10-301/601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Backpropagation
+
RNNs

Matt Gormley
Lecture 13
Feb. 28, 2022

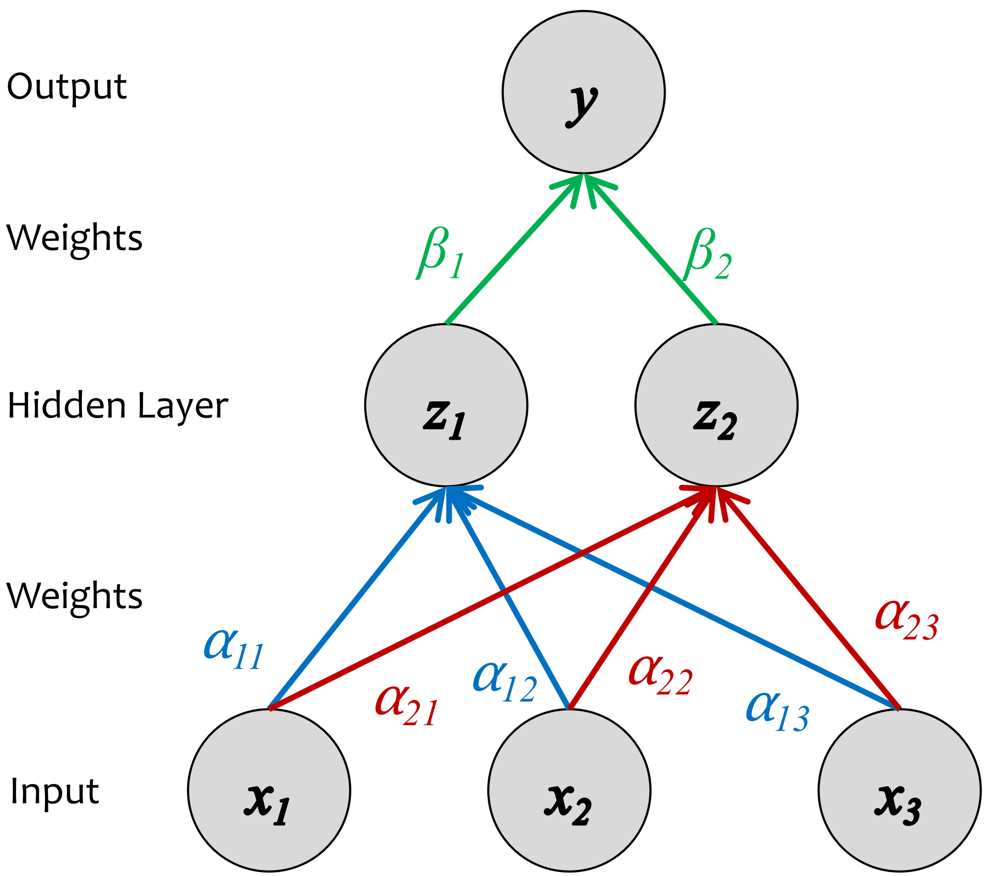
Reminders

- **Homework 5: Neural Networks**
  - Out: Sun, Feb 27
  - Due: Fri, Mar 18 at 11:59pm

A 1-Hidden Layer Neural Network

TRAINING A NEURAL NETWORK

Training Backpropagation



(E) **Output (sigmoid)**
$$y = \frac{1}{1+\exp(-b)}$$

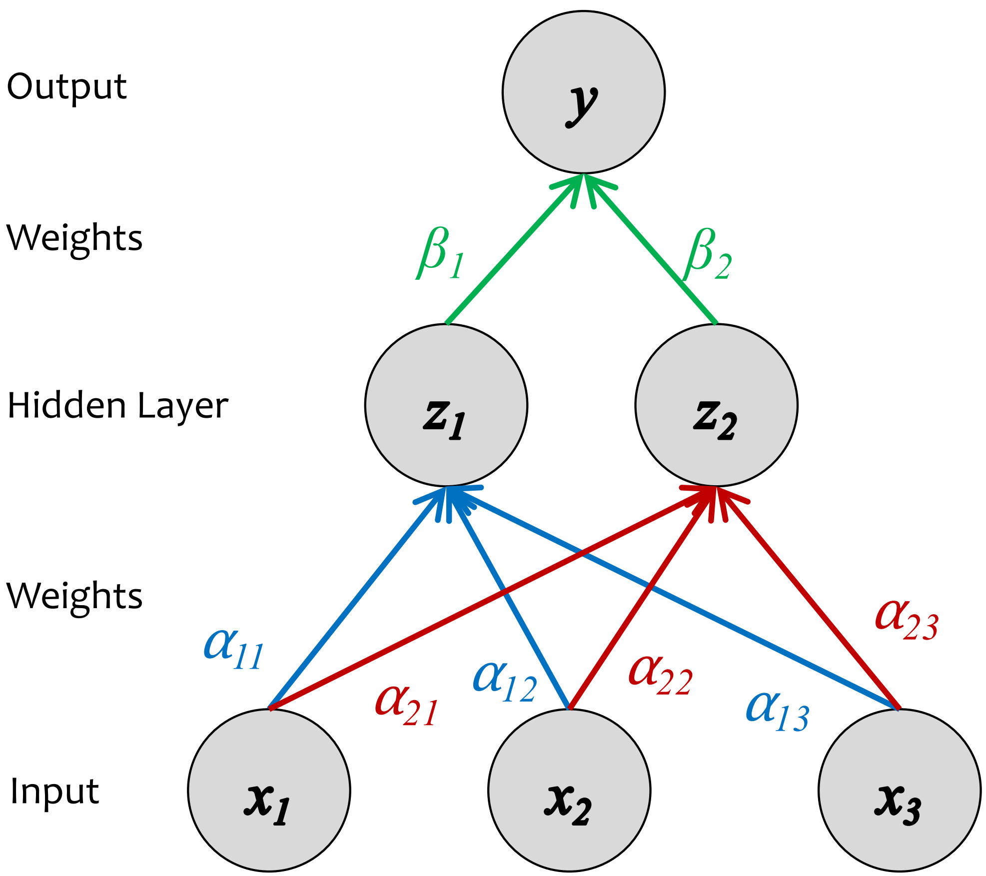
(D) **Output (linear)**
$$b = \sum_{j=0}^{D} \beta_j z_j$$

(C) **Hidden (sigmoid)**
$$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$$

(B) **Hidden (linear)**
$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$$

(A) **Input**
Given $x_i, \ \forall i$

Training Backpropagation



(F) **Loss**
$J = \frac{1}{2}(y - y^*)^2$

↑

(E) **Output (sigmoid)**
$y = \frac{1}{1+\exp(-b)}$

↑

(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

↑

(C) **Hidden (sigmoid)**
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

↑

(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

↑

(A) **Input**
Given $x_i, \ \forall i$

SGD with Backprop

Example: 1-Hidden Layer Neural Network

Algorithm 1 Stochastic Gradient Descent (SGD)

procedure SGD(Training data $\mathcal{D}$, test data $\mathcal{D}_t$)
    Initialize parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$
    for $e \in \{1, 2, \ldots, E\}$ **do**
        for $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ **do**
            Compute neural network layers:
            $\mathbf{o} = \texttt{object}(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{z}, \hat{\mathbf{y}}, J) = \text{NNFORWARD}(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$
            Compute gradients via backprop:
            $$\left.\begin{aligned} \mathbf{g}_{\boldsymbol{\alpha}} &= \nabla_{\boldsymbol{\alpha}} J \\ \mathbf{g}_{\boldsymbol{\beta}} &= \nabla_{\boldsymbol{\beta}} J \end{aligned}\right\} = \text{NNBACKWARD}(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{o})$$
            Update parameters:
            $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \gamma \mathbf{g}_{\boldsymbol{\alpha}}$
            $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \gamma \mathbf{g}_{\boldsymbol{\beta}}$
        Evaluate training mean cross-entropy $J_{\mathcal{D}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$
        Evaluate test mean cross-entropy $J_{\mathcal{D}_t}(\boldsymbol{\alpha}, \boldsymbol{\beta})$
    return parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$

