Self-Supervised Learning

Yann LeCun
New York University
Facebook AI Research
http://yann.lecun.com
Supervised Learning works but requires many labeled samples

- Training a machine by showing examples instead of programming it
- When the output is wrong, tweak the parameters of the machine

Works well for:
- Speech → words
- Image → categories
- Portrait → name
- Photo → caption
- Text → topic
- ….
Reinforcement Learning: Model-Free RL works great for games.

- **57 Atari games**: takes **83 hours equivalent real-time** (18 million frames) to reach a performance that humans reach in 15 minutes of play.
  - [Hessel ArXiv:1710.02298]
- **Elf OpenGo v2**: 20 million self-play games. (2000 GPU for 14 days)
  - [Tian arXiv:1902.04522]
- **StarCraft**: AlphaStar 200 years of equivalent real-time play
  - [Vinyals blog post 2019]
- They all use ConvNets and a few other architectural concepts.
But RL Requires too many trials in the real world

- Pure RL requires too many trials to learn anything
  - it’s OK in a game
  - it’s not OK in the real world
- RL works in simple virtual world that you can run faster than real-time on many machines in parallel.

- Anything you do in the real world can kill you
- You can’t run the real world faster than real time
- simple cells detect local features
- complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.

Y. LeCun

Convolutional Network Architecture [LeCun et al. NIPS 1989]
Convolutional Network (LeNet5, vintage 1990)

Filters-tanh → pooling → filters-tanh → pooling → filters-tanh
ConvNets can recognize multiple objects

- All layers are convolutional
- Networks performs simultaneous segmentation and recognition
Face & Pedestrian Detection with ConvNets (1993-2005)

[Osadchy, Miller LeCun JMLR 2007], [Kavukcuoglu et al. NIPS 2010] [Sermanet et al. CVPR 2013]
Semantic Segmentation with ConvNets (33 categories)

[Farabet et al. ICML 2012]
Deep ConvNets for Object Recognition (on GPU)

- AlexNet [Krizhevsky et al. NIPS 2012], OverFeat [Sermanet et al. 2013]
- 1 to 10 billion connections, 10 million to 1 billion parameters, 8 to 20 layers.

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic Fox (1.0); Eskimo Dog (0.6); White Wolf (0.4); Siberian Husky (0.4)
Deep ConvNets (depth inflation)

VGG
[Simonyan 2013]

GoogLeNet
[Szegedy 2014]

ResNet
[He et al. 2015]

DenseNet
[Huang et al. 2017]
Progress in Computer Vision

[He 2017]
Mask R-CNN: instance segmentation

- [He, Gkioxari, Dollar, Girshick, arXiv:1703.06870]
- ConvNet produces an object mask for each region of interest
- Combined ventral and dorsal pathways

<table>
<thead>
<tr>
<th></th>
<th>backbone</th>
<th>AP</th>
<th>AP_{50}</th>
<th>AP_{75}</th>
<th>AP_{S}</th>
<th>AP_{M}</th>
<th>AP_{L}</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNC [7]</td>
<td>ResNet-101-C4</td>
<td>24.6</td>
<td>44.3</td>
<td>24.8</td>
<td>4.7</td>
<td>25.9</td>
<td>43.6</td>
</tr>
<tr>
<td>FCIS [20] +OHEM</td>
<td>ResNet-101-C5-dilated</td>
<td>29.2</td>
<td>49.5</td>
<td>-</td>
<td>7.1</td>
<td>31.3</td>
<td>50.0</td>
</tr>
<tr>
<td>FCIS+++ [20] +OHEM</td>
<td>ResNet-101-C5-dilated</td>
<td>33.6</td>
<td>54.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNet-101-C4</td>
<td>33.1</td>
<td>54.9</td>
<td>34.8</td>
<td>12.1</td>
<td>35.6</td>
<td>51.1</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNet-101-FPN</td>
<td>35.7</td>
<td>58.0</td>
<td>37.8</td>
<td>15.5</td>
<td>38.1</td>
<td>52.4</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNeXt-101-FPN</td>
<td>37.1</td>
<td>60.0</td>
<td>39.4</td>
<td>16.9</td>
<td>39.9</td>
<td>53.5</td>
</tr>
</tbody>
</table>
RetinaNet, feature pyramid network

