CS 1699: Deep Learning

Advanced and Recent Topics

Prof. Adriana Kovashka
University of Pittsburgh
April 2, 2020
Plan for this lecture

• Alternative representations
  – I. Graph networks (pp 3-22)
• Alternative learning mechanisms
  – II. Self supervision (pp 23-63)
  – III. Reinforcement learning (pp 64-101)
• Alternative tasks
  – IV. Generation (pp 102-190)
• V. Bias and ethics (pp 191-256)
Part I: Graph Networks

• Types of graph networks
  – Graph convolutional networks
  – Graph attention networks

• Applications
  – Semi-supervised learning
  – Visual question answering
Types of data typically handled with Deep Learning

- **IMAGENET**
- **Speech data**
- **Natural language processing (NLP)**
- **Grid games**

...
Graph-structured data

A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

Protein interaction networks

Knowledge graphs

Molecules

Road maps

Standard deep learning architectures like CNNs and RNNs don’t work here!
Graph Neural Networks (GNNs)

The bigger picture:

Notation: \( G = (A, X) \)
- Adjacency matrix \( A \in \mathbb{R}^{N \times N} \)
- Feature matrix \( X \in \mathbb{R}^{N \times F} \)

Main idea: Pass messages between pairs of nodes & agglomerate
Graph convolutional networks

Graph: $G = (\mathcal{V}, \mathcal{E})$

Adjacency matrix: $A$

$$
A = \begin{bmatrix}
0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 \\
\end{bmatrix}
$$
Recap: Convolutional neural networks (on grids)

Single CNN layer with 3x3 filter:

$\begin{align*}
&h_0 \\
&h_1 \\
&\ldots \\
&h_i
\end{align*}$

Update for a single pixel:
- Transform messages individually $W_i h_i$
- Add everything up $\sum_i W_i h_i$

$h_i$ in $\mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:
\[
h_{4}^{(l+1)} = \sigma \left( W_{0}^{(l)} h_{0}^{(l)} + W_{1}^{(l)} h_{1}^{(l)} + \cdots + W_{8}^{(l)} h_{8}^{(l)} \right)
\]
Graph convolutional networks

**Input**: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$

$$\mathbf{X} = \mathbf{H}^{(0)}$$

$$\mathbf{H}^{(l+1)} = \sigma \left( \hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right)$$

Kipf and Welling, “Semi-supervised learning with deep generative models”, ICLR 2017 (slides by Thomas Kipf)
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:

Calculate update for node in red:

**Update rule:**

$$h_i^{(l+1)} = \sigma \left( h_i^{(l)} W_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l)} W_1^{(l)} \right)$$

**Scalability:** subsample messages [Hamilton et al., NIPS 2017]  

$\mathcal{N}_i$: neighbor indices  

$c_{ij}$: norm. constant (fixed/trainable)
Alternatives

How else can we propagate information over a graph?

How can we improve the propagation using ideas from RNNs?

From CNNs?
Graph neural networks with attention
Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)

\[ \tilde{h}_i' = \sigma \left( \frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} \alpha_{ij}^k W^k \tilde{h}_j \right) \]

\[ \alpha_{ij} = \frac{\exp \left( \text{LeakyReLU} \left( \tilde{a}^T [W \tilde{h}_i \| W \tilde{h}_j] \right) \right)}{\sum_{k \in N_i} \exp \left( \text{LeakyReLU} \left( \tilde{a}^T [W \tilde{h}_i \| W \tilde{h}_k] \right) \right)} \]
A brief history of graph neural nets

“Spatial methods”
- Original GNN: Gori et al. (2005)
- GG-NN: Li et al. (ICLR 2016)
- MoNet: Monti et al. (CVPR 2017)
- Neural MP: Gilmer et al. (ICML 2017)
- GCN: Kipf & Welling (ICLR 2017)

“Spectral methods”
- Spectral Graph CNN: Bruna et al. (ICLR 2015)
- ChebNet: Defferrard et al. (NIPS 2016)
- Relation Nets: Santoro et al. (ICLR 2018)
- GraphSAGE: Hamilton et al. (NIPS 2017)
- Programs as Graphs: Allamanis et al. (ICLR 2018)
- GAT: Veličković et al. (ICLR 2018)
- NRI: Kipf et al. (ICML 2018)

“DL on graph explosion”
- “Spatial methods”

Other early work:
- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt’s talk on GNNs)
What do learned representations look like?

Forward pass through untrained 3-layer GCN model

Parameters initialized randomly

What else are graph representations good for?
Semi-supervised classification on graphs

**Setting:**
Some nodes are labeled (black circle)
All other nodes are unlabeled

**Task:**
Predict node label of unlabeled nodes

Evaluate loss on labeled nodes only:

\[
\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}
\]

- \(\mathcal{Y}_L\) set of labeled node indices
- \(Y\) label matrix
- \(Z\) GCN output (after softmax)
Application: Classification on citation networks

**Input:** Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

**Target:** Paper category (e.g. stat.ML, cs.LG, …)

**Model:** 2-layer GCN $Z = f(X, A) = \text{softmax}(\hat{A} \text{ReLU}(\hat{A}XW^{(0)})W^{(1)})$

**Classification results (accuracy)**

<table>
<thead>
<tr>
<th>Method</th>
<th>Citeseer</th>
<th>Cora</th>
<th>Pubmed</th>
<th>NELL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManiReg [3]</td>
<td>60.1</td>
<td>59.5</td>
<td>70.7</td>
<td>21.8</td>
</tr>
<tr>
<td>SemiEmb [24]</td>
<td>59.6</td>
<td>59.0</td>
<td>71.1</td>
<td>26.7</td>
</tr>
<tr>
<td>LP [27]</td>
<td>45.3</td>
<td>68.0</td>
<td>63.0</td>
<td>26.5</td>
</tr>
<tr>
<td>DeepWalk [18]</td>
<td>43.2</td>
<td>67.2</td>
<td>65.3</td>
<td>58.1</td>
</tr>
<tr>
<td>Planetoid* [25]</td>
<td>64.7 (26s)</td>
<td>75.7 (13s)</td>
<td>77.2 (25s)</td>
<td>61.9 (185s)</td>
</tr>
<tr>
<td>GCN (this paper)</td>
<td>70.3 (7s)</td>
<td>81.5 (4s)</td>
<td>79.0 (38s)</td>
<td>66.0 (48s)</td>
</tr>
<tr>
<td>GCN (rand. splits)</td>
<td>67.9 ± 0.5</td>
<td>80.1 ± 0.5</td>
<td>78.9 ± 0.7</td>
<td>58.4 ± 1.7</td>
</tr>
</tbody>
</table>

Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017
Visual Question Answering (VQA)

Task: Given an image and a natural language open-ended question, generate a natural language answer.

Visual Question Answering (VQA)

**Image Embedding**

Convolution Layer + Non-Linearity

Pooling Layer

Convolution Layer + Non-Linearity

Pooling Layer

Fully-Connected

4096-dim

**Neural Network**

Softmax over top K answers

**Question Embedding**

“How many horses are in this image?”