A 1-Hidden Layer Neural Network

FORWARD COMPUTATION FOR A NEURAL NETWORK

Training

SGD with Backprop

Example: 1-Hidden Layer Neural Network

Algorithm 1 Stochastic Gradient Descent (SGD)

procedure SGD(Training data $\mathcal{D}$, test data $\mathcal{D}_t$)

  Initialize parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$

  for $e \in \{1, 2, \ldots, E\}$ **do**

    for $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ **do**

      Compute neural network layers:

      $\mathbf{o} = \texttt{object}(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{z}, \hat{\mathbf{y}}, J) = \text{NNFORWARD}(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

      Compute gradients via backprop:

$$\left.\begin{aligned} \mathbf{g}_{\boldsymbol{\alpha}} &= \nabla_{\boldsymbol{\alpha}} J \\ \mathbf{g}_{\boldsymbol{\beta}} &= \nabla_{\boldsymbol{\beta}} J \end{aligned}\right\} = \text{NNBACKWARD}(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{o})$$

      Update parameters:

      $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \gamma \mathbf{g}_{\boldsymbol{\alpha}}$

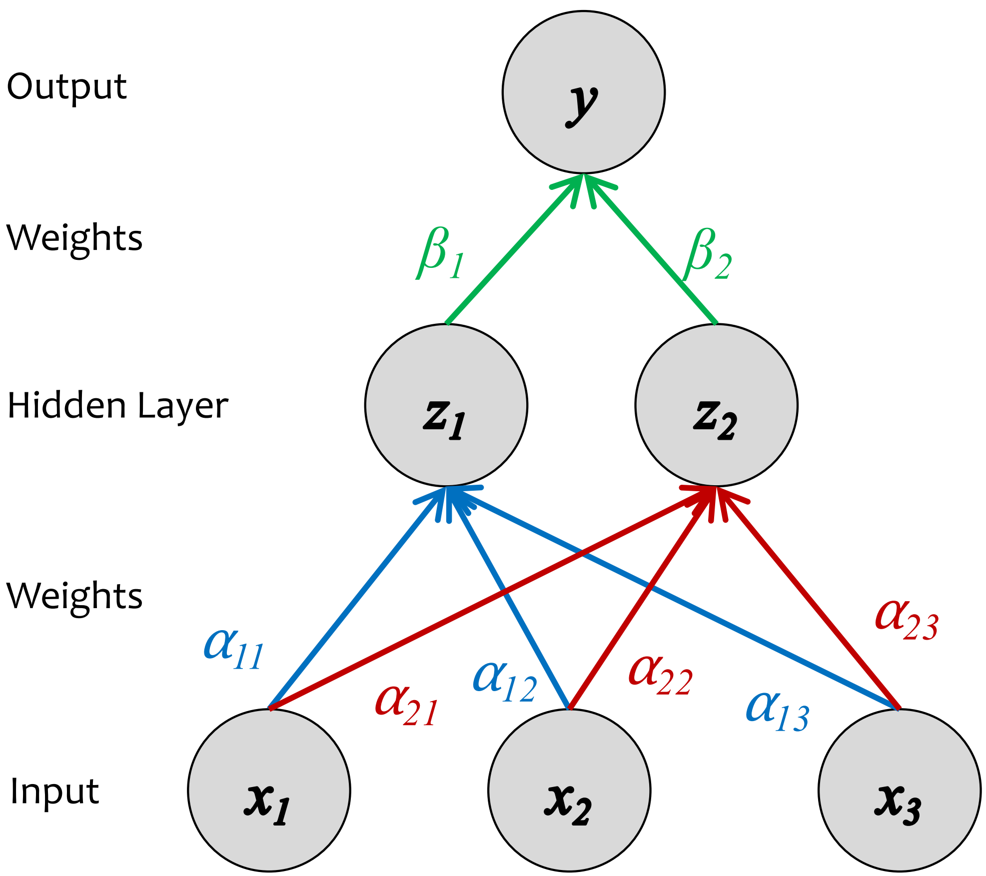
      $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \gamma \mathbf{g}_{\boldsymbol{\beta}}$

    Evaluate training mean cross-entropy $J_{\mathcal{D}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$

    Evaluate test mean cross-entropy $J_{\mathcal{D}_t}(\boldsymbol{\alpha}, \boldsymbol{\beta})$

  return parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$

Training Backpropagation



(E) Output (sigmoid)
$y = \frac{1}{1+\exp(-b)}$

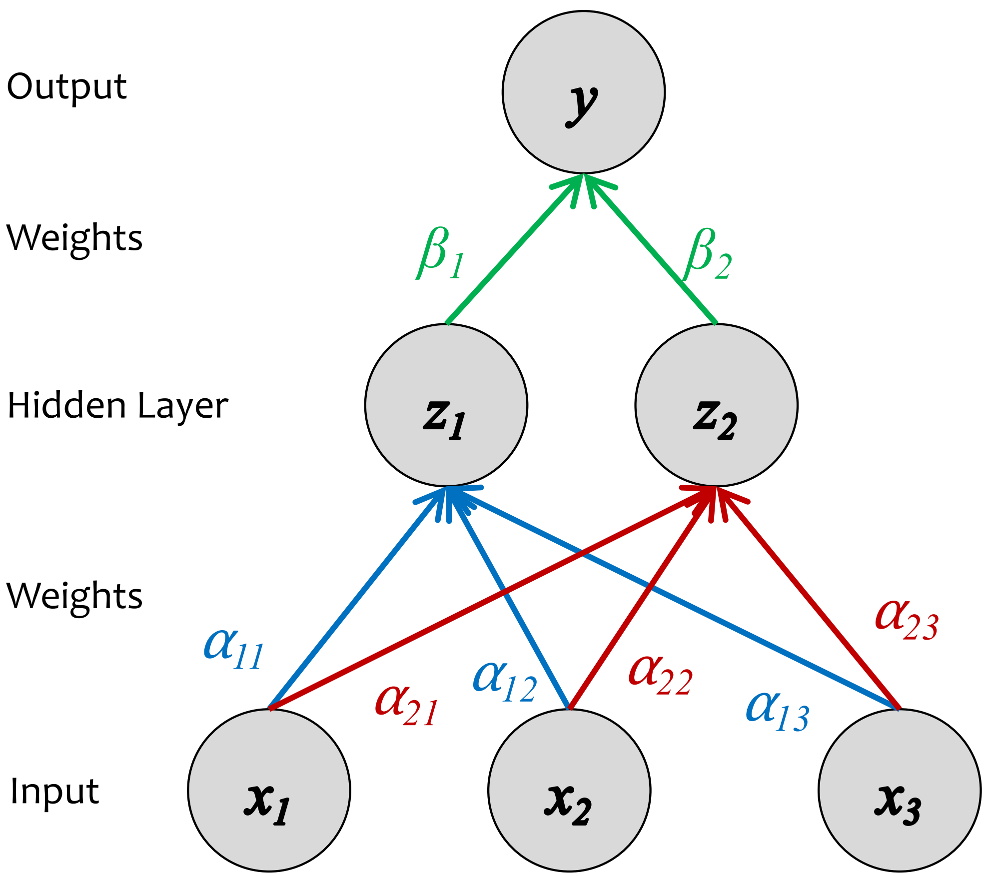
(D) Output (linear)
$b = \sum_{j=0}^{D} \beta_j z_j$

