"Creative" AI

Richard Corrado

Fat Cat Machine Learning

github.com/richcorrado

Goals

- Review some general concepts of Machine Learning.
- Motivate and review Neural Networks (NNs), including Convolutional and Recurrent NNs.
- Survey applications of NNs to visual, audio, and textual media.
- Objective is to transform an object or mimic a human task according to information learned from training examples, rather than classic ML goal of prediction.

Classic Machine Learning

From some quantities $\mathbf{x}$ (the **features**), predict a value for a quantity y (the **target**). In principle, there is some functional relationship

$$y = f(\mathbf{x}).$$

However:

- Detailed form of f is unknown (complexity of underlying system).
- Predictors $\mathbf{x}$ may be incomplete or imperfect (complexity of data, randomness). Even if we knew f, could only compute within some margin of error

$$f(\mathbf{x}) = y \pm \epsilon.$$

Machine Learning (ML) includes the study and application of algorithmic models to find approximations

$$\hat{f}(\mathbf{x}) \approx f(\mathbf{x}).$$

Statistical principles of validation, inference, etc., are used to determine the errors of the models.

Regression: Target y is a continuous variable.

- Prices or other expected values.
- Expected demand in number of units (approximately continuous for large numbers).
- Physical dimension of object (area, mass).

Classification: Target y takes some discrete and finite number of values.

- What is the species of the object? (Iris classification)

	sepal_length	sepal_width	petal_length	petal_width	species	species_name
94	5.6	2.7	4.2	1.3	1	versicolor
48	5.3	3.7	1.5	0.2	0	setosa
122	7.7	2.8	6.7	2.0	2	virginica
76	6.8	2.8	4.8	1.4	1	versicolor
4	5.0	3.6	1.4	0.2	0	setosa
40	5.0	3.5	1.3	0.3	0	setosa
92	5.8	2.6	4.0	1.2	1	versicolor
32	5.2	4.1	1.5	0.1	0	setosa
22	4.6	3.6	1.0	0.2	0	setosa
149	5.9	3.0	5.1	1.8	2	virginica

- Image recognition

Neural Networks

In order to keep the training process mathematically stable and computationally managable, NNs are based on fairly simple functions:

- Linear functions:

$$\mathbf{z^{(i)}} = \mathbf{f}^{(i-1)}\mathbf{W}^{(i)} + \mathbf{b}^{(i)},$$

$$\mathbf{W}^{(i)}: \quad \text{weights, with shape } (F_{i-1}, F_i),$$

$$\mathbf{b}^{(i)}: \quad \text{biases, with shape } (F_i).$$

- Nonlinear functions, called **activation functions**:

$$\mathbf{f}^{(i)} = g^{(i)}(\mathbf{z}^{(i)}).$$

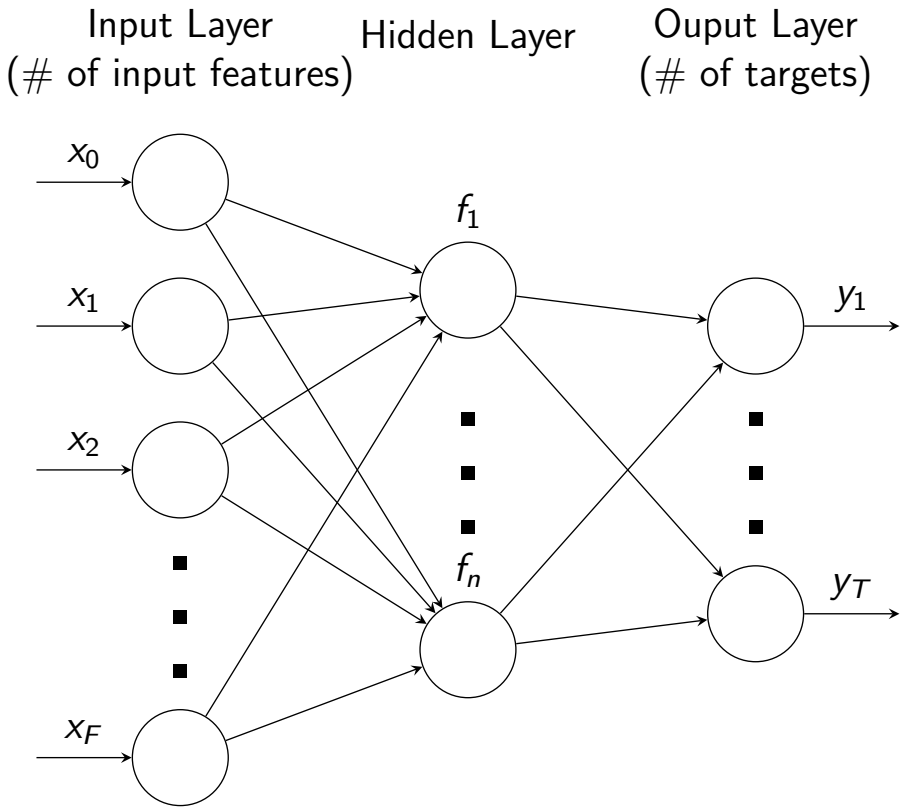
- So each layer learns **new features**:

$$\mathbf{f}^{(i)} = g^{(i)}\left(\mathbf{f}^{(i-1)}\mathbf{W}^{(i)} + \mathbf{b}^{(i)}\right).$$

Terminology for the layers is:

- **Input layer**: The first layer of the network is designed to simply load in the input features $\mathbf{x}$ present in the dataset.
- **Output layer**: The last layer is designed to produce an output that can be compared directly to the targets $\mathbf{y}$.
- **Hidden layers**: The layers in between compute the new features $\mathbf{f}^{(i)}$. Not connected to input or output.
- **Width**: Number of new features of a single hidden layer.
- **Depth**: Total number of hidden layers in the network. If depth is ≥ 3 we can say that we have a **deep neural network**.

Single Hidden Layer



A network like this, where every input feature is connected to a new feature is called a **feedforward network** or, in older terms, a **multilevel perceptron** (MLP).

Composition of Functions

Recall that a typical goal of machine learning is to approximate the functional relationship between the input features $\mathbf{x}$ and some target output $\mathbf{y}$:

$$\mathbf{y} = f(\mathbf{x}).$$

NNs naturally arrive at an approximation of this function via the composition of the functions learned at the hidden layers:

$$\hat{f}(\mathbf{x}) = f^{(H)} \circ \cdots \circ f^{(1)}(\mathbf{x}).$$

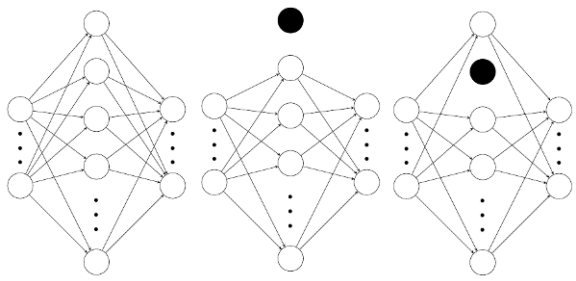
Modifications of Feedforward Architecture

In a fully-connected feed-forward network, each hidden layer feature is connected to a feature from the previous layer

$$z_{a_i}^{(i)} = \sum_{a_{i-1}=0}^{F_{i-1}-1} f_{a_{i-1}}^{(i-1)} W_{a_{i-1}a_i}^{(i)} + b_{a_i}^{(i)}.$$

We can modify the rules for computing the new features, creating new **architectures**, new types of NNs.