- One-pass object detection
- [Lin et al. ArXiv:1708.02002]
Mask-RCNN Results on COCO dataset

Individual objects are segmented.
Mask R-CNN Results on COCO test set
Panoptic Feature Pyramid Network

Segments and recognizes object instances and regions

[Kirillov arXiv:1901.0244]
3D ConvNet for Medical Image Analysis (NYU)

- **Segmentation Femur from MR Images**
- **[Deniz et al. Nature 2018]**
3D ConvNet for Medical Image Analysis

[Images of 3T MRI, 2D CNN, Trabecular Bone Probability Map, Segmentation Mask]
Breast Cancer Detection (NYU)

ConvNets in Astrophysics [He et al. PNAS 07/2019]

Siyu He\textsuperscript{a,b,c,1}, Yin Li\textsuperscript{d,e,f}, Yu Feng\textsuperscript{d,e}, Shirley Ho\textsuperscript{a,b,c,d,e,1}, Siamak Ravanbakhsh\textsuperscript{g}, Wei Chen\textsuperscript{c}, and Barnabás Póczos\textsuperscript{h}

\begin{itemize}
\item 1. Train a coarse-grained 3D U-Net to approximate a fine-grained simulation on a small volume
\item 2. Use it for a simulation on a large volume (the early universe)
\end{itemize}
Driving Cars with Convolutional Nets

MobilEye (2015)

NVIDIA
What are we missing to get to “real” AI?

What we can have
- Safer cars, autonomous cars
- Better medical image analysis
- Personalized medicine
- Adequate language translation
- Useful but stupid chatbots
- Information search, retrieval, filtering
- Numerous applications in energy, finance, manufacturing, environmental protection, commerce, law, artistic creation, games,…..

What we cannot have (yet)
- Machines with common sense
- Intelligent personal assistants
- “Smart” chatbots
- Household robots
- Agile and dexterous robots
- Artificial General Intelligence (AGI)
How do Humans and Animal Learn?
So quickly
Babies learn how the world works by observation

Largely by observation, with remarkably little interaction.

Photos courtesy of Emmanuel Dupoux
Early Conceptual Acquisition in Infants [from Emmanuel Dupoux]

- Perception
  - Social-communicative
  - Actions: face tracking, biological motion
  - Physics: Object permanence, solidity, rigidity
  - Objects: natural kind categories

- Production
  - Social-communicative: emotional contagion, proto-imitation
  - Rational, goal-directed actions: helping vs hindering
  - False perceptual beliefs: gravity, inertia, conservation of momentum
  - Shape constancy: pointing
  - Biological motion: walking, crawling

Time (months)
Prediction is the essence of Intelligence

- We learn models of the world by predicting.
The Next AI Revolution

THE REVOLUTION WILL NOT BE SUPERVISED
(nor purely reinforced)

Get the T-shirt!

With thanks to Alyosha Efros and Gil Scott Heron
The Salvation?
Self-Supervised Learning

Training very large networks to understand the world through prediction
Predict any part of the input from any other part.

Predict the future from the past.

Predict the future from the recent past.

Predict the past from the present.

Predict the top from the bottom.

Predict the occluded from the visible

Pretend there is a part of the input you don’t know and predict that.
Three Types of Learning

- **Reinforcement Learning**
  - The machine predicts a scalar reward given once in a while.
  - weak feedback

- **Supervised Learning**
  - The machine predicts a category or a few numbers for each input.
  - medium feedback

- **Self-supervised Predictive Learning**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - A lot of feedback
How Much Information is the Machine Given during Learning?

- **“Pure” Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - A few bits for some samples

- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input.
  - Predicting human-supplied data
  - 10→10,000 bits per sample

- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - Millions of bits per sample
PhysNet: tell me what happens next

[Lerer, Gross, Fergus ICML 2016, arxiv:1603.01312]

ConvNet produces object masks that predict the trajectories of falling blocks. Blurry predictions when uncertain
Energy-Based Learning

Energy Shaping

Regularized Auto-Encoders
Learning an **energy function** (or contrast function) $F(Y)$, that takes

- Low values on the data manifold
- Higher values everywhere else
Capturing Dependencies Between Variables with an Energy Function

The energy surface is a “contrast function” that takes low values on the data manifold, and higher values everywhere else.