(C) Hidden (sigmoid)
$z_j = \frac{1}{1+\exp(-a_j)}, \quad \forall j$

(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \quad \forall j$

(A) Input
Given $x_i, \quad \forall i$

Training Backpropagation



(F) **Loss**
$J = \frac{1}{2}(y - y^*)^2$

↑

(E) **Output (sigmoid)**
$y = \frac{1}{1+\exp(-b)}$

↑

(D) **Output (linear)**
$b = \sum_{j=0}^{D} \beta_j z_j$

↑

(C) **Hidden (sigmoid)**
$z_j = \frac{1}{1+\exp(-a_j)}, \;\; \forall j$

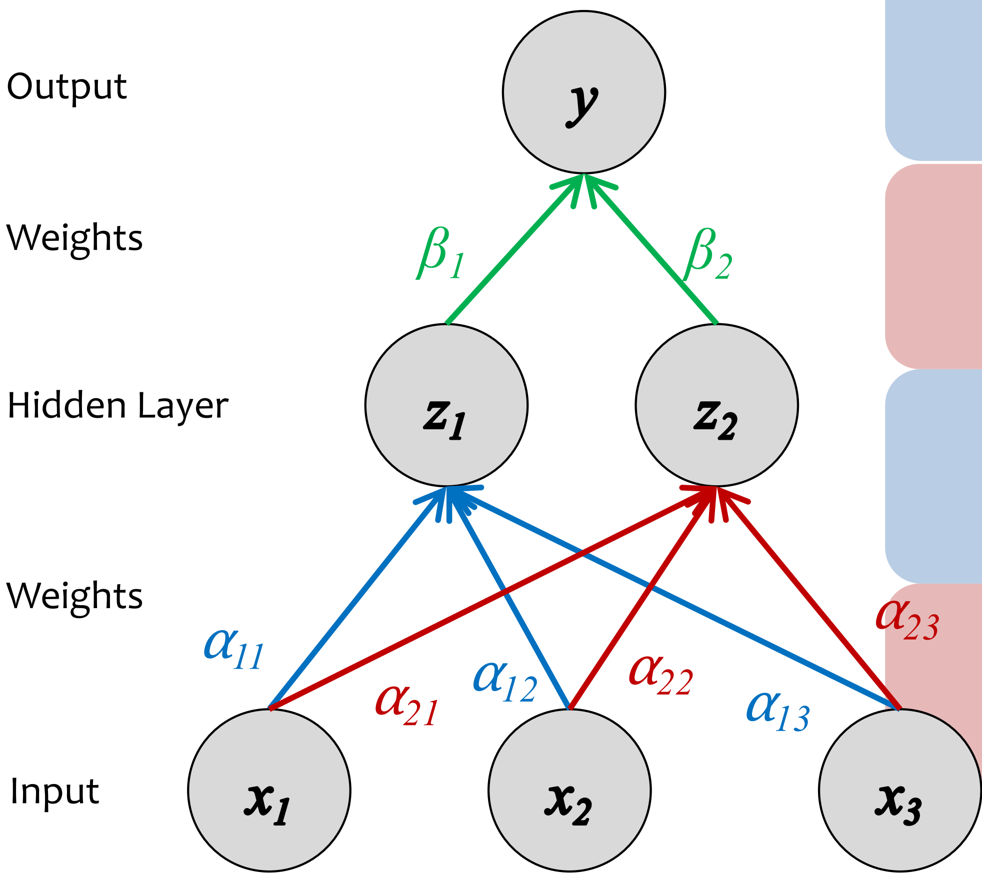
↑

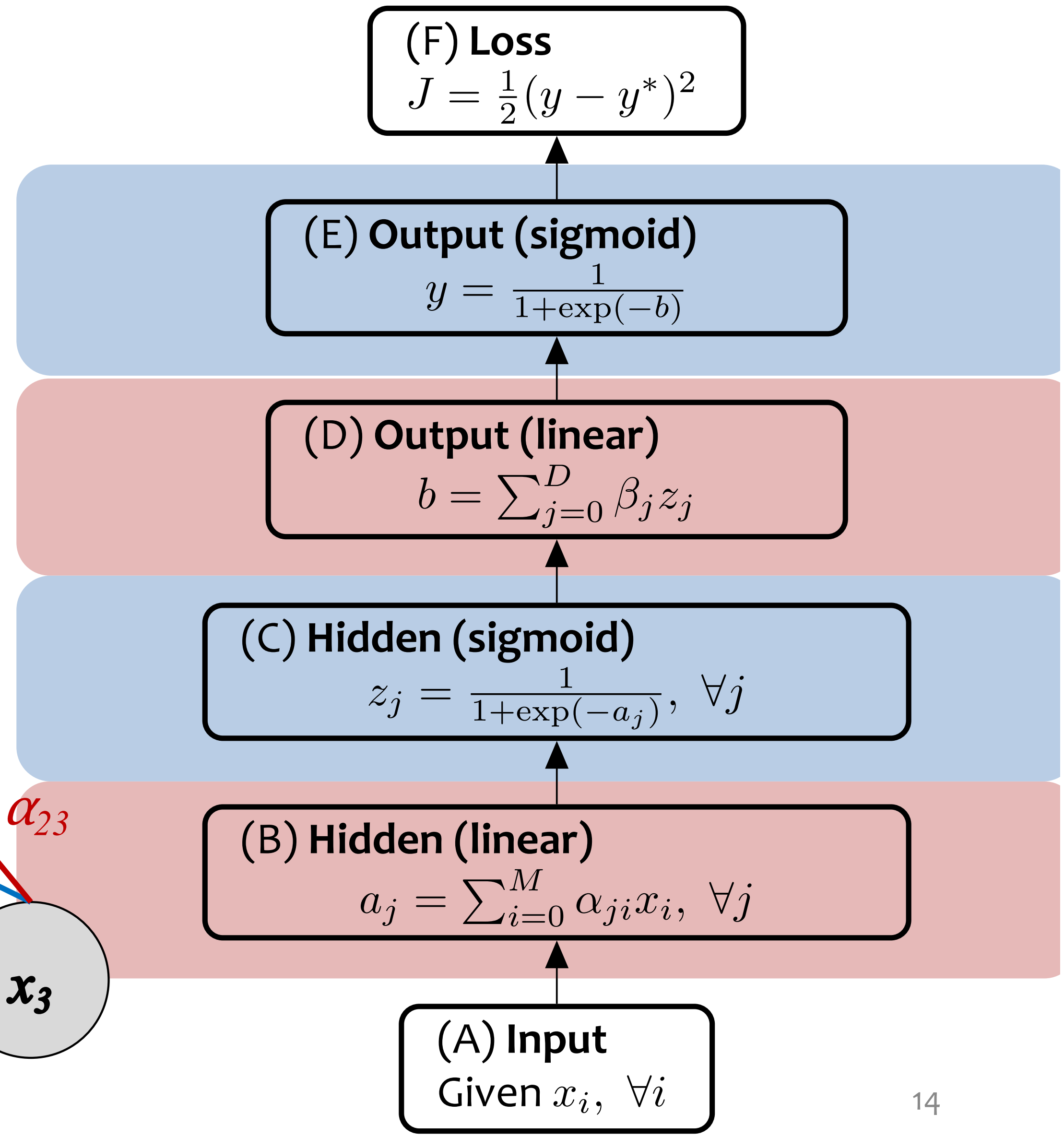
(B) **Hidden (linear)**
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \;\; \forall j$