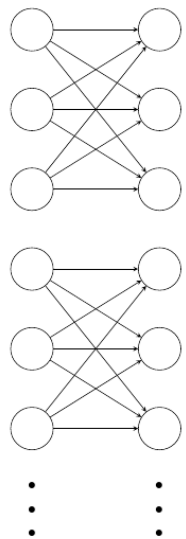
1. We can force some of the weights $W_{a_{i-1}a_i}^{(i)}$ to zero. This generally results in fully-connected models with a smaller capacity.



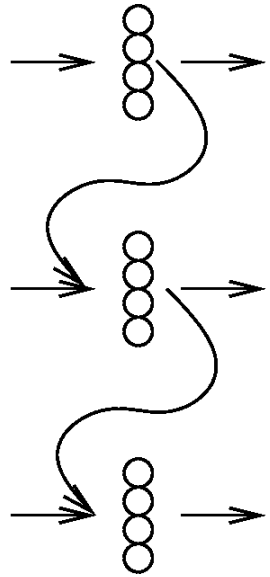
2. We can impose some symmetry on the weights $W^{(i)}_{a_{i-1}a_i}$, forcing some components to be equal to one another. This is called **weight sharing**.
3. We can remove some connections to the previous layer. This is equivalent to replacing

$$\mathbf{f}^{(i-1)}\mathbf{W}^{(i)} \rightarrow \sum_{a_{i-1}\in D} f^{(i-1)}_{a_{i-1}} W^{(i)}_{a_{i-1}a_i},$$

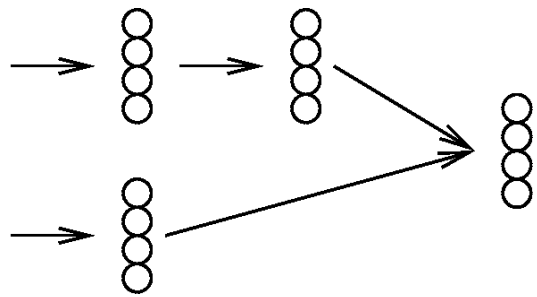
where D is some subset of the index set for the incoming features $\mathbf{f}^{(i-1)}$.



4. Can have more general connections between layers.
   - **Recurrent** networks have a chain structure between layers.

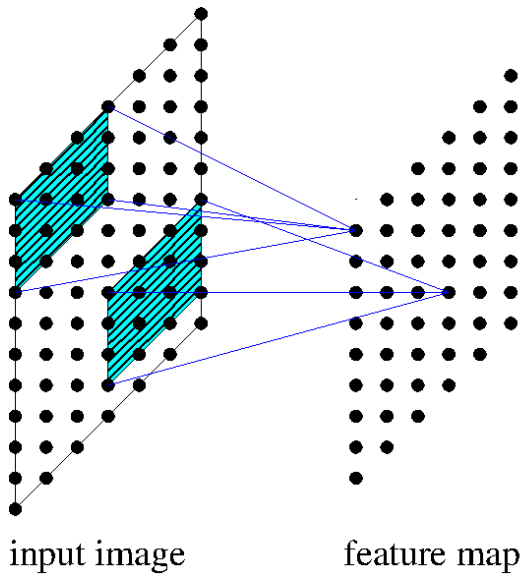


   - **Recursive** networks have a tree structure.



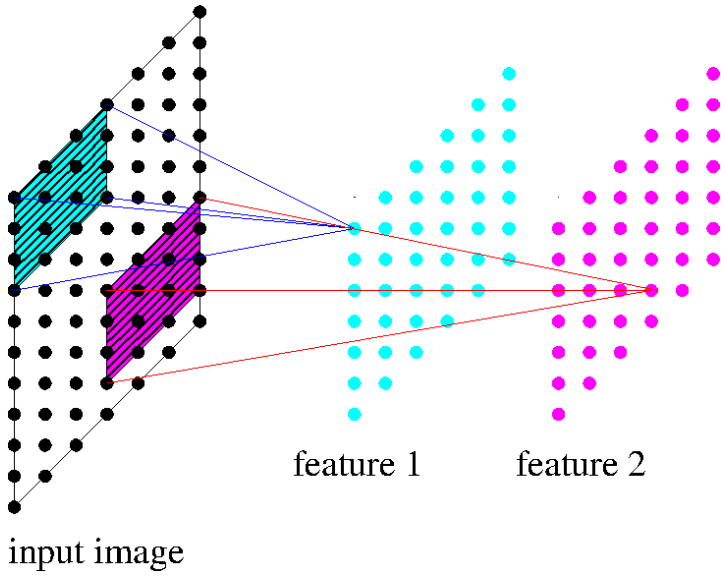
Convolutional Neural Networks

Convolutional NNs (CNNs) are designed to preserve information about how pixels in 2D images are spatially related:

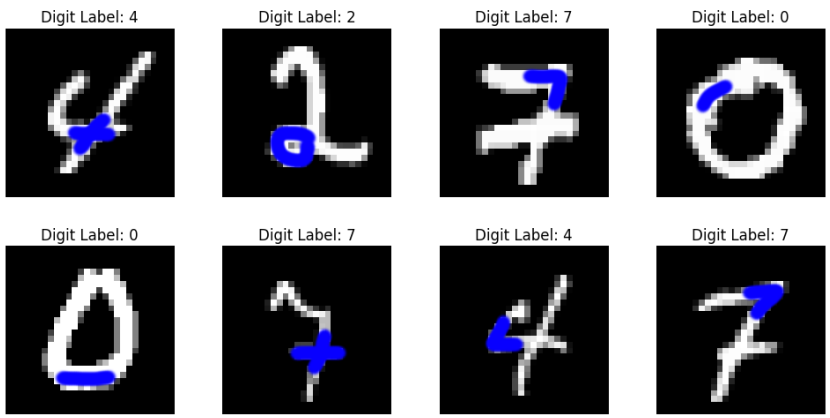


- **Partial connectivity**: Sample nearby pixels using a window.
- **Weight sharing**: Weights are shared over the whole image.