1024-dim

LSTM

Input (Features II) Softmax classifier

P(y = 0 | x)
P(y = 1 | x)
P(y = 2 | x)

Visual Question Answering (VQA)

Ask any question about this image
Visual Question Answering (VQA)

Figure 2. Our proposed framework: given an image, a CNN is first applied to produce the attribute-based representation $V_{att}(I)$. The internal textual representation is made up of image captions generated based on the image-attributes. The hidden state of the caption-LSTM after it has generated the last word in each caption is used as its vector representation. These vectors are then aggregated as $V_{cap}(I)$ with average-pooling. The external knowledge is mined from the KB (in this case DBpedia) and the responses encoded by Doc2Vec, which produces a vector $V_{know}(I)$. The 3 vectors $V$ are combined into a single representation of scene content, which is input to the VQA LSTM model which interprets the question and generates an answer.
Figure 2: Outline of the proposed approach: Given an image and a question, we use a similarity scoring technique (1) to obtain relevant facts from the fact space. An LSTM (2) predicts the relation from the question to further reduce the set of relevant facts and its entities. An entity embedding is obtained by concatenating the visual concepts embedding of the image (3), the LSTM embedding of the question (4), and the LSTM embedding of the entity (5). Each entity forms a single node in the graph and the relations constitute the edges (6). A GCN followed by an MLP performs joint assessment (7) to predict the answer. Our approach is trained end-to-end.
Graphs for advertisements

Figure 2: Overview of the proposed model. Given a single image ad, we first expand the representation using object detection and OCR, and also retrieve relevant knowledge based on slogan snippets (left). We build a graph-based model to infer the overall message using all available information (right). For more effective training, we mask query keywords and randomly drop certain knowledge pieces (shown in purple). More details are in Sec. 3.
Part II: Self-Supervised Learning

• Learn representations from context in raw data
  • Language – predict nearby words \textit{[already covered]}
    – Word2Vec
    – Transformers, BERT
  • Vision – predict pixels from other pixels
    – Predict nearby patches in an image
    – Predict order of frames in a video
    – Predict what you will see as you move
    – Predict physics

Jitendra Malik: "\textbf{Supervision} is the opium of the AI researcher"
Alyosha Efros: "The AI revolution will not be \textit{supervised}"
Yann LeCun: "\textbf{Self-supervised} learning is the cake, \textit{supervised} learning is the icing on the cake, \textbf{reinforcement learning} is the cherry on the cake"
Motivation

• What’s the data we’ve learned from thus far?
• Labeled static datasets
  – Expensive to obtain
  – Doesn’t match how humans learn
• Alternatives
  – Unsupervised learning (no labels)
  – Self-supervised learning (“fake”/emergent labels)
  – Embodied/active learning (agents in environments)
Unsupervised Visual Representation Learning by Context Prediction

Carl Doersch, Alexei Efros and Abhinav Gupta

ICCV 2015
ImageNet + Deep Learning

- Image Retrieval
- Detection (RCNN)
- Segmentation (FCN)
- Depth Estimation
- ...

Do we even need labels?

Context as Supervision
[Collobert & Weston 2008; Mikolov et al. 2013]
Context Prediction for Images

Semantics from a non-semantic task

Relative Position Task

8 possible locations

Patch Embedding

Input

Nearest Neighbors

Note: connects across instances!

What is learned?

<table>
<thead>
<tr>
<th>Input</th>
<th>Ours</th>
<th>ImageNet AlexNet</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Pre-Training for R-CNN

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

Pre-train on relative-position task, w/o labels

VOC 2007 Performance
(pretraining for R-CNN)

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet Labels</td>
<td>54.2</td>
</tr>
<tr>
<td>Ours</td>
<td>46.3</td>
</tr>
<tr>
<td>No Pretraining</td>
<td>40.7</td>
</tr>
</tbody>
</table>

• Test on dataset B
• Option 1: pretrain (unsup) on dataset B
• Option 2: pretrain (sup) on dataset A
Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert
ECCV 2016
Fig. 1: (a) A video imposes a natural temporal structure for visual data. In many cases, one can easily verify whether frames are in the correct temporal order (shuffled or not). Such a simple sequential verification task captures important spatiotemporal signals in videos. We use this task for unsupervised pre-training of a Convolutional Neural Network (CNN). (b) Some examples of the automatically extracted positive and negative tuples used to formulate a classification task for a CNN.

Fig. 2: (a) We sample tuples of frames from high motion windows in a video. We form positive and negative tuples based on whether the three input frames are in the correct temporal order. (b) Our triplet Siamese network architecture has three parallel network stacks with shared weights upto the fc7 layer. Each stack takes a frame as input, and produces a representation at the fc7 layer. The concatenated fc7 representations are used to predict whether the input tuple is in the correct temporal order.
Table 2: Mean classification accuracies over the 3 splits of UCF101 and HMDB51 datasets. We compare different initializations and finetune them for action recognition.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Initialization</th>
<th>Mean Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF101</td>
<td>Random</td>
<td>38.6</td>
</tr>
<tr>
<td></td>
<td>(Ours) Tuple verification</td>
<td>50.2</td>
</tr>
<tr>
<td>HMDB51</td>
<td>Random</td>
<td>13.3</td>
</tr>
<tr>
<td></td>
<td>UCF Supervised</td>
<td>15.2</td>
</tr>
<tr>
<td></td>
<td>(Ours) Tuple verification</td>
<td>18.1</td>
</tr>
</tbody>
</table>

Learning image representations tied to ego-motion

Dinesh Jayaraman and Kristen Grauman

ICCV 2015
The kitten carousel experiment
[Held & Hein, 1963]

Key to perceptual development: self-generated motion + visual feedback

active kitten

passive kitten

Problem with today’s visual learning

**Status quo:** Learn from “disembodied” bag of labeled snapshots.

**Our goal:** Learn in the context of **acting** and **moving** in the world.

Our idea: **Ego-motion ↔ vision**

**Goal:** Teach computer vision system the connection: “how I move” ↔ “how my visual surroundings change”

Ego-motion motor signals + Unlabeled video

Ego-motion $\leftrightarrow$ vision: view prediction

After moving:

Ego-motion ↔ vision for recognition

Learning this connection requires:

➢ Depth, 3D geometry
➢ Semantics
➢ Context

Can be learned without manual labels!

Our approach: unsupervised feature learning using egocentric video + motor signals

Approach idea: Ego-motion equivariance

**Invariant features**: unresponsive to some classes of transformations

\[ z(gx) \approx z(x) \]

**Equivariant features**: predictably responsive to some classes of transformations, through simple mappings (e.g., linear)

\[ z(gx) \approx M_g z(x) \]

Invariance **discards** information; equivariance **organizes** it.

Approach idea: Ego-motion equivariance

**Training data**
Unlabeled video + motor signals

**Equivariant embedding**
organized by ego-motions

Pairs of frames related by similar ego-motion should be related by same feature transformation

---

Approach overview

Our approach: unsupervised feature learning using egocentric video + motor signals

1. Extract training frame pairs from video
2. Learn ego-motion-equivariant image features
3. Train on target recognition task in parallel

Training frame pair mining

Discovery of ego-motion clusters

 yaw change

 forward distance

Ego-motion equivariant feature learning

**Given:**

**Desired:** for all motions $g$ and all images $x$,

$$z_\theta(gx) \approx M_g z_\theta(x)$$

**Unsupervised training**

- $x_i$ (image)
- $g$ (motion)
- $z_\theta(x_i)$ (feature)
- $M_g$ (motion equivariant transformation)
- $\| M_g z_\theta(x_i) - z_\theta(gx_i) \|_2$

**Supervised training**

- $x_k$ (image)
- $y_k$ (class)
- $z_\theta(x_i)$ (feature)
- $M_g$ (motion equivariant transformation)
- $\max L_C(x_k, y_k)$
- $\theta, M_g$ and $W_z$ jointly trained

Ego-motion training pairs

Neural network training

Equivariant embedding

Approach

Scene and object recognition

Next-best view selection

Football field?
Pagoda?
Airport?
Cathedral?
Army base?