- Special case: energy = negative log density
- Example: the samples live in the manifold

\[ Y_2 = (Y_1)^2 \]
Energy-Based Unsupervised Learning

- **Energy Function:** Takes low value on data manifold, higher values everywhere else
- **Push down on the energy of desired outputs. Push up on everything else.**
- **But how do we choose where to push up?**

![Diagram showing plausible and implausible futures](image)
Transforming Energies into Probabilities (if necessary)

- The energy can be interpreted as an unnormalized negative log density
- Gibbs distribution: Probability proportional to $\exp(-\text{energy})$
- Beta parameter is akin to an inverse temperature
- Don't compute probabilities unless you absolutely have to
- Because the denominator is often intractable

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_Y e^{-\beta E(y,W)}}$$

$$E(Y, W) \propto -\log P(Y|W)$$
Learning the Energy Function

- Parameterized energy function $F(Y,W)$
  - Make the energy low on the samples
  - Make the energy higher everywhere else
  - Making the energy low on the samples is easy
  - But how do we make it higher everywhere else?
Energy surface for PCA and K-means

1. build the machine so that the volume of low energy stuff is constant
   - PCA, K-means, GMM, square ICA...

   **PCA**
   
   $$ F(Y) = \| W^T W Y - Y \|^2 $$

   **K-Means**,  
   **Z constrained to 1-of-K code**

   $$ F(Y) = \min_z \sum_i \| Y - W_i Z_i \|^2 $$
Seven Strategies to Shape the Energy Function

1. build the machine so that the volume of low energy stuff is constant
   - PCA, K-means, GMM, square ICA

2. push down of the energy of data points, push up everywhere else
   - Max likelihood (needs tractable partition function or variational approximation)

3. push down of the energy of data points, push up on chosen locations
   - Contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Min Probability Flow, adversarial generator/GANs

4. minimize the gradient and maximize the curvature around data points
   - score matching

5. if $F(Y) = ||Y - G(Y)||^2$, make $G(Y)$ as "constant" as possible.
   - Contracting auto-encoder, saturating auto-encoder

6. train a dynamical system so that the dynamics goes to the data manifold
   - denoising auto-encoder, masked auto-encoder (e.g. BERT)

7. use a regularizer that limits the volume of space that has low energy
   - Sparse coding, sparse auto-encoder, LISTA & PSD, Variational auto-encoders
push down of the energy of data points, push up everywhere else

Max likelihood (requires a tractable partition function)

Maximizing $P(Y|W)$ on training samples

$$P(Y|W) = \frac{e^{-\beta E(Y,W)}}{\int_Y e^{-\beta E(y,W)}}$$

Minimizing $-\log P(Y,W)$ on training samples

$$L(Y, W) = E(Y, W) + \frac{1}{\beta} \log \int_Y e^{-\beta E(y,W)}$$
Gradient of the negative log-likelihood loss for one sample $Y$:

$$\frac{\partial L(Y, W)}{\partial W} = \frac{\partial E(Y, W)}{\partial W} - \int_y P(y|W) \frac{\partial E(y, W)}{\partial W}$$

Gradient descent:

$$W \leftarrow W - \eta \frac{\partial L(Y, W)}{\partial W}$$

Pushes down on the energy of the samples

Pulls up on the energy of low-energy $Y$'s

$$W \leftarrow W - \eta \frac{\partial E(Y, W)}{\partial W} + \eta \int_y P(y|W) \frac{\partial E(y, W)}{\partial W}$$
Latent Variable Models

Sparse Modeling, Regularized Auto-Encoders
**Sparse Coding & Regularized Auto-Encoders**

- **y**: variable to be predicted
- **h**: encoding of x
- **z**: latent variable
- **Sparse modeling**
  - $E(y, z) = C(y, \text{Dec}(z)) + R(z)$
  - $z^* = \arg\min E(y, z)$