↑

(A) **Input**
Given $x_i, \;\; \forall i$

Training Backpropagation

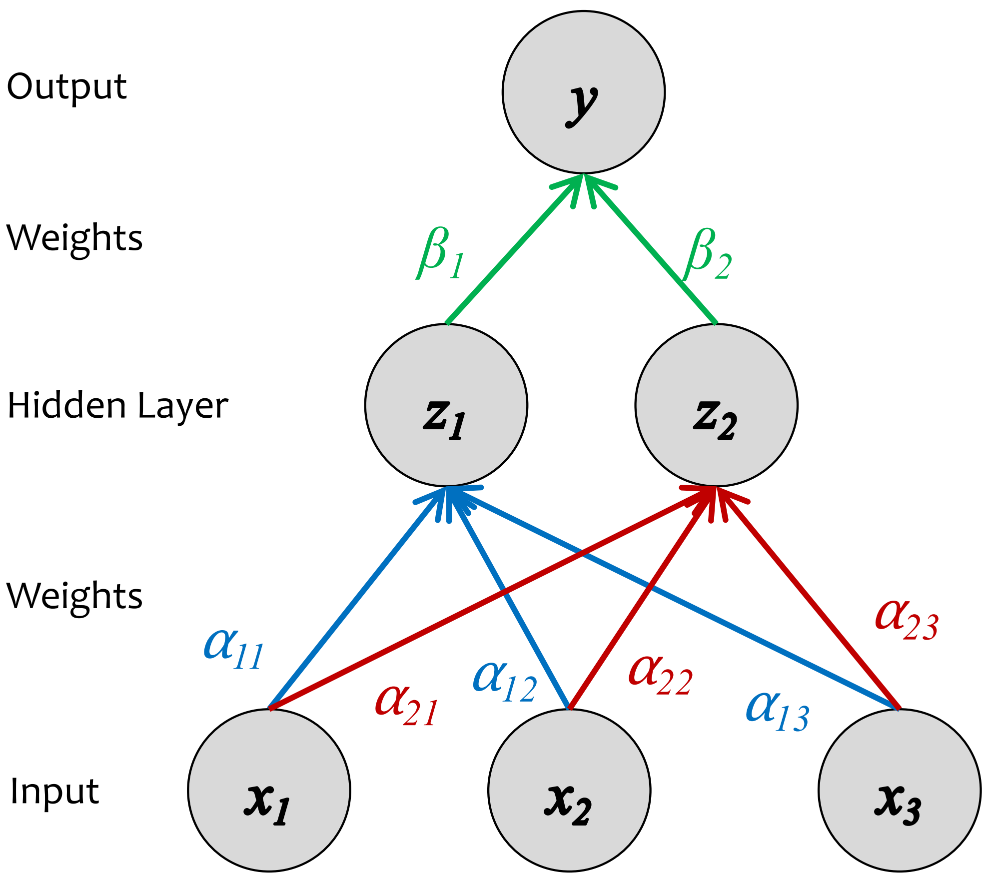


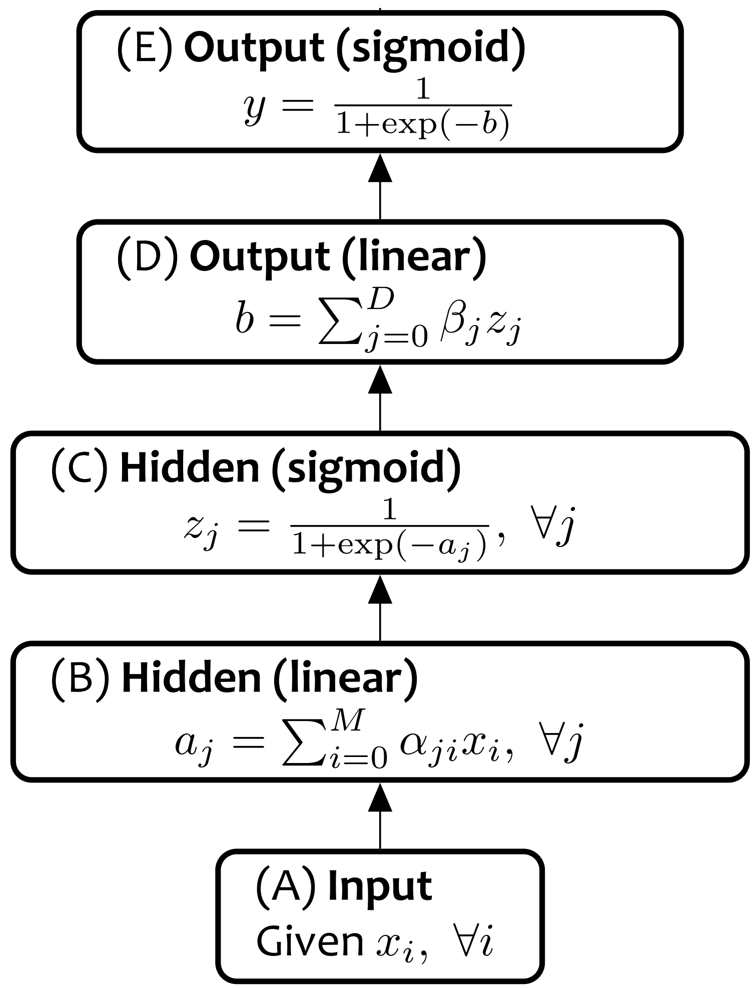


A 1-Hidden Layer Neural Network

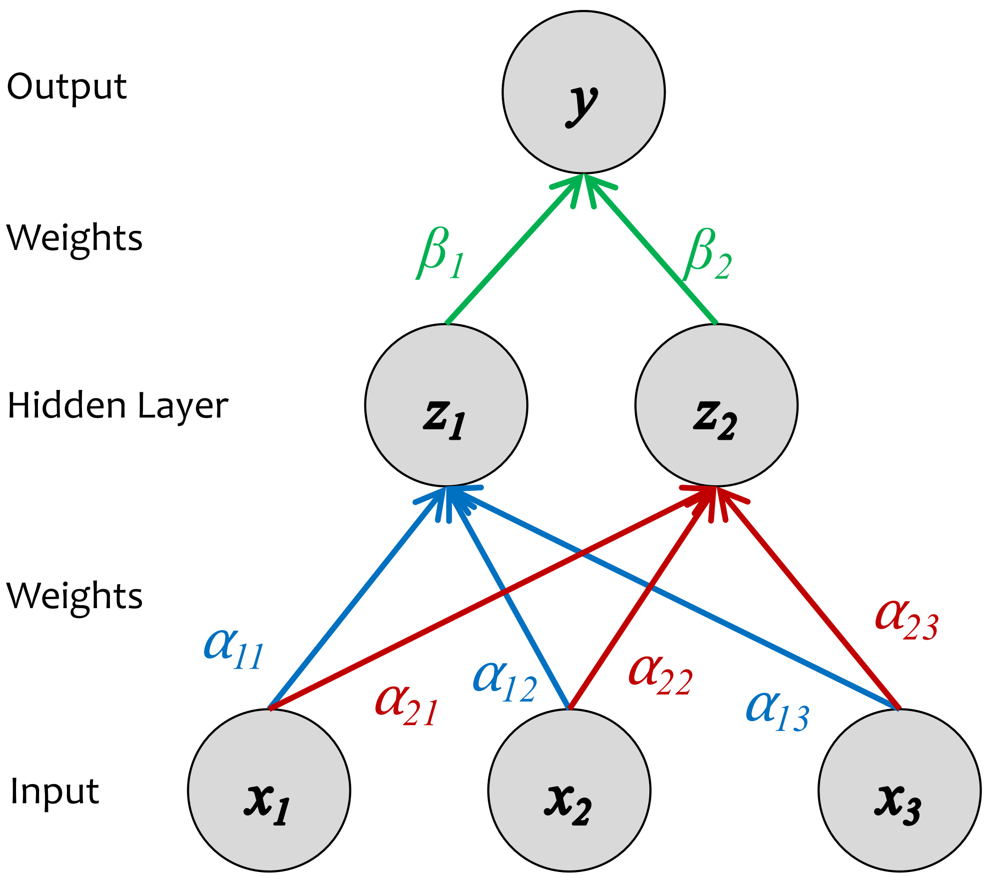
BACKPROPAGATION FOR A NEURAL NETWORK

Backpropagation





Training Backpropagation



(F) Loss
$J = \frac{1}{2}(y - y^*)^2$

↑

(E) Output (sigmoid)
$y = \frac{1}{1+\exp(-b)}$

↑

(D) Output (linear)
$b = \sum_{j=0}^{D} \beta_j z_j$

↑

(C) Hidden (sigmoid)
$z_j = \frac{1}{1+\exp(-a_j)}, \ \forall j$

↑

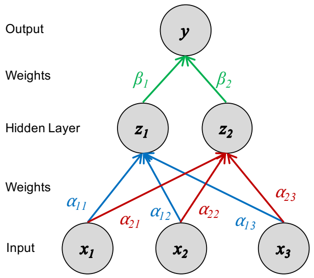
(B) Hidden (linear)
$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i, \ \forall j$