Results: Recognition

Learn from **unlabeled car video** (KITTI)

Exploit features for **static scene classification**
(SUN, 397 classes)

Geiger et al, IJRR ’13

Xiao et al, CVPR ’10

Results: Recognition
Do ego-motion equivariant features improve recognition?

Up to 30% accuracy increase over state of the art!

The Curious Robot: Learning Visual Representations via Physical Interactions

Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta

ECCV 2016
Embodied representations

Grasping

Fig. 2. Examples of successful (left) and unsuccessful grasps (right). We use a patch based representation: given an input patch we predict 18-dim vector which represents whether the center location of the patch is graspable at $0^\circ$, $10^\circ$, $\ldots$ $170^\circ$.

### Pushing

**Objects and push action pairs**

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
<th>Initial state</th>
<th>Final state</th>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="initial-state1" alt="Initial state" /></td>
<td><img src="final-state1" alt="Final state" /></td>
<td><img src="initial-state2" alt="Initial state" /></td>
<td><img src="final-state2" alt="Final state" /></td>
<td><img src="initial-state3" alt="Initial state" /></td>
<td><img src="final-state3" alt="Final state" /></td>
</tr>
<tr>
<td><img src="initial-state4" alt="Initial state" /></td>
<td><img src="final-state4" alt="Final state" /></td>
<td><img src="initial-state5" alt="Initial state" /></td>
<td><img src="final-state5" alt="Final state" /></td>
<td><img src="initial-state6" alt="Initial state" /></td>
<td><img src="final-state6" alt="Final state" /></td>
</tr>
</tbody>
</table>

**Fig. 4.** Examples of initial state and final state images taken for the push action. The arrows demonstrate the direction and magnitude of the push action.

Fig. 6. Examples of the data collected by the poking action. On the left we show the object poked, and on the right we show force profiles as observed by the tactile sensor.
Fig. 7. Examples of objects in different poses provided to the embedding network.
Fig. 8. Our shared convolutional architecture for four different tasks.

Fig. 10. The first column corresponds to query image and rest show the retrieval. Note how the network learns that cups and bowls are similar (row 5).
Classification/retrieval performance

Table 1. Classification accuracy on ImageNet Household, UW RGBD and Caltech-256

<table>
<thead>
<tr>
<th>Model</th>
<th>Household</th>
<th>UW RGBD</th>
<th>Caltech-256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root network with random init.</td>
<td>0.250</td>
<td>0.468</td>
<td>0.242</td>
</tr>
<tr>
<td>Root network trained on robot tasks (ours)</td>
<td>0.354</td>
<td>0.693</td>
<td>0.317</td>
</tr>
<tr>
<td>AlexNet trained on ImageNet</td>
<td>0.625</td>
<td>0.820</td>
<td>0.656</td>
</tr>
</tbody>
</table>

Table 2. Image Retrieval with Recall@k metric

<table>
<thead>
<tr>
<th>Model</th>
<th>Instance level</th>
<th>Category level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>k=1</td>
<td>k=5</td>
</tr>
<tr>
<td>Random Network</td>
<td>0.062</td>
<td>0.219</td>
</tr>
<tr>
<td>Our Network</td>
<td>0.720</td>
<td>0.831</td>
</tr>
<tr>
<td>AlexNet</td>
<td>0.686</td>
<td>0.857</td>
</tr>
</tbody>
</table>

Part III: Reinforcement Learning

• Basics: actions, states, rewards, MDP
• Different techniques (Q learning, policy gradients, actor-critic, etc.)
• Example applications
Reinforcement Learning

Agent

Environment

State $s_t$

Reward $r_t$

Next state $s_{t+1}$

Action $a_t$
Cart-Pole Problem

Objective: Balance a pole on top of a movable cart

State: angle, angular speed, position, horizontal velocity

Action: horizontal force applied on the cart

Reward: 1 at each time step if the pole is upright
Atari Games

**Objective:** Complete the game with the highest score

**State:** Raw pixel inputs of the game state

**Action:** Game controls e.g. Left, Right, Up, Down

**Reward:** Score increase/decrease at each time step
Go

Objective: Win the game!

State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise
How can we mathematically formalize the RL problem?

State $s_t$

Reward $r_t$

Next state $s_{t+1}$

Action $a_t$

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung
Markov Decision Process

- Mathematical formulation of the RL problem
- **Markov property**: Current state completely characterises the state of the world

Defined by: \( (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma) \)

- \( \mathcal{S} \) : set of possible states
- \( \mathcal{A} \) : set of possible actions
- \( \mathcal{R} \) : distribution of reward given (state, action) pair
- \( \mathbb{P} \) : transition probability i.e. distribution over next state given (state, action) pair
- \( \gamma \) : discount factor
Markov Decision Process

- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for $t=0$ until done:
  - Agent selects action $a_t$
  - Environment samples reward $r_t \sim R( . | s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P( . | s_t, a_t)$
  - Agent receives reward $r_t$ and next state $s_{t+1}$

- A policy $u$ is a function from $S$ to $A$ that specifies what action to take in each state
- **Objective**: find policy $u^*$ that maximizes cumulative discounted reward: $\sum_{t>0} \gamma^t r_t$
A simple MDP: Grid World

actions = {
1. right
2. left
3. up
4. down
}

Objective: reach one of terminal states (greyed out) in least number of actions

Set a negative "reward" for each transition (e.g. \( r = -1 \))
A simple MDP: Grid World

Random Policy

Optimal Policy
The optimal policy $u^*$

We want to find optimal policy $u^*$ that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?
The optimal policy $u^*$

We want to find optimal policy $u^*$ that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?
Maximize the **expected sum of rewards**!

Formally: $\pi^* = \arg \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | \pi \right]$ with $s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$
Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$

How good is a state?
The value function at state $s$, is the expected cumulative reward from following the policy from state $s$:

$$V^\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, \pi \right]$$

How good is a state-action pair?
The Q-value function at state $s$ and action $a$, is the expected cumulative reward from taking action $a$ in state $s$ and then following the policy:

$$Q^\pi(s, a) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi \right]$$
**Bellman equation**

The optimal Q-value function $Q^*$ is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s, a) = \max_\pi \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

$Q^*$ satisfies the following **Bellman equation**:

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[ r + \gamma \max_{a'} Q^*(s', a') | s, a \right]$$

**Intuition:** if the optimal state-action values for the next time-step $Q^*(s', a')$ are known, then the optimal strategy is to take the action that maximizes the expected value of $r + \gamma Q^*(s', a')$

The optimal policy $u^*$ corresponds to taking the best action in any state as specified by $Q^*$
Solving for the optimal policy: Q-learning

Q-learning: Use a function approximator to estimate the action-value function

\[ Q(s, a; \theta) \approx Q^*(s, a) \]

If the function approximator is a deep neural network => **deep q-learning**!
Q-network Architecture

\[ Q(s, a; \theta) : \]
neural network with weights \( \theta \)

A single feedforward pass to compute Q-values for all actions from the current state \( \Rightarrow \) efficient!