Filter: Collection of weights forming a window. Can use different filters to compute multiple types of features for a given image:



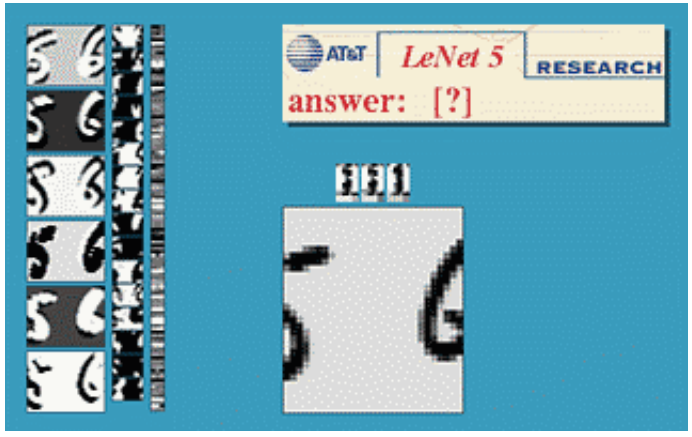
Translational invariance: CNN filters tend to learn geometric shapes, regardless of where they appear in an image:



Computer Vision

Since CNNs are well-suited for processing 2D images (but can also be used for 3D and non-image problems), they are extensively used in machine learning with images.

- **Character recognition**: MNIST digits, zip codes:



Google Streetview: **localization** and character recognition

100 vs. 676


1110 vs. 2641


23 vs. 37

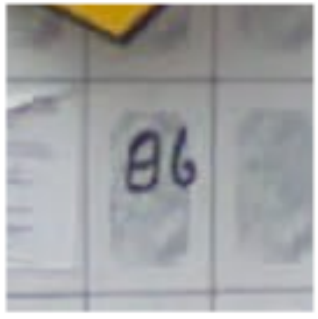
1 vs. 198


4 vs. 332

2 vs 239

1879 vs. 1879-1883

228 vs. 22B



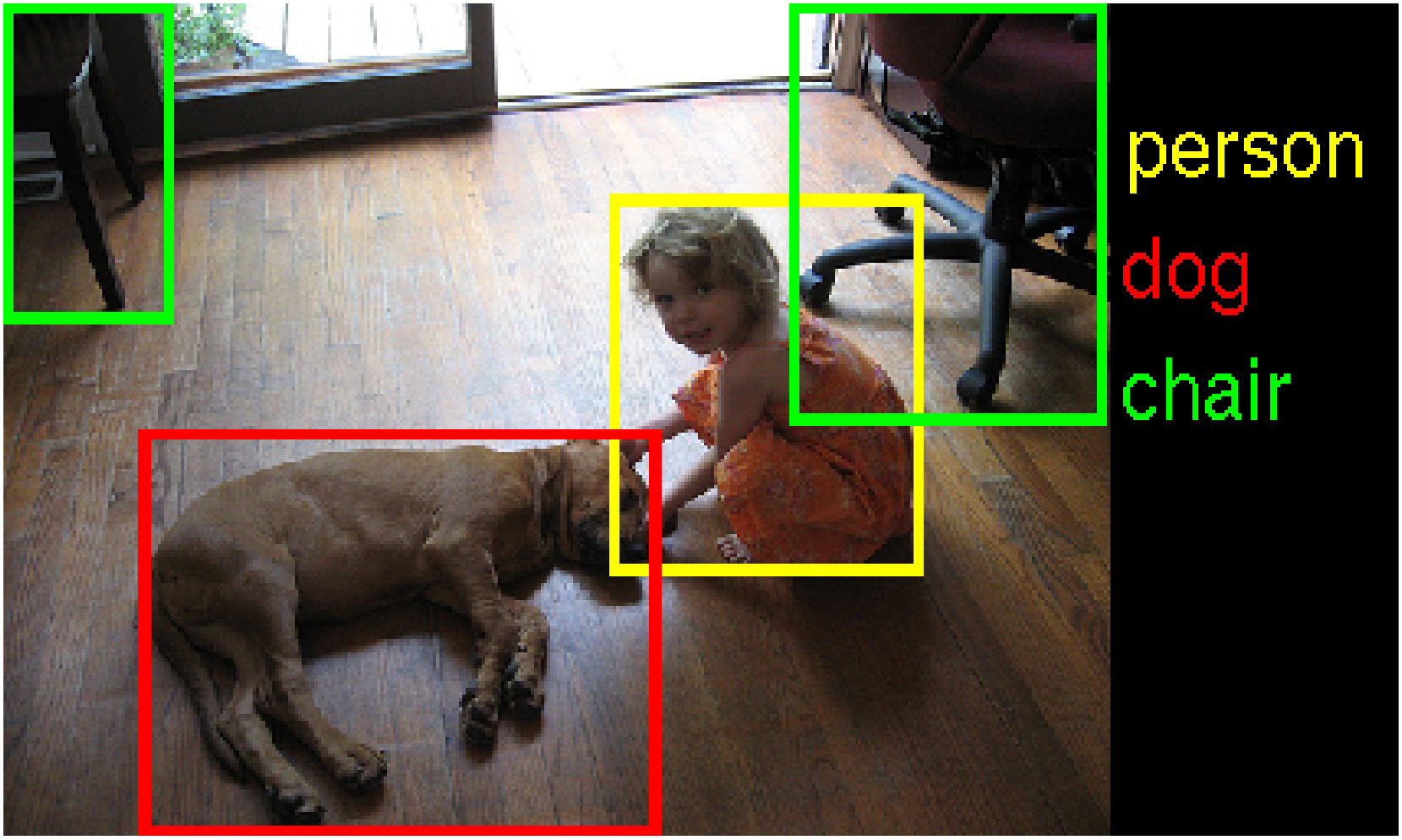
96 vs. 86

1844 vs. 184

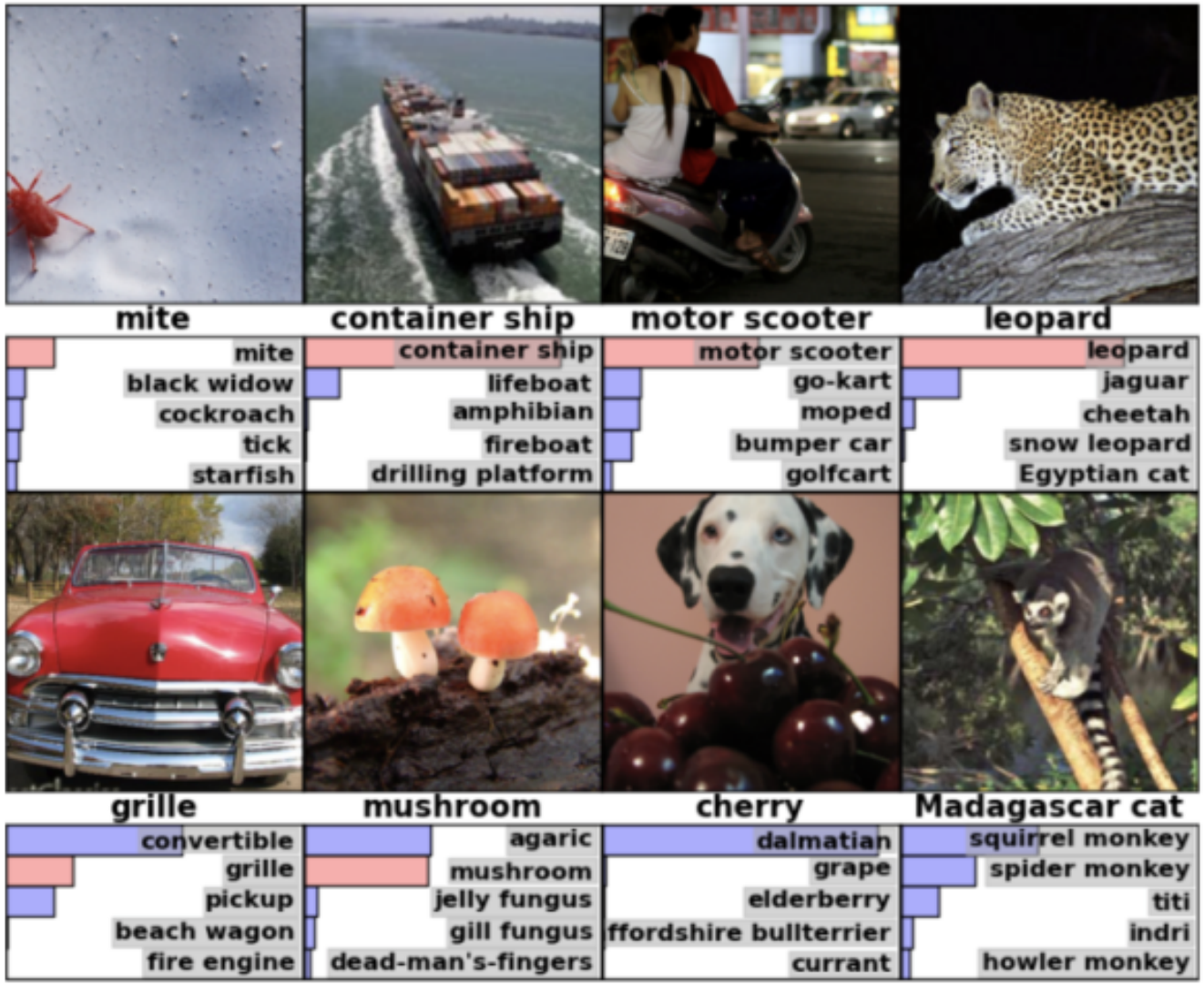
62 vs. 62-37

1180 vs. 1780

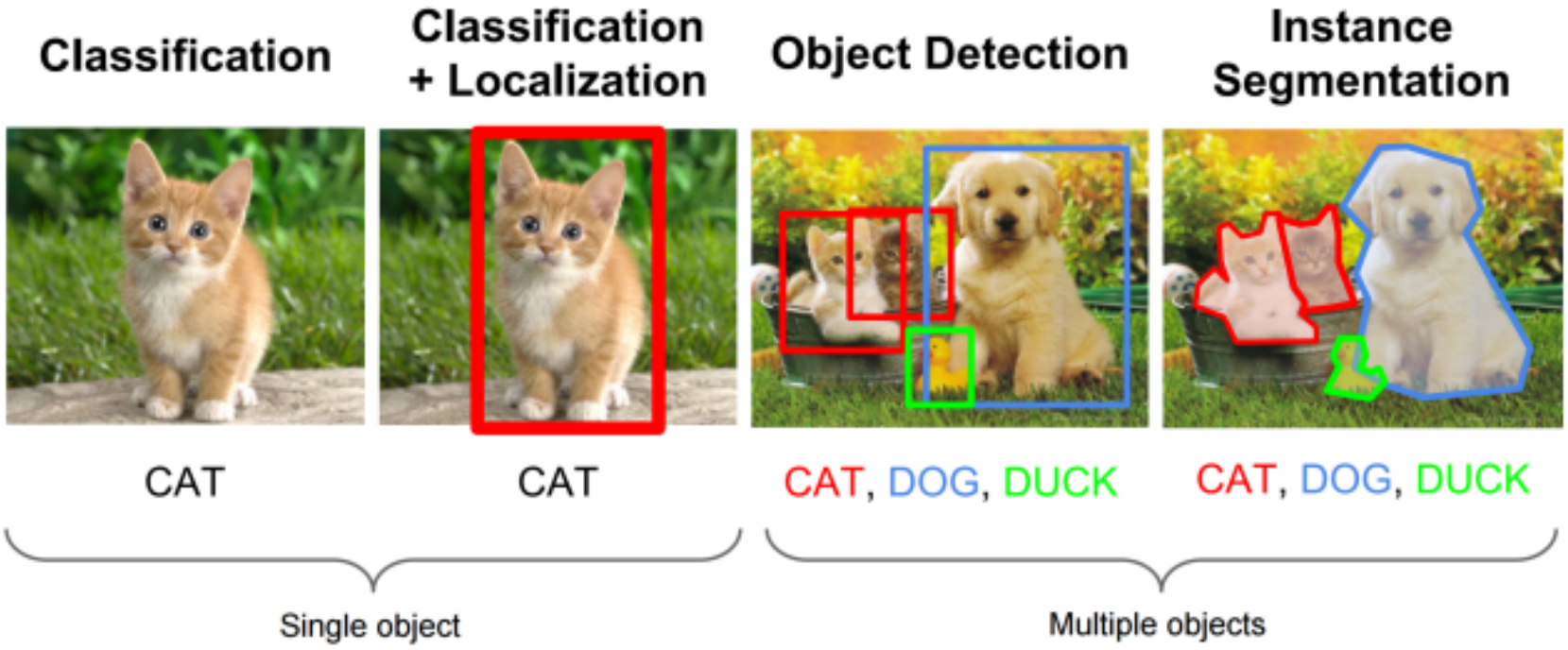
ImageNet: **object detection**