↑

(A) Input
Given $x_i, \ \forall i$

Training Backpropagation

Case 2: Neural Network



Forward

$$J = y^* \log y + (1 - y^*) \log(1 - y)$$

$$y = \frac{1}{1 + \exp(-b)}$$

$$b = \sum_{j=0}^{D} \beta_j z_j$$

$$z_j = \frac{1}{1 + \exp(-a_j)}$$

$$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i$$

Backward

$$\frac{dJ}{dy} = \frac{y^*}{y} + \frac{(1 - y^*)}{y - 1}$$

$$\frac{dJ}{db} = \frac{dJ}{dy}\frac{dy}{db}, \quad \frac{dy}{db} = \frac{\exp(-b)}{(\exp(-b) + 1)^2}$$

$$\frac{dJ}{d\beta_j} = \frac{dJ}{db}\frac{db}{d\beta_j}, \quad \frac{db}{d\beta_j} = z_j$$

$$\frac{dJ}{dz_j} = \frac{dJ}{db}\frac{db}{dz_j}, \quad \frac{db}{dz_j} = \beta_j$$

$$\frac{dJ}{da_j} = \frac{dJ}{dz_j}\frac{dz_j}{da_j}, \quad \frac{dz_j}{da_j} = \frac{\exp(-a_j)}{(\exp(-a_j) + 1)^2}$$

$$\frac{dJ}{d\alpha_{ji}} = \frac{dJ}{da_j}\frac{da_j}{d\alpha_{ji}}, \quad \frac{da_j}{d\alpha_{ji}} = x_i$$

$$\frac{dJ}{dx_i} = \sum_{j=0}^{D} \frac{dJ}{da_j}\frac{da_j}{dx_i}, \quad \frac{da_j}{dx_i} = \alpha_{ji}$$

Training Backpropagation

Case 2:

	Forward	Backward
Loss	$J = y^* \log y + (1 - y^*) \log(1 - y)$	$\frac{dJ}{dy} = \frac{y^*}{y} + \frac{(1 - y^*)}{y - 1}$
Sigmoid	$y = \frac{1}{1 + \exp(-b)}$	$\frac{dJ}{db} = \frac{dJ}{dy}\frac{dy}{db}, \quad \frac{dy}{db} = \frac{\exp(-b)}{(\exp(-b) + 1)^2}$
Linear	$b = \sum_{j=0}^{D} \beta_j z_j$	$\frac{dJ}{d\beta_j} = \frac{dJ}{db}\frac{db}{d\beta_j}, \quad \frac{db}{d\beta_j} = z_j$ $\frac{dJ}{dz_j} = \frac{dJ}{db}\frac{db}{dz_j}, \quad \frac{db}{dz_j} = \beta_j$
Sigmoid	$z_j = \frac{1}{1 + \exp(-a_j)}$	$\frac{dJ}{da_j} = \frac{dJ}{dz_j}\frac{dz_j}{da_j}, \quad \frac{dz_j}{da_j} = \frac{\exp(-a_j)}{(\exp(-a_j) + 1)^2}$
Linear	$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i$	$\frac{dJ}{d\alpha_{ji}} = \frac{dJ}{da_j}\frac{da_j}{d\alpha_{ji}}, \quad \frac{da_j}{d\alpha_{ji}} = x_i$ $\frac{dJ}{dx_i} = \sum_{j=0}^{D} \frac{dJ}{da_j}\frac{da_j}{dx_i}, \quad \frac{da_j}{dx_i} = \alpha_{ji}$

Derivative of a Sigmoid

First suppose that

$$s = \frac{1}{1 + \exp(-b)} \tag{1}$$

To obtain the simplified form of the derivative of a sigmoid.

$$\begin{aligned}
\frac{ds}{db} &= \frac{\exp(-b)}{(\exp(-b)+1)^2} && (2)\\
&= \frac{\exp(-b)+1-1}{(\exp(-b)+1+1-1)^2} && (3)\\
&= \frac{\exp(-b)+1-1}{(\exp(-b)+1)^2} && (4)\\
&= \frac{\exp(-b)+1}{(\exp(-b)+1)^2} - \frac{1}{(\exp(-b)+1)^2} && (5)\\
&= \frac{1}{(\exp(-b)+1)} - \frac{1}{(\exp(-b)+1)^2} && (6)\\
&= \frac{1}{(\exp(-b)+1)} - \left(\frac{1}{(\exp(-b)+1)}\frac{1}{(\exp(-b)+1)}\right) && (7)\\
&= \frac{1}{(\exp(-b)+1)}\left(1 - \frac{1}{(\exp(-b)+1)}\right) && (8)\\
&= s(1-s) && (9)
\end{aligned}$$

Training Backpropagation

Case 2:

	Forward	Backward
Loss	$J = y^* \log y + (1 - y^*) \log(1 - y)$	$\frac{dJ}{dy} = \frac{y^*}{y} + \frac{(1-y^*)}{y-1}$
Sigmoid	$y = \frac{1}{1+\exp(-b)}$	$\frac{dJ}{db} = \frac{dJ}{dy}\frac{dy}{db}, \quad \frac{dy}{db} = \frac{\exp(-b)}{(\exp(-b)+1)^2}$
Linear	$b = \sum_{j=0}^{D} \beta_j z_j$	$\frac{dJ}{d\beta_j} = \frac{dJ}{db}\frac{db}{d\beta_j}, \quad \frac{db}{d\beta_j} = z_j$ $\frac{dJ}{dz_j} = \frac{dJ}{db}\frac{db}{dz_j}, \quad \frac{db}{dz_j} = \beta_j$
Sigmoid	$z_j = \frac{1}{1+\exp(-a_j)}$	$\frac{dJ}{da_j} = \frac{dJ}{dz_j}\frac{dz_j}{da_j}, \quad \frac{dz_j}{da_j} = \frac{\exp(-a_j)}{(\exp(-a_j)+1)^2}$
Linear	$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i$	$\frac{dJ}{d\alpha_{ji}} = \frac{dJ}{da_j}\frac{da_j}{d\alpha_{ji}}, \quad \frac{da_j}{d\alpha_{ji}} = x_i$ $\frac{dJ}{dx_i} = \sum_{j=0}^{D} \frac{dJ}{da_j}\frac{da_j}{dx_i}, \quad \frac{da_j}{dx_i} = \alpha_{ji}$

Training

Backpropagation

Case 2:

	Forward	Backward
Loss	$J = y^* \log y + (1-y^*) \log(1-y)$	$\frac{dJ}{dy} = \frac{y^*}{y} + \frac{(1-y^*)}{y-1}$
Sigmoid	$y = \frac{1}{1+\exp(-b)}$	$\frac{dJ}{db} = \frac{dJ}{dy}\frac{dy}{db}, \ \frac{dy}{db} = y(1-y)$
Linear	$b = \sum_{j=0}^{D} \beta_j z_j$	$\frac{dJ}{d\beta_j} = \frac{dJ}{db}\frac{db}{d\beta_j}, \ \frac{db}{d\beta_j} = z_j$ $\frac{dJ}{dz_j} = \frac{dJ}{db}\frac{db}{dz_j}, \ \frac{db}{dz_j} = \beta_j$
Sigmoid	$z_j = \frac{1}{1+\exp(-a_j)}$	$\frac{dJ}{da_j} = \frac{dJ}{dz_j}\frac{dz_j}{da_j}, \ \frac{dz_j}{da_j} = z_j(1-z_j)$
Linear	$a_j = \sum_{i=0}^{M} \alpha_{ji} x_i$	$\frac{dJ}{d\alpha_{ji}} = \frac{dJ}{da_j}\frac{da_j}{d\alpha_{ji}}, \ \frac{da_j}{d\alpha_{ji}} = x_i$ $\frac{dJ}{dx_i} = \sum_{j=0}^{D}\frac{dJ}{da_j}\frac{da_j}{dx_i}, \ \frac{da_j}{dx_i} = \alpha_{ji}$