Current state \( s_t \): 84x84x4 stack of last 4 frames
(after RGB->grayscale conversion, downsampling, and cropping)

Last FC layer has 4-d output (if 4 actions), corresponding to \( Q(s_t, a_1), Q(s_t, a_2), Q(s_t, a_3), Q(s_t,a_4) \)

Number of actions between 4-18 depending on Atari game

[Mnih et al. NIPS Workshop 2013; Nature 2015]
Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

1. Initialize replay memory $D$ to capacity $N$
2. Initialize action-value function $Q$ with random weights
3. for episode = 1, $M$ do
4. \hspace{1em} Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
5. \hspace{1em} for $t = 1, T$ do
6. \hspace{2em} With probability $\epsilon$ select a random action $a_t$
7. \hspace{2em} otherwise select $a_t = \max_a Q^* (\phi(s_t), a; \theta)$
8. \hspace{2em} Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
9. \hspace{2em} Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
10. \hspace{2em} Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
11. \hspace{2em} Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
12. \hspace{2em} Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
13. \hspace{2em} Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
14. \hspace{1em} end for
15. \hspace{1em} end for
Putting it together: Deep Q-Learning with Experience Replay

**Algorithm 1** Deep Q-learning with Experience Replay

- Initialize replay memory $\mathcal{D}$ to capacity $N$
- Initialize action-value function $Q$ with random weights

  for episode = 1, $M$ do
  
  - Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  
  for $t = 1, T$ do
    
    - With probability $\epsilon$ select a random action $a_t$
    - otherwise select $a_t = \max_a Q^*(s_t; a; \theta)$
    - Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    - Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    - Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    - Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    
    Set $y_j = \begin{cases} 
    r_j & \text{for terminal } \phi_{j+1} \\
    r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1}
    \end{cases}$
    
    - Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  
  end for

end for

[Mnih et al. NIPS Workshop 2013; Nature 2015]
Putting it together: Deep Q-Learning with Experience Replay

**Algorithm 1** Deep Q-learning with Experience Replay

1. Initialize replay memory $\mathcal{D}$ to capacity $N$
2. Initialize action-value function $Q$ with random weights
3. For episode $= 1, M$ do
   4. Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
5. For $t = 1, T$ do
   6. With probability $\epsilon$ select a random action $a_t$
   7. Otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
   8. Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
   9. Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
   10. Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
   11. Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
   12. Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
   13. Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

Play $M$ episodes (full games)
Putting it together: Deep Q-Learning with Experience Replay

**Algorithm 1** Deep Q-learning with Experience Replay

- Initialize replay memory $D$ to capacity $N$
- Initialize action-value function $Q$ with random weights

```
for episode = 1, M do
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \max_{a} Q^*(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\
        r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for
```

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung
Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights

for episode = 1, $M$ do

Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$

for $t = 1, T$ do

With probability $\epsilon$ select a random action $a_t$
otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$

Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$

Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$

Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$

Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3

end for

end for

For each timestep $t$ of the game
Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights

for episode $= 1, M$ do
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ do
    With probability $\epsilon$, select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

With small probability, select a random action (explore), otherwise select greedy action from current policy
Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

1. Initialize replay memory $\mathcal{D}$ to capacity $N$
2. Initialize action-value function $Q$ with random weights
3. For episode $= 1, M$ do
   4. Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
   5. For $t = 1, T$ do
      6. With probability $\epsilon$ select a random action $a_t$
      7. Otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
      8. Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
      9. Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
      10. Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
      11. Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
      12. Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
      13. Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
   14. End for
4. End for

Take the action $(a_t)$, and observe the reward $r_t$ and next state $s_{t+1}$
Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights

for episode = 1, $M$ do
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ do
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Fei-Fei Li, Andrej Karpathy, Justin Johnson, Serena Yeung
Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity $N$
Initialize action-value function $Q$ with random weights

for episode = 1, $M$ do
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} 
            r_j & \text{for terminal } \phi_{j+1} \\
            r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1}
        \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

[ experience replay: sample a random minibatch of transitions from replay memory and perform a gradient descent step ]
Policy Gradients

What is a problem with Q-learning?
The Q-function can be very complicated!

Example: a robot grasping an object has a very high-dimensional state => hard to learn exact value of every (state, action) pair

But the policy can be much simpler: just close your hand
Can we learn a policy directly, e.g. finding the best policy from a collection of policies?
Formally, let’s define a class of parameterized policies: \( \mathcal{P} = \{\pi_\theta, \theta \in \mathbb{R}^m\} \)

For each policy, define its value:

\[
J(\theta) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | \pi_\theta \right]
\]

We want to find the optimal policy \( \theta^* = \arg \max_{\theta} J(\theta) \)

How can we do this?

**Gradient ascent on policy parameters!**
REINFORCE Algorithm (orig. Williams 1992)

Gradient estimator: \[ \nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t) \]

**Interpretation:**
- If \( r(r) \) is high, push up the probabilities of the actions seen
- If \( r(r) \) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. **But in expectation, it averages out!**
Policy Gradients
Policy Gradients

- Objective: $\sum_i A_i \log p(y_i|x_i)$
- $x_i = \text{state}$
- $y_i = \text{sampled action}$
- $A_i = \text{“advantage” e.g. +1/-1 for win/lose in simplest version, or discounted, or improvement over “baseline”}$
Policy Gradients vs Q-Learning

- Policy gradients suffers from high variance and instability; might want to make gradients smaller (e.g. relative to a baseline)
- Policy gradients can handle continuous action spaces (Gaussian policy)
- Estimating exact value of state-action pairs vs choosing what actions to take (value not important)
- Step-by-step (did I correctly estimate the reward at this time) vs delayed feedback (run policy and wait until game terminates)
Actor-Critic Algorithm

We can combine Policy Gradients and Q-learning by training both an **actor** (the policy) and a **critic** (the Q-function).

- The actor decides which action to take, and the critic tells the actor how good its action was and how it should adjust.
- Also alleviates the task of the critic as it only has to learn the values of (state, action) pairs generated by the policy.
- Can also incorporate Q-learning tricks e.g. experience replay.
- **Remark**: we can define by the **advantage function** how much an action was better than expected.

\[
A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)
\]
RL for object detection

Figure 1. A sequence of actions taken by the proposed algorithm to localize a cow. The algorithm attends regions and decides how to transform the bounding box to progressively localize the object.
RL for object detection

Figure 2. Illustration of the actions in the proposed MDP, giving 4 degrees of freedom to the agent for transforming boxes.

\[ R_{a}(s, s') = \text{sign} (\text{IoU}(b', g) - \text{IoU}(b, g)) \]

\[ R_{\omega}(s, s') = \begin{cases} 
+\eta & \text{if } \text{IoU}(b, g) \geq \tau \\
-\eta & \text{otherwise}
\end{cases} \]
**RL for navigation**

![Diagram](image)

**Fig. 1.** The goal of our deep reinforcement learning model is to navigate towards a visual target with a minimum number of steps. Our model takes the current observation and the image of the target as input and generates an action in the 3D environment as the output. Our model learns to navigate to different targets in a scene without re-training.

RL for navigation

Figure 1: Our goal is to use scene priors to improve navigation in unseen scenes and towards novel objects. (a) There is no mug in the field of view of the agent, but the likely location for finding a mug is the cabinet near the coffee machine. (b) The agent has not seen a mango before, but it infers that the most likely location for finding a mango is the fridge since similar objects such as apple appear there as well. The most likely locations are shown with the orange box.

Figure 2: **Overview of the architecture.** Our model to incorporate semantic knowledge into semantic navigation. Specifically, we learn a policy network that decides an action based on the visual features of the current state, the semantic target category feature and the features extracted from the knowledge graph. We extract features from the parts of the knowledge graph that are activated.