**Sparse Auto-Encoder**

- $E(y, z) = C(y, \text{Dec}(\text{Enc}(y))) + R(z)$
- [Kavukcuoglu CVPR 2009] [Gregor ICML 2010]

\[
F(Y) = \min_z E(Y, Z)
\]

\[
E(Y, Z) = \|Y - \sum_k W_k Z_k\|^2 + \alpha \sum_k |Z_k|
\]
The “Decoder with Regularized Latent Variable” Model

- $Y' = \text{Dec}(Z)$  
  $Z^* = \text{argmin} \| Y - \text{Dec}(Z) \| + R(Z)$
- Linear decoder: K-Means, basis pursuit, K-SVD, sparse coding,…
- Sparse modeling: [Olshausen Field 1997]

$$F(Y) = \min_{Z} E(Y, Z)$$
$$E(Y, Z) = \| Y - \sum_{k} W_k Z_k \|^2 + \alpha \sum_{k} |Z_k|$$
Energy Surface for Sparse Modeling

Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition
Sparse Modeling on handwritten digits (MNIST)

- Basis functions (columns of decoder matrix) are digit parts.
- All digits are a linear combination of a small number of these.
Y. LeCun

Generation through Latent Optimization

[Bojanowski, Joulin, Lopez-Paz, Szlam arxiv:1707.05776]

\[ Y' = \text{Dec}(Z) \quad Z^* = \text{argmin} \| Y - \text{Dec}(Z) \|^2 \]

- Z is not explicitly regularized, just low-dimensional
Generation through Latent Optimization

- **Original**

- **Reconstr. D=100**

- **Reconstr. D=512**
Interpolation in Z space

Interpolation in pixels
Sparse Auto-Encoders

Computing the sparse code efficiently
Regularized Auto-Encoders

- $y$: variable to be predicted
- $h$: encoding of $x$
- $z$: internal representation

**Sparse Auto-Encoder**
- $E(y,z) = C(y, \text{Dec(Enc(y))}) + R(z)$
- $F(y) = \min\{z\} E(y,z)$
Regularized Auto-Encoders

- $y$: variable to be predicted
- $h$: encoding of $x$
- $z$: latent variable

**Sparse Auto-Encoder with latent variable**
- $E(y,z) = C[y, \text{Dec}(\text{Enc}(y))] + D[z,\text{Enc}(y)] + R(z)$
- [Kavukcuoglu CVPR 2009] [Gregor ICML 2010]
The “Encoder-Decoder with latent vars” Model

- \( E(Y,Z) = \| Y - \text{Dec}(Z) \|^2 + R(Z) + \| Z - \text{Enc}(Y) \|^2 \)
- \( Z^* = \arg\min_z E(Y,Z) \)

Linear decoder: Predictive Sparse Decomposition
- [Kavukcuoglu 2008 (arxiv:1010.3467)] [Kavukcuoglu CVPR 2009]
Training on natural images patches.
- 12X12
- 256 basis functions
Learned Features on natural patches: V1-like receptive fields
Replace the dot products with dictionary element by convolutions.
- Input $Y$ is a full image
- Each code component $Z_k$ is a feature map (an image)
- Each dictionary element is a convolution kernel

**Regular sparse coding**

$$E(Y, Z) = ||Y - \sum_k W_k Z_k||^2 + \alpha \sum_k |Z_k|$$

**Convolutional S.C.**

$$E(Y, Z) = ||Y - \sum_k W_k \ast Z_k||^2 + \alpha \sum_k |Z_k|$$

Also used in “deconvolutional networks” [Zeiler, Taylor, Fergus CVPR 2010]
Convolutional PSD: Encoder with a soft $sh()$ Function

**Convolutional Formulation**
- Extend sparse coding from **PATCH** to **IMAGE**

$$\mathcal{L}(x, z, D) = \frac{1}{2} \| x - \sum_{k=1}^{K} D_k * z_k \|_2^2 + \sum_{k=1}^{K} \| z_k - f(W^k * x) \|_2^2 + \| z \|_1$$

- **PATCH** based learning
- **CONVOLUTIONAL** learning
Filters and Basis Functions obtained

- with 1, 2, 4, 8, 16, 32, and 64 filters.
ISTA/FISTA: iterative algorithm that converges to optimal sparse code

\[ Z(t + 1) = \text{Shrinkage}_{\lambda/L} \left[ Z(t) - \frac{1}{L} W_d^T (W_d Z(t) - Y) \right] \]