SGD with Backprop

Example: 1-Hidden Layer Neural Network

Algorithm 1 Stochastic Gradient Descent (SGD)

procedure SGD(Training data $\mathcal{D}$, test data $\mathcal{D}_t$)
    Initialize parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$
    for $e \in \{1, 2, \ldots, E\}$ **do**
        for $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ **do**
            Compute neural network layers:
            $\mathbf{o} = \texttt{object}(\mathbf{x}, \mathbf{a}, \mathbf{b}, \mathbf{z}, \hat{\mathbf{y}}, J) = \text{NNFORWARD}(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$
            Compute gradients via backprop:
            $$\left.\begin{aligned} \mathbf{g}_{\boldsymbol{\alpha}} &= \nabla_{\boldsymbol{\alpha}} J \\ \mathbf{g}_{\boldsymbol{\beta}} &= \nabla_{\boldsymbol{\beta}} J \end{aligned}\right\} = \text{NNBACKWARD}(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{o})$$
            Update parameters:
            $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - \gamma \mathbf{g}_{\boldsymbol{\alpha}}$
            $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \gamma \mathbf{g}_{\boldsymbol{\beta}}$
        Evaluate training mean cross-entropy $J_{\mathcal{D}}(\boldsymbol{\alpha}, \boldsymbol{\beta})$
        Evaluate test mean cross-entropy $J_{\mathcal{D}_t}(\boldsymbol{\alpha}, \boldsymbol{\beta})$
    return parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}$

THE BACKPROPAGATION ALGORITHM

Backpropagation

Automatic Differentiation – Reverse Mode (aka. Backpropagation)

Forward Computation

1. Write an **algorithm** for evaluating the function $y = f(\mathbf{x})$. The algorithm defines a **directed acyclic graph**, where each variable is a node (i.e. the "**computation graph**")
2. Visit each node in **topological order**.
   For variable u_i with inputs $v_1,\ldots, v_N$
   a. Compute $u_i = g_i(v_1,\ldots, v_N)$
   b. Store the result at the node

Backward Computation (Version A)

1. **Initialize** $dy/dy = 1$.
2. Visit each node v_j in **reverse topological order**.
   Let $u_1,\ldots, u_M$ denote all the nodes with v_j as an input
   Assuming that $y = h(\mathbf{u}) = h(u_1,\ldots, u_M)$
   and $\mathbf{u} = g(\mathbf{v})$ or equivalently $u_i = g_i(v_1,\ldots, v_j,\ldots, v_N)$ for all i
   a. We already know dy/du_i for all i
   b. Compute dy/dv_j as below (Choice of algorithm ensures computing (du_i/dv_j) is easy)

$$\frac{dy}{dv_j} = \sum_{i=1}^{M} \frac{dy}{du_i}\frac{du_i}{dv_j}$$

Return partial derivatives dy/du_i for all variables

Automatic Differentiation – Reverse Mode (aka. Backpropagation)

Forward Computation

1. Write an **algorithm** for evaluating the function y = f(**x**). The algorithm defines a **directed acyclic graph**, where each variable is a node (i.e. the "**computation graph**")
2. Visit each node in **topological order**.
   For variable u_i with inputs $v_1, \ldots, v_N$
   a. Compute $u_i = g_i(v_1, \ldots, v_N)$
   b. Store the result at the node

Backward Computation (Version B)

1. **Initialize** all partial derivatives dy/du_j to 0 and $dy/dy = 1$.
2. Visit each node in **reverse topological order**.
   For variable $u_i = g_i(v_1, \ldots, v_N)$
   a. We already know dy/du_i
   b. Increment dy/dv_j by $(dy/du_i)(du_i/dv_j)$
   (Choice of algorithm ensures computing (du_i/dv_j) is easy)

Return partial derivatives dy/du_i for all variables

Backpropagation

Why is the backpropagation algorithm efficient?

1. Reuses **computation from the forward pass** in the backward pass
2. Reuses **partial derivatives** throughout the backward pass (*but only if the algorithm reuses shared computation in the forward pass*)

   (Key idea: partial derivatives in the backward pass should be thought of as variables stored for reuse)

Background

A Recipe for

Gradients

Backpropagation can compute this gradient!

And it's a **special case of a more general algorithm** called reverse-mode automatic differentiation that can compute the gradient of any differentiable function efficiently!

1. Given training dat

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose each of t

– Decision functio

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

– Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

opp site the gradient)

$$\boldsymbol{\theta}^{(t} \quad {}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

MATRIX CALCULUS

Q&A

Q: Do I need to know **matrix calculus** to derive the backprop algorithms used in this class?

A: Well, we've carefully constructed our assignments so that you do **not** need to know matrix calculus.

That said, it's pretty handy. So we *added matrix calculus to our learning objectives* for backprop.

Matrix Calculus

Let $y, x \in \mathbb{R}$ be scalars, $\mathbf{y} \in \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^P$ be vectors, and $\mathbf{Y} \in \mathbb{R}^{M \times N}$ and $\mathbf{X} \in \mathbb{R}^{P \times Q}$ be matrices

Numerator

Denominator

Types of Derivatives	**scalar**	**vector**	**matrix**
scalar	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{Y}}{\partial x}$
vector	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{x}}$
matrix	$\frac{\partial y}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{X}}$	$\frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$

Matrix Calculus

Types of Derivatives	**scalar**
scalar	$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x}\right]$
vector	$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix}$
matrix	$\frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial X_{11}} & \frac{\partial y}{\partial X_{12}} & \cdots & \frac{\partial y}{\partial X_{1Q}} \\ \frac{\partial y}{\partial X_{21}} & \frac{\partial y}{\partial X_{22}} & \cdots & \frac{\partial y}{\partial X_{2Q}} \\ \vdots & & & \vdots \\ \frac{\partial y}{\partial X_{P1}} & \frac{\partial y}{\partial X_{P2}} & \cdots & \frac{\partial y}{\partial X_{PQ}} \end{bmatrix}$

Matrix Calculus

Types of Derivatives	**scalar**	**vector**
scalar	$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x} \right]$	$\frac{\partial \mathbf{y}}{\partial x} = \left[\frac{\partial y_1}{\partial x} \quad \frac{\partial y_2}{\partial x} \quad \cdots \quad \frac{\partial y_N}{\partial x} \right]$
vector	$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_N}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_N}{\partial x_2} \\ \vdots & & & \\ \frac{\partial y_1}{\partial x_P} & \frac{\partial y_2}{\partial x_P} & \cdots & \frac{\partial y_N}{\partial x_P} \end{bmatrix}$

Matrix Calculus

Whenever you read about matrix calculus, you'll be confronted with two layout conventions:

Let $y, x \in \mathbb{R}$ be scalars, $\mathbf{y} \in \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^P$ be vectors.

1. In numerator layout:

$$\frac{\partial y}{\partial \mathbf{x}} \text{ is a } 1 \times P \text{ matrix, i.e. a row vector}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \text{ is an } M \times P \text{ matrix}$$

2. In denominator layout:

$$\frac{\partial y}{\partial \mathbf{x}} \text{ is a } P \times 1 \text{ matrix, i.e. a column vector}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \text{ is an } P \times M \text{ matrix}$$

In this course, **we use denominator layout.**

Why? This ensures that our gradients of the objective function with respect to some subset of parameters are the same shape as those parameters.