RL for question-answering

Figure 1: Embodied Question Answering – EmbodiedQA– tasks agents with navigating rich 3D environments in order to answer questions. These agents must jointly learn language understanding, visual reasoning, and goal-driven navigation to succeed.

Das et al., “Embodied Question Answering”, CVPR 2018
Figure 4: Our PACMAN navigator decomposes navigation into a planner and a controller. The planner selects actions and the controller executes these actions a variable number of times. This enables the planner to operate on shorter timescales, strengthening gradient flows.

Das et al., “Embodied Question Answering”, CVPR 2018
Part IV: Generation

- Motivation and taxonomy of methods
- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)
- Applications and variants of GANs
- Dealing with sparse data, progressive training
Generative Models

Training data $\sim p_{data}(x)$

Generated samples $\sim p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$
Generative Models

Addresses density estimation, a core problem in unsupervised learning

Several flavors:
- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it
Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.

- Generative models can be used to enhance training datasets with diverse synthetic data
- Generative models of time-series data can be used for simulation

Adapted from Serena Young
Taxonomy of Generative Models

Generative models

Explicit density

- Tractable density
  - Fully Visible Belief Nets
    - NADE
    - MADE
    - PixelRNN/CNN

Implicit density

- Approximate density
  - Variational
    - Variational Autoencoder

- Markov Chain
  - Markov Chain
  - Boltzmann Machine

Direct

GAN

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.
PixelRNN and PixelCNN
Explicit density model
Use chain rule to decompose likelihood of an image $x$ into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

- Likelihood of image $x$
- Probability of i’th pixel value given all previous pixels
- Will need to define ordering of “previous pixels”
- Complex distribution over pixel values $\Rightarrow$ Express using a neural network!

Then maximize likelihood of training data

Serena Young
PixelRNN

[van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow!
PixelCNN

[van der Oord et al. 2016]

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

Training: maximize likelihood of training images

$$p(x) = \prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1})$$

Training is faster than PixelRNN (can parallelize convolutions since context region values known from training images)

Generation must still proceed sequentially => still slow

Serena Young
Variational Autoencoders (VAEs)
So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

\[ p_{\theta}(x) = \prod_{i=1}^{n} p_{\theta}(x_i|x_1, \ldots, x_{i-1}) \]

VAEs define intractable density function with latent \( z \):

\[ p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz \]

Cannot optimize directly, derive and optimize lower bound on likelihood instead.
Some background: Autoencoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

- $z$ usually smaller than $x$ (dimensionality reduction)
- Originally: Linear + nonlinearity (sigmoid)
- Later: Deep, fully-connected
- Later: ReLU CNN

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data
Some background: Autoencoders

How to learn this feature representation?
Train such that features can be used to reconstruct original data
“Autoencoding” - encoding itself

Reconstructed input data \( \hat{x} \) → Decoder
Features \( z \) → Encoder
Input data \( x \)

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN (upconv)
Some background: Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function:
\[ \| x - \hat{x} \|^2 \]

Doesn't use labels!

Encoder: 4-layer conv
Decoder: 4-layer upconv

Serena Young
Some background: Autoencoders

Encoder can be used to initialize a supervised model

Encoder

Input data

Features

Loss function (Softmax, etc)

Predicted Label

Classifier

Fine-tune encoder jointly with classifier

Train for final task (sometimes with small data)

bird  plane

dog  deer  truck

Serena Young
Some background: Autoencoders

Autoencoders can reconstruct data, and can learn features to initialize a supervised model.

Features capture factors of variation in training data. Can we generate new images from an autoencoder?
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data \( \{x^{(i)}\}_{i=1}^{N} \) is generated from underlying unobserved (latent) representation \( z \)

Sample from true conditional \( p_{\theta^*}(x \mid z^{(i)}) \)

Sample from true prior \( p_{\theta^*}(z) \)

**Intuition** (remember from autoencoders!): \( x \) is an image, \( z \) is latent factors used to generate \( x \): attributes, orientation, etc.

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model.

How should we represent this model?

Choose prior $p(z)$ to be simple, e.g. Gaussian.

Conditional $p(x|z)$ is complex (generates image) => represent with neural network

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model.

How to train the model?

Learn model parameters to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

Now with latent $z$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Adapted from Serena Young
Variational Autoencoders

We want to estimate the true parameters $\theta^*$ of this generative model.

How to train the model?

Learn model parameters to maximize likelihood of training data

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

Q: What is the problem with this?

Intractable!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders: Intractability

Data likelihood: \[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

Simple Gaussian prior

Intractable to compute \( p(x|z) \) for every \( z \)!

Decoder neural network

Posterior density also intractable: \[ p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x) \]

Intractable data likelihood

- Solution: In addition to decoder network modeling \( p_\theta(x|z) \), define additional encoder network \( q_\phi(z|x) \) that approximates \( p_\theta(z|x) \)

- This allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize – omitted, see hidden slides

Adapted from Serena Young

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Since we’re modeling probabilistic generation of data, encoder and decoder networks are probabilistic.

Encoder network
\[ q_\phi(z|x) \]
(parameters \( \phi \))

Decoder network
\[ p_\theta(x|z) \]
(parameters \( \theta \))

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Serena Young
Variational Autoencoders

Since we’re modeling probabilistic generation of data, encoder and decoder networks are probabilistic.

Sample $z$ from $z | x \sim \mathcal{N}(\mu_z | x, \Sigma_z | x)$

Encoder network
$q_\phi(z | x)$
(parameters $\phi$)

$\mathcal{X}$

Decoder network
$p_\theta(x | z)$
(parameters $\theta$)

Sample $x | z \sim \mathcal{N}(\mu_x | z, \Sigma_x | z)$

$\mathcal{Z}$

Encoder and decoder networks also called “recognition”/“inference” and “generation” networks

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$E_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_{\theta}(z))$$

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!
VAEs: Generating Data

Use decoder network. Now sample $z$ from prior!

Data manifold for 2-d $z$

Sample $z$ from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
VAEs: Generating Data

Diagonal prior on $z$ => independent latent variables

Different dimensions of $z$ encode interpretable factors of variation

Also good feature representation that can be computed using $q_\phi(z|x)$!

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014
VAEs: Generating Data

32x32 CIFAR-10

Labeled Faces in the Wild

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a lower bound

**Pros:**
- Principled approach to generative models
- Allows inference of $q(z|x)$, can be useful feature representation for other tasks

**Cons:**
- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Adapted from Serena Young
So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, \ldots, x_{i-1})$$

VAEs define intractable density function with latent $z$:

$$p_\theta(x) = \int p_\theta(z)p_\theta(x | z)dz$$

Cannot optimize directly, derive and optimize lower bound on likelihood instead.

What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don’t work with any explicit density function!

Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game.
Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?
Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

Input: Random noise

Output: Sample from training distribution

Serena Young
Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

---

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Adversarial Networks Framework

Discriminator
Real vs. Fake

Generator
\( x \sim G(z) \)

Differentiable function \( D \)

Input noise \( Z \)

Differentiable function \( G \)

\( x \) sampled from data

\( D \) tries to output 1

\( D \) tries to output 0
Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in minimax game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]

- Discriminator ($\theta_d$) wants to maximize objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- Generator ($\theta_g$) wants to minimize objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Serena Young
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:
1. **Gradient ascent** on discriminator
   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$
2. **Gradient descent** on generator
   $$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z)))$$
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator
   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator
   $$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Gradient signal dominated by region where sample is already good.

