scalars
intervals
distributions
…

supervision:
supervised
unsupervised
semi-supervised
weakly supervised
zero-shot
…

approach:
discriminative
generative
hybrid
…

environment:
batch learning
online learning
reinforcement learning
active learning
metric learning
multi-task learning
learning to teach
meta-learning
…

types of learning

input:
feature vectors
images
text
speech
behavior
graphs
time-series
…

output:
binary
multiclass
multilabel
structured
scalars
intervals
distributions
…

supervision:
supervised
unsupervised
semi-supervised
weakly supervised
zero-shot
…

approach:
discriminative
generative
hybrid
…

environment:
batch learning
online learning
reinforcement learning
active learning
metric learning
multi-task learning
learning to teach
meta-learning
…

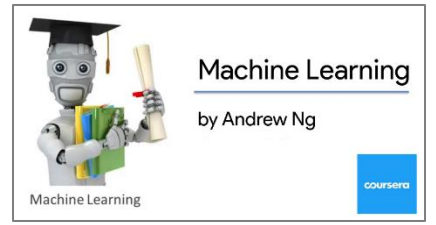
Related and advanced courses

- **קורסי בסיס בפקולטה:**
  - מבוא לבינה מלאכותית
  - מבוא לאופטימיזציה
  - מבוא לסטטיסטיקה
  - ...
- **קורסים מתקדמים בפקולטה:**
  - למידה עמוקה על מאיצי חישוביים
  - מבוא לעיבוד שפות טבעיות
  - מערכות לומדות והתנהגות אנושית
  - חיזוי והסקה סטטיסטית
  - מודלי דיפוזיה בלמידה עמוקה
  - נושאים על מודלי למידה עמוקה
  - נושאים נבחרים בטרנספורמרים
  - ...

- **קורסים מחוץ לפקולטה:**
  - תכנון ולמידה מחיזוקים (RL)
  - עיבוד וניתוח מידע (unsupervised)
  - מבוא להסקה סיבתית
  - מערכות סוכנים חכמים
  - למידה עמוקה ותורת הקירובים
  - ...

Recap

=



+

- **Recall**: one of our goals for the course was to give you an "edge" in machine learning
- Made up, roughly correct statistic:
  - ML is 90% predictive, discriminative, supervised learning
  - 10% other stuff
- **The edge lies in the 10% –**
  but succeeding there requires a fundamental understanding of the core 90%
- This means *theory* and *practice*
- **Key take-away**: remember that

  machine learning = **handling uncertainty** using **modeling** + **statistics** + **optimization**

  (and that they are tightly and inseparably interconnected)
- Now you are ready!

Good luck!

The exam

- In principal, everything in the lectures is kosher exam material (except “bonus” sections)
- We trust you to infer what is essential and what is not
- **Rule of thumb**: ask yourself if we did something *with it*, *towards it*, or *used it*
- **Formulas**:
  - Should know basic formulas, for example:
    gradient descent update, bias variance, PAC learning definition, shattering
  - In-depth example – generalization:
    - exact form of generalization bound – no
    - O-notation of rates per setting – yes
- More details in the tirgul