ISTA/FastISTA reparameterized:

\[ Z(t + 1) = \text{Shrinkage}_{\lambda/L} \left[ W_e^T Y + S Z(t) \right] ; \quad W_e = \frac{1}{L} W_d; \quad S = I - \frac{1}{L} W_d^T W_d \]

LISTA (Learned ISTA): learn the \( W_e \) and \( S \) matrices to get fast solutions

[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]
LISTA: Train We and S matrices to give a good approximation quickly

Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters

- Time-Unfold the flow graph for K iterations
- Learn the We and S matrices with “backprop-through-time”
- Get the best approximate solution within K iterations
Learning ISTA (LISTA) vs ISTA/FISTA

Number of LISTA or FISTA iterations

Reconstruction Error

error

FISTA (4x)
FISTA (1x)
LISTA (4x)
LISTA (1x)

iter

0 1 2 3 4 5 6 7
LISTA with partial mutual inhibition matrix

![Graph showing reconstruction error vs. proportion of S matrix elements that are non-zero. The graph includes different data points for different conditions such as dim reduction (4x), elements removal (4x), dim reduction (1x), and elements removal (1x). There is an arrow pointing to the smallest elements removed.]
Learning Coordinate Descent (LcoD): faster than LISTA

![Graph showing error vs. number of iterations for different iterations of CoD and LCoD compared to LISTA.](image)
Unsupervised PSD ignores the spatial pooling step.

Could we devise a similar method that learns the pooling layer as well?

Idea [Hyvarinen & Hoyer 2001]: **group sparsity** on pools of features

- Minimum number of pools must be non-zero
- Number of features that are on within a pool doesn't matter
- Polls tend to regroup similar features

\[
\|Y^i - \bar{Y}\|^2 \quad W_d Z \quad \|

\lambda \sum_j
\]
Why just pool over space? Why not over orientation?

- Using an idea from Hyvärinen: topographic square pooling (subspace ICA)
- 1. Apply filters on a patch (with suitable non-linearity)
- 2. Arrange filter outputs on a 2D plane
- 3. Square filter outputs
- 4. Minimize sqrt of sum of blocks of squared filter outputs
Why just pool over space? Why not over orientation?

- The filters arrange themselves spontaneously so that similar filters enter the same pool.
- The pooling units can be seen as complex cells.
- They are invariant to local transformations of the input.
- For some it's translations, for others rotations, or other transformations.
Pinwheels!

Does that look pinwheely to you?
Variational Auto-Encoder

Limiting the information content of the code by adding noise to it
Variational Auto-Encoder

- Corrupt the code to limit its information content
Variational Auto-Encoder

Code vectors for training samples
Variational Auto-Encoder

- Code vectors for training sample with Gaussian noise
- Some fuzzy balls overlap, causing bad reconstructions
Variational Auto-Encoder

- The code vectors want to move away from each other to minimize reconstruction error
- But that does nothing for us
Variational Auto-Encoder

- Attach the balls to the center with a spring, so they don’t fly away
- Minimize the square distances of the balls to the origin
- Center the balls around the origin
- Make the center of mass zero
- Make the sizes of the balls close to 1 in each dimension
- Through a so-called KL term
Denoising Auto-Encoders

Masked AE, BERT...
Denoising Auto-Encoder

- Corrupt the input, train the network to recover the original input
Denoising Auto-Encoder

- **Pink points:**
  - training samples

- **Orange points:**
  - corrupted training samples
  - Additive Gaussian noise

- Figures credit:
  Alfredo Canziani
Data

- **Pink points:**
  - training samples

- **Orange points:**
  - corrupted training samples
  - Additive Gaussian noise

- **Blue points:**
  - Network outputs
  - Denoised samples
Data

- Orange points:
  - Test samples on a grid

- Blue points:
  - Network outputs
Data

- **Pink points:**
  - Training samples

- **Color:**
  - Reconstruction energy

- **Vector field:**
  - Displacement from network input to network output (scaled down)
Self-Supervised Learning: filling in the bl_nks

Natural Language Processing: works great!