Serena Young
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead**: **Gradient ascent** on generator, different objective

   $$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.
Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Putting it together: GAN training algorithm

for number of training iterations do
  for k steps do
    • Sample minibatch of \( m \) noise samples \( \{ z^{(1)}, \ldots, z^{(m)} \} \) from noise prior \( p_g(z) \).
    • Sample minibatch of \( m \) examples \( \{ x^{(1)}, \ldots, x^{(m)} \} \) from data generating distribution \( p_{\text{data}}(x) \).
    • Update the discriminator by ascending its stochastic gradient:
      \[
      \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]
      \]
  end for
  • Sample minibatch of \( m \) noise samples \( \{ z^{(1)}, \ldots, z^{(m)} \} \) from noise prior \( p_g(z) \).
  • Update the generator by ascending its stochastic gradient (improved objective):
    \[
    \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))
    \]
end for
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

---

After training, use generator network to generate new images
GAN training is challenging

• Vanishing gradient – when discriminator is very good
• Mode collapse – too little diversity in the samples generated
• Lack of convergence because hard to reach Nash equilibrium
• Loss metric doesn’t always correspond to image quality; Frechet Inception Distance (FID) is a decent choice
### Alternative loss functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Paper Link</th>
<th>Value Function</th>
</tr>
</thead>
</table>
| GAN         | Arxiv      | $L_{GAN}^{D} = E[\log(D(x))] + E[\log(1 - D(G(z)))]$
|             |            | $L_{GAN}^{G} = E[\log(D(G(z)))]$ |
| LSGAN       | Arxiv      | $L_{LSGAN}^{D} = E[(D(x) - 1)^2] + E[D(G(z))^2]$ |
|             |            | $L_{LSGAN}^{G} = E[(D(G(z)) - 1)^2]$ |
| WGAN        | Arxiv      | $L_{WGAN}^{D} = E[D(x)] - E[D(G(z))]$
|             |            | $L_{WGAN}^{G} = E[D(G(z))]$ |
|             |            | $W_{D} \leftarrow \text{clip}_{by\_value}(W_{D}, -0.01, 0.01)$ |
| WGAN_GP     | Arxiv      | $L_{WGAN\_GP}^{D} = L_{WGAN}^{D} + \lambda E[(V_D(ax - (1 - aG(z))) - 1)^2]$ |
|             |            | $L_{WGAN\_GP}^{G} = L_{WGAN}^{G}$ |
| DRAGAN      | Arxiv      | $L_{DRAGAN}^{D} = L_{GAN}^{D} + \lambda E[(V_D(ax - (1 - ax_p)) - 1)^2]$ |
|             |            | $L_{DRAGAN}^{G} = L_{GAN}^{G}$ |
| CGAN        | Arxiv      | $L_{CGAN}^{D} = E[\log(D(x,c))] + E[\log(1 - D(G(z),c)))]$
|             |            | $L_{CGAN}^{G} = E[\log(D(G(z),c))]$ |
| InfoGAN     | Arxiv      | $L_{InfoGAN}^{D} = L_{GAN}^{D} - \lambda I_{I}(c,c')$
|             |            | $L_{InfoGAN}^{G} = L_{GAN}^{G} - \lambda I_{I}(c,c')$ |
| ACGAN       | Arxiv      | $L_{ACGAN}^{D} = L_{GAN}^{D} + E[P(\text{class} = c|x)] + E[P(\text{class} = c|G(z))]$
|             |            | $L_{ACGAN}^{G} = L_{GAN}^{G} + E[P(\text{class} = c|G(z))]$ |
| EBGAN       | Arxiv      | $L_{EBGAN}^{D} = D_{AE}(x) + \max(0, m - D_{AE}(G(z)))$
|             |            | $L_{EBGAN}^{G} = D_{AE}(G(z)) + \lambda \cdot PT$ |
| BEGAN       | Arxiv      | $L_{BEGAN}^{D} = D_{AE}(x) - k_{t}D_{AE}(G(z))$
|             |            | $L_{BEGAN}^{G} = D_{AE}(G(z))$
|             |            | $k_{t+1} = k_{t} + \lambda(y_{D_{AE}}(x) - D_{AE}(G(z)))$ |

[https://github.com/hwalsuklee/tensorflow-generative-model-collections](https://github.com/hwalsuklee/tensorflow-generative-model-collections)
[https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490](https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490)
Tips and tricks

• Use batchnorm, ReLU
• Regularize norm of gradients
• Use one of the new loss functions
• Add noise to inputs or labels
• Append image similarity to avoid mode collapse
• Use labels when available (CGAN)
• …
Celebrities Who Never Existed

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Creative Adversarial Networks

CAN: Top ranked by human subjects

(Elgammal et al., 2017)
Conditional GANs

https://medium.com/@jonathan_hui/gan-cgan-infogan-using-labels-to-improve-gan-8ba4de5f9c3d
GANs

$G$: generate fake samples that can fool $D$

$D$: classify fake samples vs. real images

[Goodfellow et al. 2014]

Jun-Yan Zhu
Conditional GANs

$\text{G}(x)$

real or fake pair?
Edges → Images

Edges from [Xie & Tu, 2015]

Pix2pix / CycleGAN
$\text{Sketches} \rightarrow \text{Images}$

Trained on Edges → Images

Data from [Eitz, Hays, Alexa, 2012]

Pix2pix / CycleGAN
#edges2cats  [Christopher Hesse]

@gods_tail

Ivy Tasi @ivymyt

https://affinelayer.com/pixsrv/

Pix2pix / CycleGAN
Paired

\[ x_i \]

\[ y_i \]

Unpaired

\[ X \]

\[ Y \]

Jun-Yan Zhu
Cycle Consistency

Discriminator $D_Y$: $L_{GAN}(G(x), y)$
Real zebras vs. generated zebras

Zhu et al., “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”, ICCV 2017
Cycle Consistency

Discriminator $D_X: L_{GAN}(F(y), x)$
Real horses vs. generated horses

Discriminator $D_Y: L_{GAN}(G(x), y)$
Real zebras vs. generated zebras

Zhu et al., “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”, ICCV 2017
Cycle Consistency

Forward cycle loss: $\|F(G(x)) - x\|_1$

Zhu et al., “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”, ICCV 2017
Cycle Consistency

Forward cycle loss: $\|F(G(x)) - x\|_1$

Zhu et al., “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”, ICCV 2017

Helps cope with mode collapse
Training Details: Objective

\[ \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G'(x))], \]

\[ \mathcal{L}_{\text{cyc}}(G, F') = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \]

\[ \mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F'), \]

\[ G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y). \]

Zhu et al., “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”, ICCV 2017
Pix2pix / CycleGAN
StarGAN

Choi et al., “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation”, CVPR 2018
SinGAN

Single training image \rightarrow Random samples from a single image

Paint to image \quad Editing \quad Harmonization \quad Super-resolution \quad Animation

Training Image

Input

Output

Generating with little data for ads

- Faces are persuasive and carry meaning/sentiment

- We learn to generate faces appropriate for each ad category

- Because our data is so diverse yet limited in count, standard approaches that directly model pixel distributions don’t work well

Thomas and Kovashka, BMVC 2018
Generating with little data for ads

- Instead we model the distribution over *attributes* for each category (e.g. domestic violence ads contain “black eye”, beauty contains “red lips”)
- Generate an image with the attributes of an ad class
- Model attributes w/ help from external large dataset

**Diagram**

- **Input**: 128x128x3
- **Encoder**: 32x32x16 → 8x8x64 → 2x2x256
- **Sampling**: 100 (μ), 512, 1024
- **Decoder**: 8x8x64 → 32x32x16 → 128x128x3

**Embedding**

- **Latent (100-D)**
  - Captures non-semantic appearance properties (colors, etc.)