ImageNet: **classification**

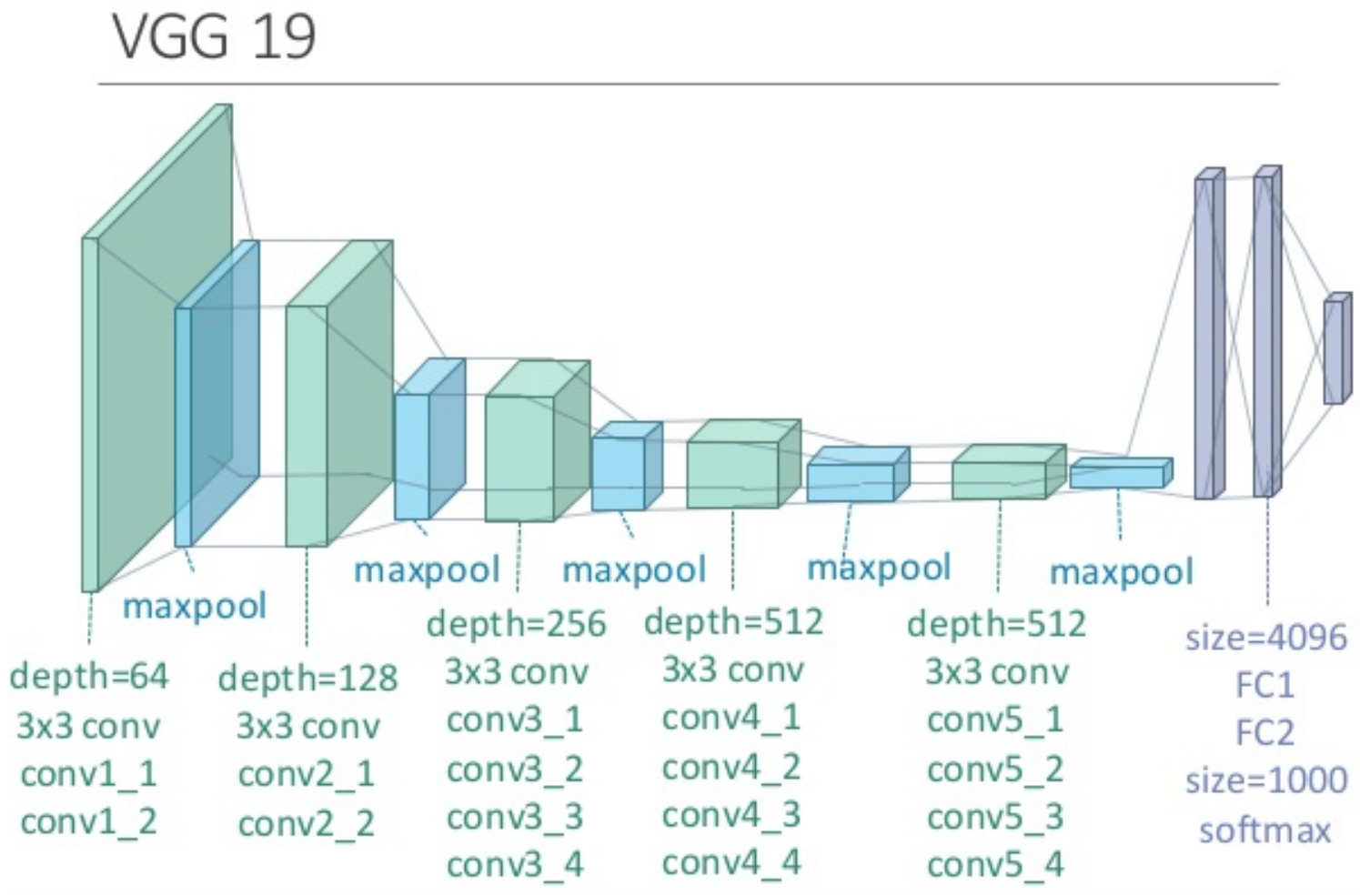


ImageNet: classification + **localization**



VGG-Network

Simonyan and Zisserman, ICLR 2015, arxiv:1409.1556,
1st/2nd in localization/classification, ImageNet ILSVRC-2014.



A Neural Algorithm of Artistic Style

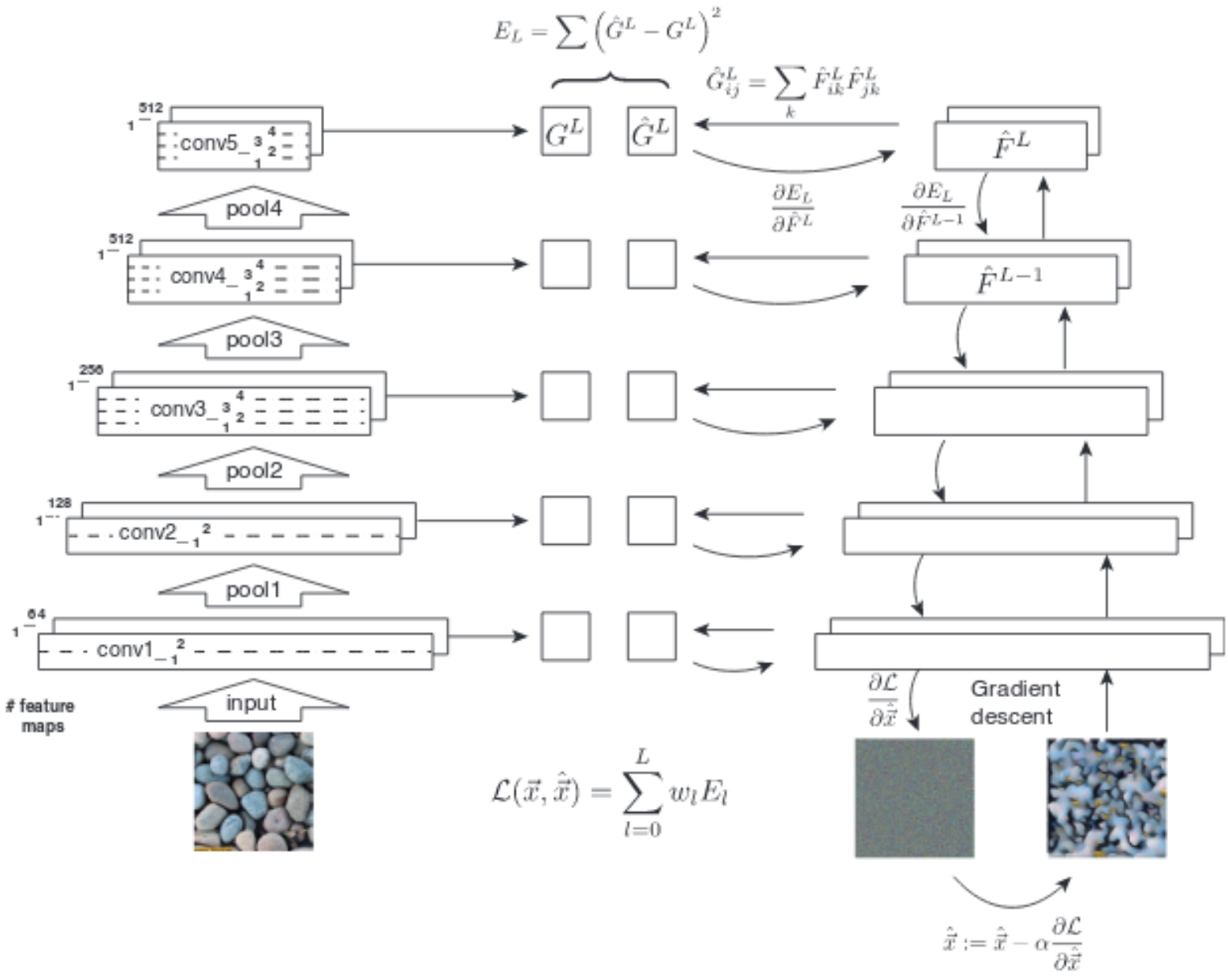
Gatys, Ecker, Bethge, Journal of Vision (2015), arxiv:1508.06576.