**OUTPUT:** This is a piece of text extracted from a large set of news articles

Image Recognition / Understanding: works so-so

**INPUT:** This is a [……] of text extracted [……] a large set of [……] articles

[Pathak et al 2014]
Self-Supervised Learning works well for text

- **Word2vec**
  - [Mikolov 2013]

- **FastText**
  - [Joulin 2016] (FAIR)

- **BERT**
  - Bidirectional Encoder Representations from Transformers
    - [Devlin 2018]

- **Cloze-Driven Auto-Encoder**
  - [Baevski 2019] (FAIR)
Self-Supervised Learning: Filling in the Blanks

input

Barnes et al. | 2009

Darabi et al. | 2012

Huang et al. | 2014

Pathak et al. | 2016

Iizuka et al. | 2017
Self-supervised: jigsaw problem and colorization problem
Wav2vec [Schneider et al. 2019]

- Self-supervised speech features
- Classify a future audio segment as compatible or not
- Similar to jigsaw scenario
- Improves performance of wav2letter
SSL works with discrete data

- **BERT / LM**: discrete distribution on words
- **Colorization**: discrete distribution on quantized color bins
- **Jigsaw**: fundamentally a classification problem
- **Wav2vec**: fundamentally a 2-class classification problem

- SSL works because we know how to represent uncertainty with discrete distributions (see also [Ranzato 2014] on video prediction)

- But how do we make it work with high-dimensional continuous data?
- Video prediction, audio prediction....
The world is not entirely predictable / stochastic

- **Video prediction:**
  - Multiple futures are possible.
  - Training a system to make a single prediction results in “blurry” results
  - the average of all the possible futures
SSL through Video Prediction

- Some success
  - [Mathieu, Couprie, YLC ICLR’16 arXiv:1511:05440]
  - [Luc, Couprie, LeCun, Verbeek ECCV 2018]
- But we are far from a complete solution
Adversarial Training & Video Prediction
Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).
Adversarial Training: the key to prediction under uncertainty?

- Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]
Adversarial Training: the key to prediction under uncertainty?

- Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]
Adversarial Training: the key to prediction under uncertainty?

- Generative Adversarial Networks (GAN) [Goodfellow et al. NIPS 2014],
- Energy-Based GAN [Zhao, Mathieu, LeCun ICLR 2017 & arXiv:1609.03126]

Dataset $T(X)$

$\text{Generator } G(X,Z)$

Discriminator $F(X,Y)$

F: minimize $F(X,Y)$

F: maximize $F(X,Y)$

Past: $X$ $\rightarrow$ Actual future $\rightarrow$ $Y$ $\rightarrow$ Predicted future $\rightarrow$ $Z$
Faces “invented” by a GAN (Generative Adversarial Network)

Random vector $\rightarrow$ Generator Network $\rightarrow$ output image [Goodfellow NIPS 2014]
[Karras et al. ICLR 2018] (from NVIDIA)
Generative Adversarial Networks for Creation

[Sbai 2017]
Our brains are “prediction machines”

Can we train machines to predict the future?

Some success with “adversarial training”

[Mathieu, Couprie, LeCun arXiv:1511:05440]

But we are far from a complete solution.
Learning Forward Models Through Video Prediction

[Lerer, Gross, Fergus ICML 2016, arxiv:1603.01312]
[Mathieu, Couprie, LeCun ICLR 2014]
[Luc, Couprie, LeCun, Verbeek ECCV 2018]
PhysNet: tell me what happens next

[Lerer, Gross, Fergus ICML 2016, arxiv:1603.01312]

ConvNet produces object masks that predict the trajectories of falling blocks. **Blurry predictions when uncertain**
Self-supervised Adversarial Learning for Video Prediction

- Our brains are “prediction machines”
- Can we train machines to predict the future?
- Some success with “adversarial training”
- But we are far from a complete solution.
Predicting Instance Segmentation Maps

- [Luc, Couprie, LeCun, Verbeek ECCV 2018]
- Mask R-CNN Feature Pyramid Network backbone
- Trained for instance segmentation on COCO
- Separate predictors for each feature level
Long-term predictions (10 frames, 1.8 seconds)

[Luc, Couprie, LeCun, Verbeek ECCV 2018]
Video Prediction With Uncertainty

Conditional Regularized Auto-Encoders
The world is not entirely predictable / stochastic