**Externally Enforced Semantics**

- **Facial Attributes (40-D)**
  - *Facial attributes*: Attractive, Baggy eyes, Big lips, Bushy eyebrows, Eyeglasses, Gray hair, Makeup, Male, Pale skin, Rosy cheeks, etc.

- **Facial Expressions (10-D)**
  - *Facial expressions*: Anger, Contempt, Disgust, Fear, Happy, Neutral, Sad, Surprise

  + Valence and Arousal scores

**Numbers**

- 150
- 64x64x8, 16x16x32, 4x4x128, 100 (μ), 100 (σ), 16x16x32, 8x8x64, 32x32x16, 128x128x3

---

Thomas and Kovashka, BMVC 2018
Generating with little data for ads

- Reconstruction
- Alcohol
- Beauty
- Clothing
- D.V.
- Safety
- Soda

Ours

Conditional

Latent

StarGAN (T)
StarGAN (C)

Transform

Original Face

Thomas and Kovashka, BMVC 2018
Stagewise generation

Stagewise generation

Johnson et al., “Image Generation from Scene Graphs”, CVPR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

There’s waves everywhere!

But where’s the shore?

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Latent $\rightarrow$ $4 \times 4$ $\rightarrow$ $64 \times 64$ $\rightarrow$ $1024 \times 1024$ $\rightarrow$ $1024 \times 1024$ $\rightarrow$ $64 \times 64$ $\rightarrow$ $4 \times 4$ $\rightarrow$ Real or fake

There it is!

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Progressive generation

Karras et al., “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018
Part V: Ethics (Politics, Privacy, Bias)

- Politics and deep fakes
  - Examples from DARPA
  - Detection methods
- Privacy
  - GANs for anonymity in the cloud
  - What can be reconstructed
- Security
  - Adversarial perturbations
- Bias
  - What models show
  - How to cope
- AI for the people
“Deepfakes”
You can be anyone you want...
Detection methods

FaceForensics++: Learning to Detect Manipulated Facial Images

Andreas Rössler¹  Davide Cozzolino²  Luisa Verdoliva²  Christian Riess³
Justus Thies¹  Matthias Nießner¹
¹Technical University of Munich  ²University Federico II of Naples  ³University of Erlangen-Nuremberg

Figure 1: FaceForensics++ is a dataset of facial forgeries that enables researchers to train deep-learning-based approaches in a supervised fashion. The dataset contains manipulations created with four state-of-the-art methods, namely, Face2Face, FaceSwap, DeepFakes, and NeuralTextures.

“We describe a forensic technique that models facial expressions and movements that typify an individual’s speaking pattern. Although not visually apparent, these correlations are often violated by the nature of how deep-fake videos are created and can, therefore, be used for authentication.

Agarwal et al., “Protecting World Leaders Against Deep Fakes”, CVPR Workshops, 2019
Incredible Pace of Synthetic Media Generation

Legend:
Single modality
Multi-modality

2018
May June July Aug Sep Nov Dec

2019
Jan Feb Mar Apr May

Interactive audio
Attribute-guided face generation
Unsupervised text generation
Fake resumes
Fake dating profiles
Video dialog replacement
Scenes from sketches
Fake rental ads

ENTIRE GUEST SUITE
Luxury Condo 3 Bed + 3 Bath
Port Melbourne

- 8 guests
- 3 bedrooms
- 4 beds
- 2 baths

Bathroom (with seating for 2 more people), basin and eclectic French garden and kitchen. 24/7 carpeted charc. Laundromemberly: More balcony – Garden – Metro, Liverpool Street (15 min walk) Walking distance to Wyckofferdon
State of the Art Detection is Statistically Based, Narrow, or Both

**Audio: ASVspoof**

- Hand-crafted Features
- Neural Networks
- Temporal Neural Networks
- Fusion

*(Lavrentyeva et al. 2017)*

**Text: GLTR**

- Input text
- Word Prediction Probability

NY Times:

> Here's a pop quiz for you

AI:

> I've been a gamer for over ten years.

AI methods choose more predictable next-words than humans, statistically

*(MIT-IBM Watson AI lab, HarvardNLP 2019)*

**Image/Video: DARPA MediFor**

- Noise Fingerprint Network
- Manipulation detection heatmap

*(MediFor: USC/ISI, Univ. Naples 2019)*
DARPA

Expected Threats

Targeted Personal Attacks
Peele 2017

AI Multimedia Algorithms

Generated Events at Scale

AI Multimedia Algorithms

Ransomfake concept: Identity Attacks as a service (IAaaS)
Bricman 2019

AI Multimedia Algorithms

Forged Evidence

Identity Attacks

Examples of possible fakes:
- Substance abuse
- Foreign contacts
- Compromising events
- Social media postings
- Financial inconsistencies
- Forging identity

On a rainy spring day, a vast, violent group gathered in front of the US Capitol to protest recent cuts in Social Security.

Text
Video & Audio
Image

Believable fake events

Highly realistic video

Undermines key individuals and organizations
GANs for Privacy (Action Detection)

Ren et al., “Learning to Anonymize Faces for Privacy Preserving Action Detection”, ECCV 2018
Adversarial Attacks

https://bair.berkeley.edu/blog/2017/12/30/yolo-attack/
Adversarial Attacks

Adversarial Attacks

Tom Goldstein [https://www.cs.umd.edu/~tomg/projects/invisible/]
Adversarial Attacks

This object-recognition dataset stumped the world’s best computer vision models

Objects are posed in varied positions and shot at odd angles to spur new AI techniques.

Bias in the Vision and Language of Artificial Intelligence
What do you see?

- Bananas
- Stickers
- Dole Bananas
- Bananas at a store
- Bananas on shelves
- Bunches of bananas
- Bananas with stickers on them
- Bunches of bananas with stickers on them on shelves in a store

...We don’t tend to say **Yellow Bananas**
What do you see?

Green Bananas

Unripe Bananas
What do you see?

**Ripe Bananas**

**Bananas with spots**

**Bananas good for banana bread**
What do you see?

Yellow Bananas

*Yellow* is prototypical for bananas
Prototype Theory

One purpose of categorization is to reduce the infinite differences among stimuli to behaviourally and cognitively usable proportions.

There may be some central, prototypical notions of items that arise from stored typical properties for an object category (Rosch, 1975).

May also store exemplars (Wu & Barsalou, 2009)

Fruit

Bananas
“Basic Level”

Unripe Bananas, Cavendish Bananas
A man and his son are in a terrible accident and are rushed to the hospital in critical care.

The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

How could this be?

Margaret Mitchell
A man and his son are in a terrible accident and are rushed to the hospital in critical care.

The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

How could this be?
A man and his son are in a terrible accident and are rushed to the hospital in critical care.

The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

*How could this be?*
The majority of test subjects overlooked the possibility that the doctor is a she - including men, women, and self-described feminists.