Consider a CNN trained on object recognition:

Content representation: At each layer, features capture the high-level **content** (objects + arrangement), but not exact pixel values

Style representation: Study correlations between the features at a given layer $\longrightarrow$ discards spatial information, but provides a stationary description of the texture of an image.

Method: Learn style representation by mapping a white noise image onto the texture of the image.


$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\hat{F}^{L-1}$
conv5_3
conv4_3
conv3_3
conv2_2
conv1_2
pool4
pool3
pool2
pool1
input
512
512
256
128
64
1
feature maps
$\frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$
Gradient descent
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$
$\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$

Details

F^l_{ij} is the activation of the i^{th} filter at position j in layer l.
$\vec{p}$: original image, $\vec{x}$: white-noise image
Cost function for content representation:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 \quad .$$

Cost function for style representation:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l,$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2,$$

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$

Style Transfer

Combine **content representation** of one image with **style representation** of another image.

Suppose:

$\vec{p}$ is a photograph: the **content**,
$\vec{a}$ is an artwork with a distinct textural **style**

Minimize the cost function

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Style Transfer Examples

AI and Art

Gatys, *et al.*, "...in light of the striking similarities between performance-optimised artificial neural networks and biological vision, our work offers a path forward to an algorithmic understanding of how humans create and perceive artistic imagery."

Some more examples from Gene Kogan

Torch implementation by jcjohnson.

TensorFlow implementation by lengstrom.
TensorFlow implementation by cysmith.
TensorFlow implementation by anishathalye.

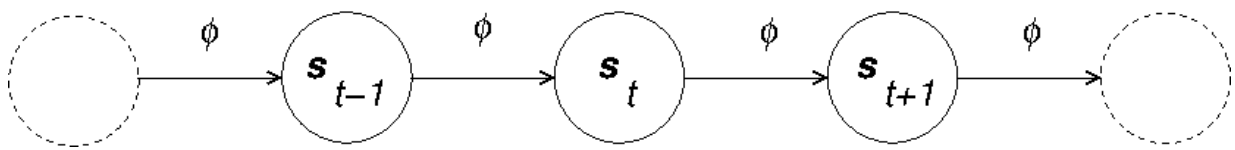
Recurrent Neural Nets

Time-varying data: $(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{s}_t \longrightarrow$ **state** of system.

Causality: state at time τ, $\mathbf{s}_\tau$ must only depend on states at times $t < \tau$.

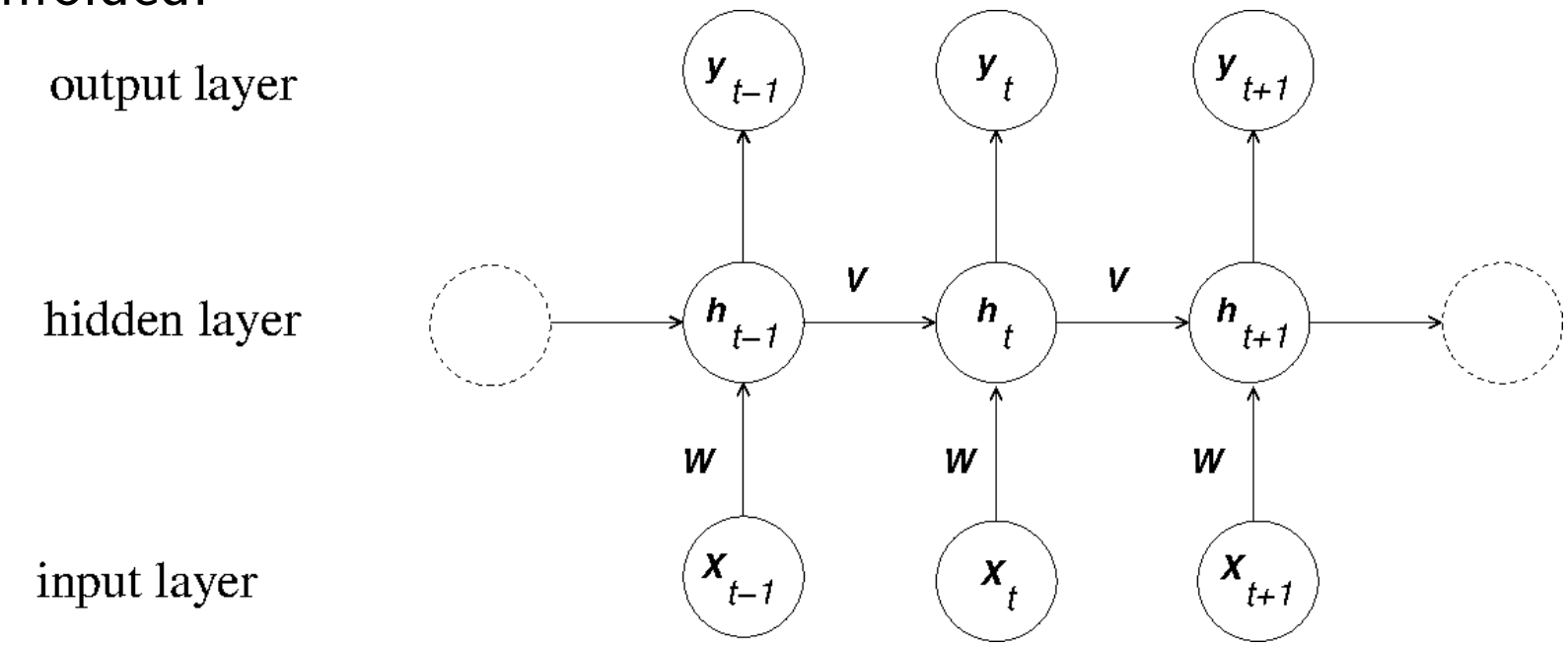
Dynamical system w/ evolution map ϕ:

$$\mathbf{s}_t = \phi(\mathbf{s}_{t-1}) = \phi(\phi(\mathbf{s}_{t-2})) = \cdots .$$

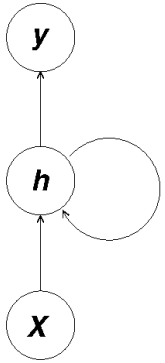


Recurrent Neural Network to Learn Dynamics

Unfolded:



Folded:

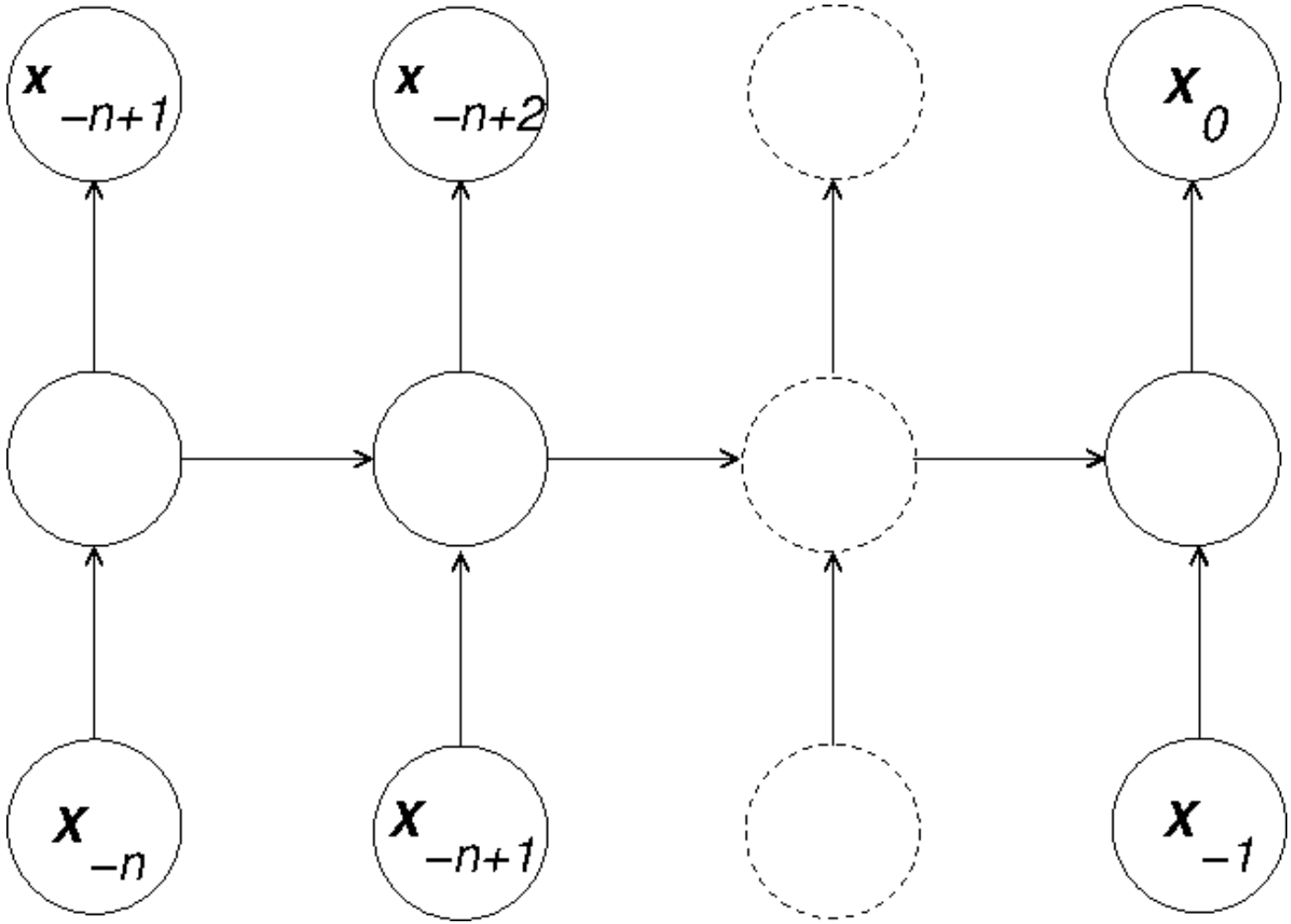


RNN Applications

Important when sequential relationship is important.

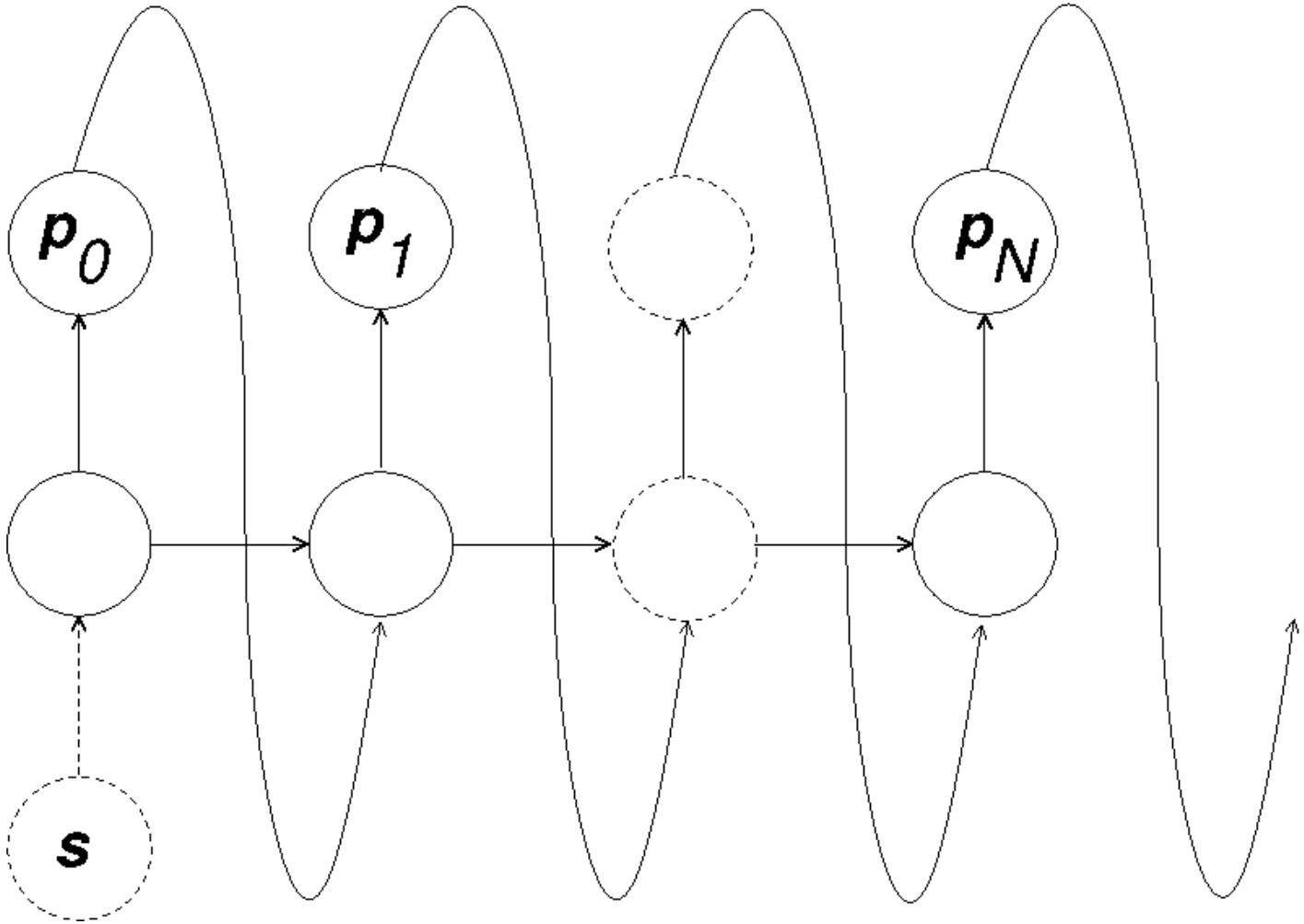
- Time series: stock price, weather, resource demand
- Language processing: word order $\rightarrow$ grammar, semantics
  spell/grammar-checking, translation
- Audio: speech recognition, signal processing/detection

For **prediction**: train on $(x_{-n}, \cdots x_{-1})$,
with targets $(x_{-n+1}, \cdots, x_0)$:



Creative RNNs

Take a trained model, seed it with $\mathbf{x} = \mathbf{s}$ and feed the output predictions back into the inputs:



Character-Level Language Models

See e.g., Andrej Karpathy blog (model uses Torch).