- **Video prediction:**
  - Multiple futures are possible.
  - Training a system to make a single prediction results in “blurry” results
  - the average of all the possible futures
We need to use latent-variable to pick up the uncertainty

Invariant prediction: The training samples are merely representatives of a whole set of possible outputs (e.g. a manifold of outputs).
Regularized Latent-Variable Predictive Models

- **x**: observed variables
- **y**: variable to be predicted
- **h**: encoding of x
- **z**: latent variable
  - Captures what is unpredictable

\[ E(x,y,z) = C[y, \text{Dec(Pred(x), z)}] + R(z) \]
Regularized Conditional Auto-Encoder

- $x$: observed variables
- $y$: variable to be predicted
- $h$: encoding of $x$
- $z$: latent variable
  - Captures what is unpredictable
Regularized Auto-Encoders

- **y**: variable to be predicted
- **h**: encoding of x
- **z**: latent variable

**Sparse Auto-Encoder**
- \( E(y,z) = C[y, \text{Dec}(\text{Enc}(y))] + D[z,\text{Enc}(y)] + R(z) \)
- [Kavukcuoglu CVPR 2009] [Gregor ICML 2010]
Learning Models of the World

Learning motor skills with no interaction with the real world
[Henaff, Canziani, LeCun ICLR 2019]
[Henaff, Zhao, LeCun ArXiv:1711.04994]
[Henaff, Whitney, LeCun Arxiv:1705.07177]
Planning Requires Prediction

To plan ahead, we simulate the world

Diagram:
- World
- Perceps
- Actions/Outputs
- Agent
- Objective
- Cost
- Actor
- Critic
- Inferred World State
- Action Proposals
- Actor State
- Predicted Perceps
- Predicted Cost
- Agent State

Summary:
- Planning requires prediction.
- To plan ahead, we simulate the world.
- Diagram illustrates the interactions between the world, agent, actor, critic, inferred world state, action proposals, actor state, predicted percepts, predicted cost, agent state, and objective actions/outputs.
Training the Actor with Optimized Action Sequences

- 1. Find action sequence through optimization
- 2. Use sequence as target to train the actor
- Over time we get a compact policy that requires no run-time optimization
Planning/learning using a self-supervised predictive world model

- Feed initial state
- Run the forward model
- Backpropagate gradient of cost
- Act
  - (model-predictive control)
  or
- Use the gradient to train a policy network.
- Iterate
Using Forward Models to Plan (and to learn to drive)

- Overhead camera on highway.
- Vehicles are tracked.
- A “state” is a pixel representation of a rectangular window centered around each car.
- Forward model is trained to predict how every car moves relative to the central car.
- Steering and acceleration are computed.
Forward Model Architecture

Architecture:

Encoder

Decoder

Latent variable predictor
Latent variable is predicted from the target.

The latent variable is set to zero half the time during training (drop out) and corrupted with noise.

The model predicts as much as it can without the latent var.

The latent var corrects the residual error.
Actual, Deterministic, VAE+Dropout Predictor/encoder
**Cost optimized for Planning & Policy Learning**

- **Differentiable cost function**
  - Increases as car deviates from lane
  - Increases as car gets too close to other cars nearby in a speed-dependent way

- **Uncertainty cost:**
  - Increases when the costs from multiple predictions (obtained through sampling of drop-out) have high variance.
  - Prevents the system from exploring unknown/unpredictable configurations that may have low cost.

---

![Images](a) 19.8 km/h  
(b) 50.3 km/h
Learning to Drive by Simulating it in your Head

- Feed initial state
- Sample latent variable sequences of length 20
- Run the forward model with these sequences
- Backpropagate gradient of cost to train a policy network.
- Iterate

No need for planning at run time.
Adding an Uncertainty Cost (doesn’t work without it)

- Estimates epistemic uncertainty
- Samples multiple dropouts in forward model
- Computes variance of predictions (differentiably)
- Train the policy network to minimize the lane&proximity cost plus the uncertainty cost.
- Avoids unpredictable outcomes
Driving an Invisible Car in “Real” Traffic
Driving!

- Yellow: real car
- Blue: bot-driven car
Yellow: real car
Blue: bot-driven car
Thank you