Matrix Calculus

Common Vector Derivatives

Let $\frac{\partial f(\vec{x})}{\partial \vec{x}} = \nabla_x f(\vec{x})$ be the vector derivative of f, $B \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^m$

Scalar Derivative		Vector Derivative	
$f(x)$	$\rightarrow \frac{\partial f}{\partial x}$	$f(x)$	$\rightarrow \frac{\partial f}{\partial \vec{x}}$
bx	$\rightarrow b$	$x^T B$	$\rightarrow B$
xb	$\rightarrow b$	$x^T b$	$\rightarrow b$
x^2	$\rightarrow 2x$	$x^T x$	$\rightarrow 2x$
bx^2	$\rightarrow 2bx$	$x^T B x$	$\rightarrow 2Bx$

(B symmetric)

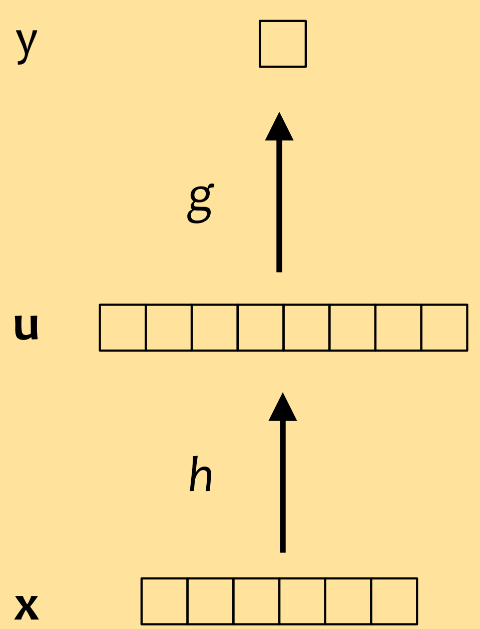
Matrix Calculus

Recall:

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_P} \end{bmatrix} \qquad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \dots & \frac{\partial y_N}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \dots & \frac{\partial y_N}{\partial x_2} \\ \vdots & & & \\ \frac{\partial y_1}{\partial x_P} & \frac{\partial y_2}{\partial x_P} & \dots & \frac{\partial y_N}{\partial x_P} \end{bmatrix}$$

Question:

Suppose y = g(**u**) and **u** = h(**x**)



Which of the following is the correct definition of the chain rule?

Answer:

$$\frac{\partial y}{\partial \mathbf{x}} = \dots$$

A. $\frac{\partial y}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$

B. $\frac{\partial y}{\partial \mathbf{u}}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$

C. $\frac{\partial y}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}}^T$

D. $\frac{\partial y}{\partial \mathbf{u}}^T \frac{\partial \mathbf{u}}{\partial \mathbf{x}}^T$

E. $(\frac{\partial y}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{x}})^T$

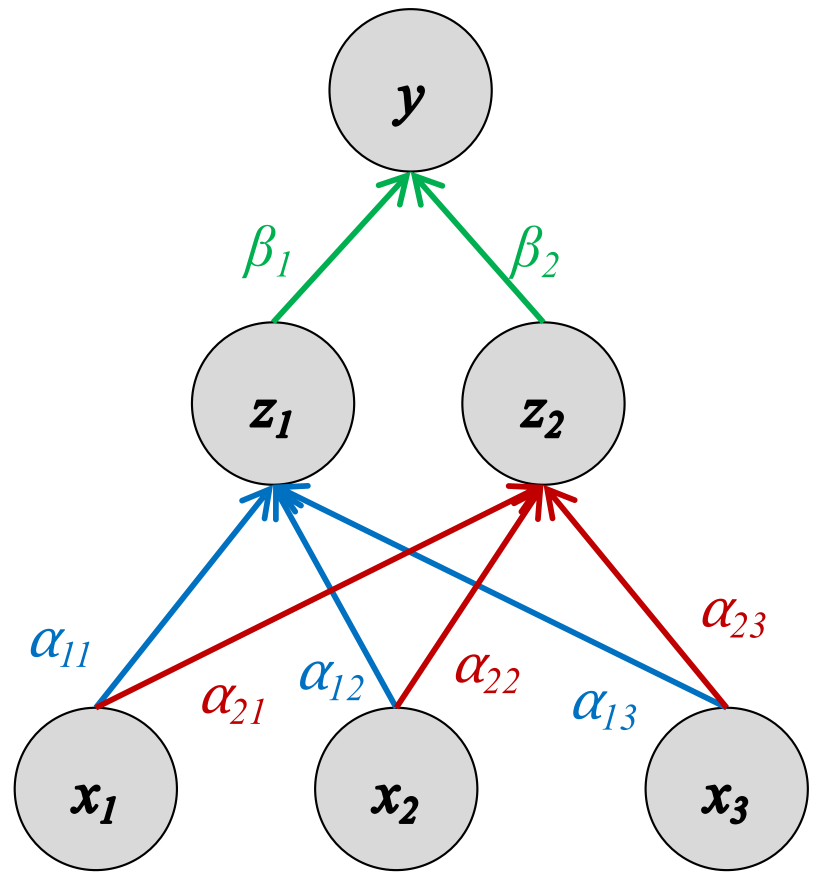
F. None of the above

DRAWING A NEURAL NETWORK

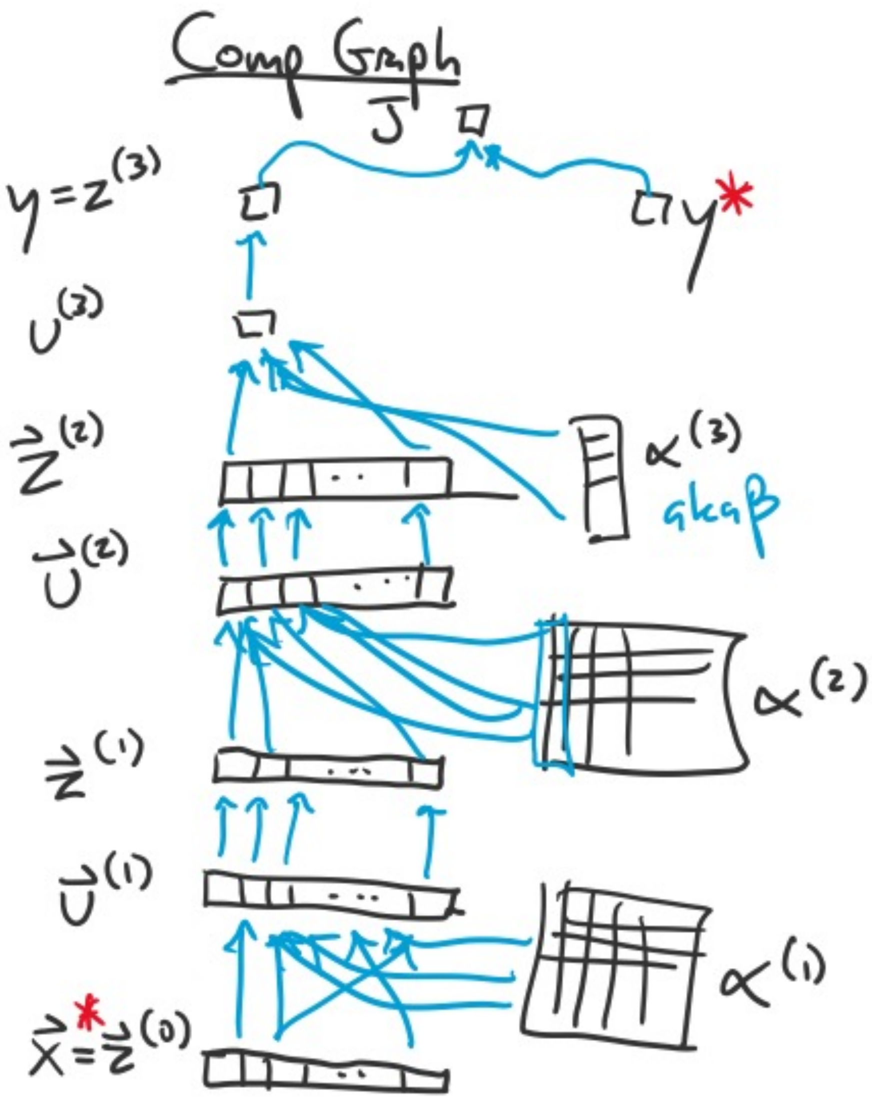
Ways of Drawing Neural Networks

Neural Network Diagram

- The diagram represents a neural network
- Nodes are **circles**
- One node per hidden unit
- Node is labeled with the variable corresponding to the hidden unit
- For a fully connected feed-forward neural network, a hidden unit is a nonlinear function of nodes in the previous layer
- *Edges are directed*
- Each edge is labeled with its weight (side note: we should be careful about ascribing how a matrix can be used to indicate the labels of the edges and pitfalls there)
- Other details:
  - Following standard convention, the intercept term is NOT shown as a node, but rather is assumed to be part of the non-linear function that yields a hidden unit. (i.e. its weight does NOT appear in the picture anywhere)
  - The diagram does NOT include any nodes related to the loss computation