Given a large text corpus (all works of Shakespeare, wikipedia, etc.), learn the probability that each character follows another character.

These kinds of ideas have been around for decades in linguistics and natural language processing (NLP), but deep networks have pushed the applications along.

Ex: Shakespeare, sample inputs look like:

Second Witch:
By the pricking of my thumbs, Something wicked this way comes. Open, locks, Whoever knocks!

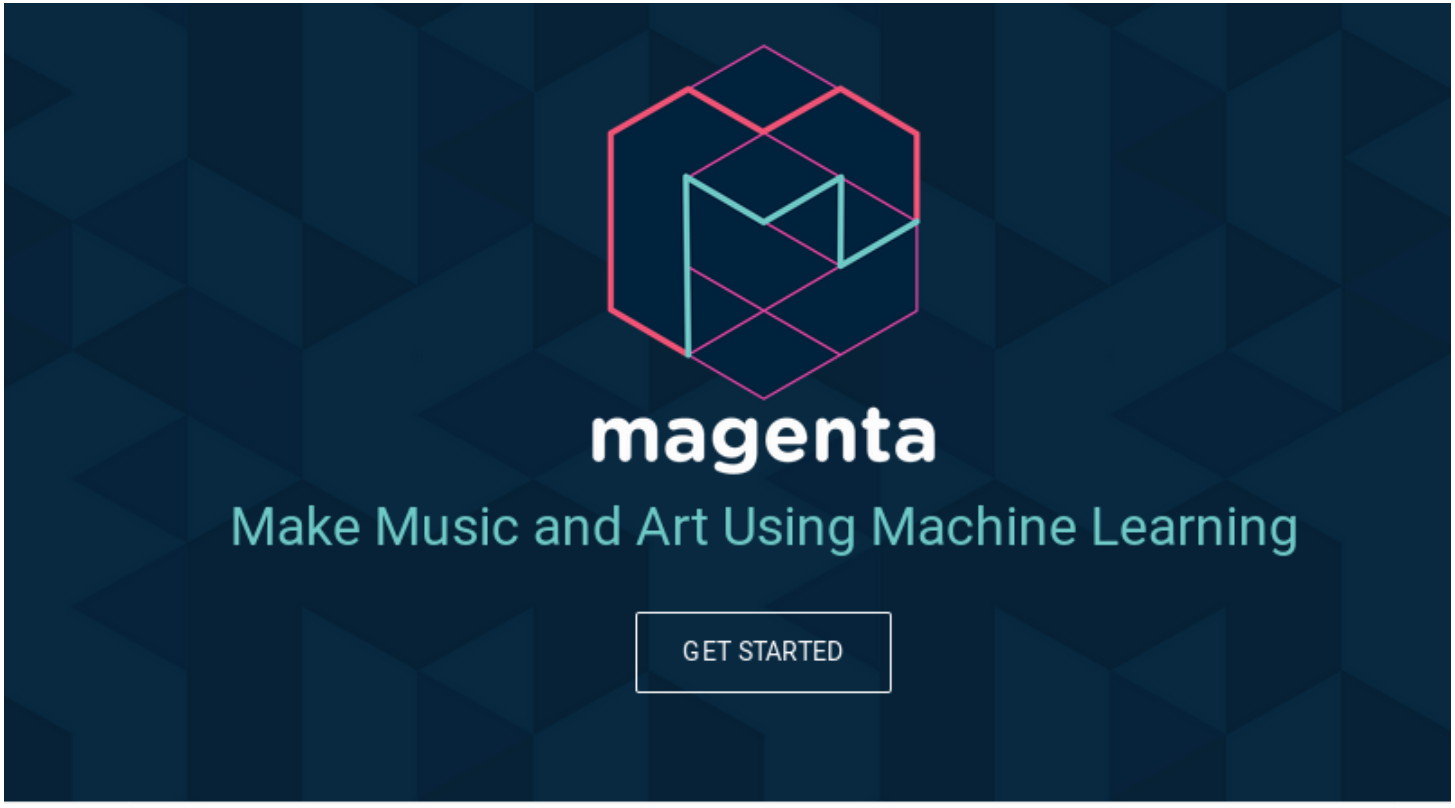
Sample output of model:

> *Second Senator:*
> *They are away this miseries, produced upon my soul,*
> *Breaking and strongly should be buried, when I*
> *perish The earth and thoughts of many states.*
>
> *DUKE VINCENTIO:*
> *Well, your wit is in the care of side and that.*
>
> *Clown:*
> *Come, sir, I will make did behold your worship.*
>
> *VIOLA:*
> *I'll drink it.*

Note that, from character-level learning, the model creates reasonable words and even reasonable character references!

Google Magenta

https://magenta.tensorflow.org/



About Magenta

Magenta is a Google Brain project to ask and answer the questions, “Can we use machine learning to create compelling art and music? If so, how? If not, why not?” Our work is done in TensorFlow, and we regularly release our models and tools in open source. These are accompanied by demos, tutorial blog postings and technical papers. To follow our progress, watch our GitHub and join our discussion group.

Note-Level Synthesis

See e.g., Paul Calhoun tutorial.

Corpus: Collection of midi files of Scottish folk music.

Summary

- We've seen a few different types of neural networks and how their architecture makes them suitable for certain learning problems.
- We've seen how CNNs can be used to learn content and style representations of images, leading to some dramatically interesting transformations of images.
- We've learned about how we can use RNNs to train to predict and create. We've seen applications of creating new text and audio works after learning sequence probabilities from an existing corpus.

Credits:

Several images and other graphics in this presentation are reproduced as a fair use of the original sources:

- The original source of the photo used in the twitter post appears to be yamesjames@reddit.
- The graphic on page 16 was copied from the section of Yann LeCun's website on LeNet-5.
- The image on page 17 was obtained from Goodfellow et al., arxiv:1312.6082.
- The image on page 18 was obtained from ImageNet.
- The image on page 19 is from Krizhevsky et al., NIPS 2012.
- The image on page 20 is from Stanford CS231n lecture slides, obtained here via C. Körner.
- The figure on page 21 was obtained from slides for a lecture by M. Chang in the Applied Deep Learning course by Y. Chen.
- The figure on page 23 was obtained from Gatys et al., arxiv:1505.07376.
- The images on page 26 and quote on page 27 were obtained from Gatys et al., arxiv:1508.06576.
- The sample output text quoted on page 34 is from the Andrej Karpathy blog.

The original creators of the content highlighted and linked to have my thanks and appreciation.