Wapman & Belle, Boston University
Human Reporting Bias

The frequency with which people write about actions, outcomes, or properties is not a reflection of real-world frequencies or the degree to which a property is characteristic of a class of individuals.
Bias in Language

**Extreme she occupations**
1. homemaker
2. nurse
3. receptionist
4. librarian
5. socialite
6. hairdresser
7. nanny
8. bookkeeper
9. stylist
10. housekeeper
11. interior designer
12. guidance counselor

**Extreme he occupations**
1. maestro
2. skipper
3. protege
4. philosopher
5. captain
6. architect
7. financier
8. warrior
9. broadcaster
10. magician
11. fighter pilot
12. boss

Figure 1: The most extreme occupations as projected on to the *she–he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

**Gender stereotype she-he analogies.**
- sewing-carpentry
- nurse-surgeon
- blond-burly
- giggle-chuckle
- sassy-snappy
- volleyball-football
- register-nurse-physician
- interior-designer-architect
- feminism-conservatism
- vocalist-guitarist
- diva-superstar
- cupcakes-pizzas
- housewife-shopkeeper
- softball-baseball
- cosmetics-pharmaceuticals
- petite-lanky
- charming-affable
- hairdresser-barber

**Gender appropriate she-he analogies.**
- queen-king
- waitress-waiter
- ovarian cancer-prostate cancer
- mother-father
- convent-monastery

Figure 2: **Analogy examples.** Examples of automatically generated analogies for the pair *she-he* using the procedure described in text. For example, the first analogy is interpreted as *she:sewing :: he:carpentry* in the original w2vNEWS embedding. Each automatically generated analogy is evaluated by 10 crowd-workers as to whether or not it reflects gender stereotype. Top: illustrative gender stereotypic analogies automatically generated from w2vNEWS, as rated by at least 5 of the 10 crowd-workers. Bottom: illustrative generated gender-appropriate analogies.
Bias in Language

Adjectives

Or type your own words...
doctor

he (158)  

she (42)

http://wordbias.umiacs.umd.edu/
Fig. 1: Examples where our proposed model (Equalizer) corrects bias in image captions. The overlaid heatmap indicates which image regions are most important for predicting the gender word. On the left, the baseline predicts gender incorrectly, presumably because it looks at the laptop (not the person). On the right, the baseline predicts the gender correctly but it does not look at the person when predicting gender and is thus not acceptable. In contrast, our model predicts the correct gender word and correctly considers the person when predicting gender.
Bias in Vision

Figure 2. In our bias mitigation approach, we learn a task-specific model with an adversarial loss that removes features corresponding to a protected variable from an intermediate representation in the model – here we illustrate our pipeline to visualize the removal of features in image space through an auto-encoder network.

Figure 3. Images after adversarial removal of gender when applied to the image space. The objective was to preserve information about objects and verbs, e.g. scissors, banana (COCO) or vaulting, lifting (imSitu) while removing gender correlated features.

Training data are collected and annotated → Model is trained → Media are filtered, ranked, aggregated, or generated → People see output
Biases in Data
Biases in Data

**Selection Bias:** Selection does not reflect a random sample

Map of Amazon Mechanical Turk Workers

© 2013–2016 Michael Yoshitaka Erlewine and Hadas Kotek

Margaret Mitchell
Out-group homogeneity bias: Tendency to see outgroup members as more alike than ingroup members
It's possible that you have an appropriate amount of data for every group you can think of but that some groups are represented less positively than others.
Annotations in your dataset will reflect the worldviews of your annotators.
Predicting Policing

- Algorithms identify potential crime hot-spots
- Based on where crime is previously reported, not where it is known to have occurred
- Predicts future events from past

CREDIT
Smithsonian, Artificial Intelligence Is Now Used to Predict Crime. But Is It Biased? 2018
Predicting Sentencing

- Prater (who is white) rated **low risk** after shoplifting, despite two armed robberies; one attempted armed robbery.

- Borden (who is black) rated **high risk** after she and a friend took (but returned before police arrived) a bike and scooter sitting outside.

- Two years later, Borden has not been charged with any new crimes. Prater serving 8-year prison term for grand theft.

**CREDIT**
Predicting Criminality

Israeli startup, Faception

“Faception is first-to-technology and first-to-market with proprietary computer vision and machine learning technology for profiling people and revealing their personality based only on their facial image.”

Offering specialized engines for recognizing “High IQ”, “White-Collar Offender”, “Pedophile”, and “Terrorist” from a face image.

Main clients are in homeland security and public safety.
Predicting Criminality

“Automated Inference on Criminality using Face Images” Wu and Zhang, 2016. arXiv

1,856 closely cropped images of faces; Includes “wanted suspect” ID pictures from specific regions.

“[..] angle θ from nose tip to two mouth corners is on average 19.6% smaller for criminals than for non-criminals …”

See our longer piece on Medium, “Physiognomy’s New Clothes”
It’s up to us to influence how AI evolves.
Get paper award, 15 minutes of fame for ___thing___

Find local optimum given task, data, etc

Get paper published, product launched

Positive outcomes for humans and their environment.

Today

Short-term

Longer-term
human

augment

intellect.

Fei-Fei Li
AI must incorporate more of the versatility, nuance, and depth of the human intellect.
AI should augment human skills, not replace them.

AI must incorporate more of the versatility, nuance, and depth of the human intellect.

Fei-Fei Li
The development of AI should be guided by a concern for its impact on human society. AI should augment human skills, not replace them. AI must incorporate more of the versatility, nuance, and depth of the human intellect.
AI must incorporate more of the versatility, nuance, and depth of the human intellect.
From academic backwater to center of attention in 5 years

What happened?

The Deep Learning Revolution
Hello,

I am hurt

Hello, hurt! 😊

The limits of chatbot conversation
Dog’s Owner (Angry)

Couch (Torn Up)

Upset About damage

Frustrated with dog

Dog (Guilty)

Responsible for damage

Fei-Fei Li
Dog’s Owner (Angry)

Couch (Torn Up)

Frustated with dog

Context

Situational Awareness

Prior Knowledge

Responsible for damage

Dog (Guilty)

Upset

About damage

Fei-Fei Li
Curiosity-based Learning

- A baby’s learning is exploratory, curiosity-driven, multi-modal, active and social.
- Can we model this process and apply it in machines?

Mrowca, Haber, Fei-Fei & Yamins, *CogSci*, 2018
“Thinking slow”
Commonsense knowledge and reasoning

- Reasoning requires combining previously acquired knowledge to address new tasks
- Can a neural network reason more like a human?

He served chicken.

Not like serving a tennis ball
He probably cooked it first!

The trophy wouldn’t fit in the suitcase because it was too big.

The trophy?
The suitcase?

Hudson and Manning, 2018
AI should **augment** human skills, not replace them.
~50% current work activities can be theoretically automated now

100% current work activities can be potentially enhanced by intelligent technology
Enhancing human care with intelligent systems
Hospital-Acquired Infections

99,000 Deaths
Annually

Unmonitored Elderly Fall Injuries

$36.4 Billion
Annually

A. Houser, W. Fox-Grage & K. Ujvari, AARP Public Policy Institute, 2012


Annual Review of Medicine 2012
From: Inconsistent hand hygiene

To: Intelligent monitors placed throughout hospitals

A. Haque, E. Peng, A. Luo, A. Alahi, S. Yeung & L. Fei-Fei, *ECCV, 2016*
From: Ineffective wearables, lack of human caretakers

To: Intelligent monitors placed throughout senior living homes
Giving human specialists more time

Fei-Fei Li
Lowers costs

Improves safety and outcomes

Reduces burden on human caregivers
An algorithm for automating simple radiology analysis

More time for human specialists to do what they do best

Z. Li, C. Wang, M. Han, Y. Xue, W. Wei, Li-J. Li, L. Fei-Fei, *CVPR, 2018*