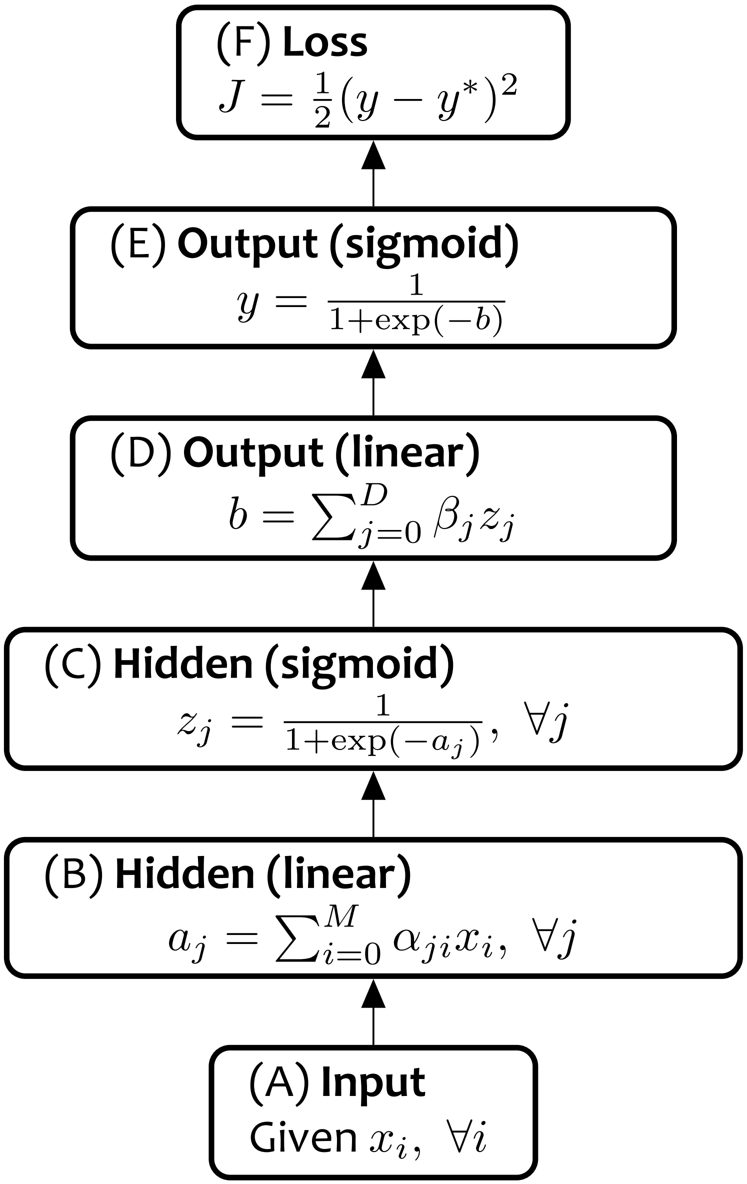
Ways of Drawing Neural Networks



Computation Graph

- The diagram represents an algorithm
- Nodes are **rectangles**
- One node per intermediate variable in the algorithm
- Node is labeled with the function that it computes (inside the box) and also the variable name (outside the box)
- *Edges are directed*
- Edges do not have labels (since they don't need them)
- For neural networks:
  - Each intercept term should appear as a node (if it's not folded in somewhere)
  - Each parameter should appear as a node
  - Each constant, e.g. a true label or a feature vector should appear in the graph
  - It's perfectly fine to include the loss

Ways of Drawing Neural Networks



Computation Graph

- The diagram represents an algorithm
- Nodes are **rectangles**
- One node per intermediate variable in the algorithm
- Node is labeled with the function that it computes (inside the box) and also the variable name (outside the box)
- *Edges are directed*
- Edges do not have labels (since they don't need them)
- For neural networks:
  - Each intercept term should appear as a node (if it's not folded in somewhere)
  - Each parameter should appear as a node
  - Each constant, e.g. a true label or a feature vector should appear in the graph
  - It's perfectly fine to include the loss

Ways of Drawing Neural Networks

Neural Network Diagram

- The diagram represents a neural network
- Nodes are **circles**
- One node per hidden unit
- Node is labeled with the variable corresponding to the hidden unit
- For a fully connected feed-forward neural network, a hidden unit is a nonlinear function of nodes in the previous layer
- *Edges are directed*
- Each edge is labeled with its weight (side note: we should be careful about ascribing how a matrix can be used to indicate the labels of the edges and pitfalls there)
- Other details:
  - Following standard convention, the intercept term is NOT shown as a node, but rather is assumed to be part of the non-linear function that yields a hidden unit. (i.e. its weight does NOT appear in the picture anywhere)
  - The diagram does NOT include any nodes related to the loss computation

Computation Graph

- The diagram represents an algorithm
- Nodes are **rectangles**
- One node per intermediate variable in the algorithm
- Node is labeled with the function that it computes (inside the box) and also the variable name (outside the box)
- *Edges are directed*
- Edges do not have labels (since they don't need them)
- For neural networks:
  - Each intercept term should appear as a node (if it's not folded in somewhere)
  - Each parameter should appear as a node
  - Each constant, e.g. a true label or a feature vector should appear in the graph
  - It's perfectly fine to include the loss

Important!

Some of these conventions are specific to 10-301/601. The literature abounds with varations on these conventions, but it's helpful to have *some* distinction nonetheless.

Summary

1. **Neural Networks…**
   – provide a way of learning features
   – are highly nonlinear prediction functions
   – (can be) a highly parallel network of logistic regression classifiers
   – discover useful hidden representations of the input
2. **Backpropagation…**
   – provides an efficient way to compute gradients
   – is a special case of reverse-mode automatic differentiation

Backprop Objectives

You should be able to…

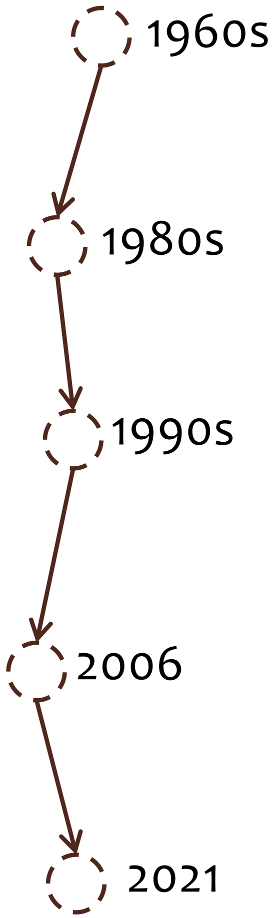
- Differentiate between a neural network diagram and a computation graph
- Construct a computation graph for a function as specified by an algorithm
- Carry out the backpropagation on an arbitrary computation graph
- Construct a computation graph for a neural network, identifying all the given and intermediate quantities that are relevant
- Instantiate the backpropagation algorithm for a neural network
- Instantiate an optimization method (e.g. SGD) and a regularizer (e.g. L2) when the parameters of a model are comprised of several matrices corresponding to different layers of a neural network
- Apply the empirical risk minimization framework to learn a neural network
- Use the finite difference method to evaluate the gradient of a function
- Identify when the gradient of a function can be computed at all and when it can be computed efficiently
- Employ basic matrix calculus to compute vector/matrix/tensor derivatives.

DEEP LEARNING

Why is everyone talking about Deep Learning?

- Because a lot of money is invested in it…
  - DeepMind: Acquired by Google for **$400 million**
  - Deep Learning startups command **millions of VC dollars**
  - Demand for deep learning engineers continually outpaces supply
- Because it made the **front page** of the New York Times

Why is everyone talking about Deep Learning?



Deep learning:

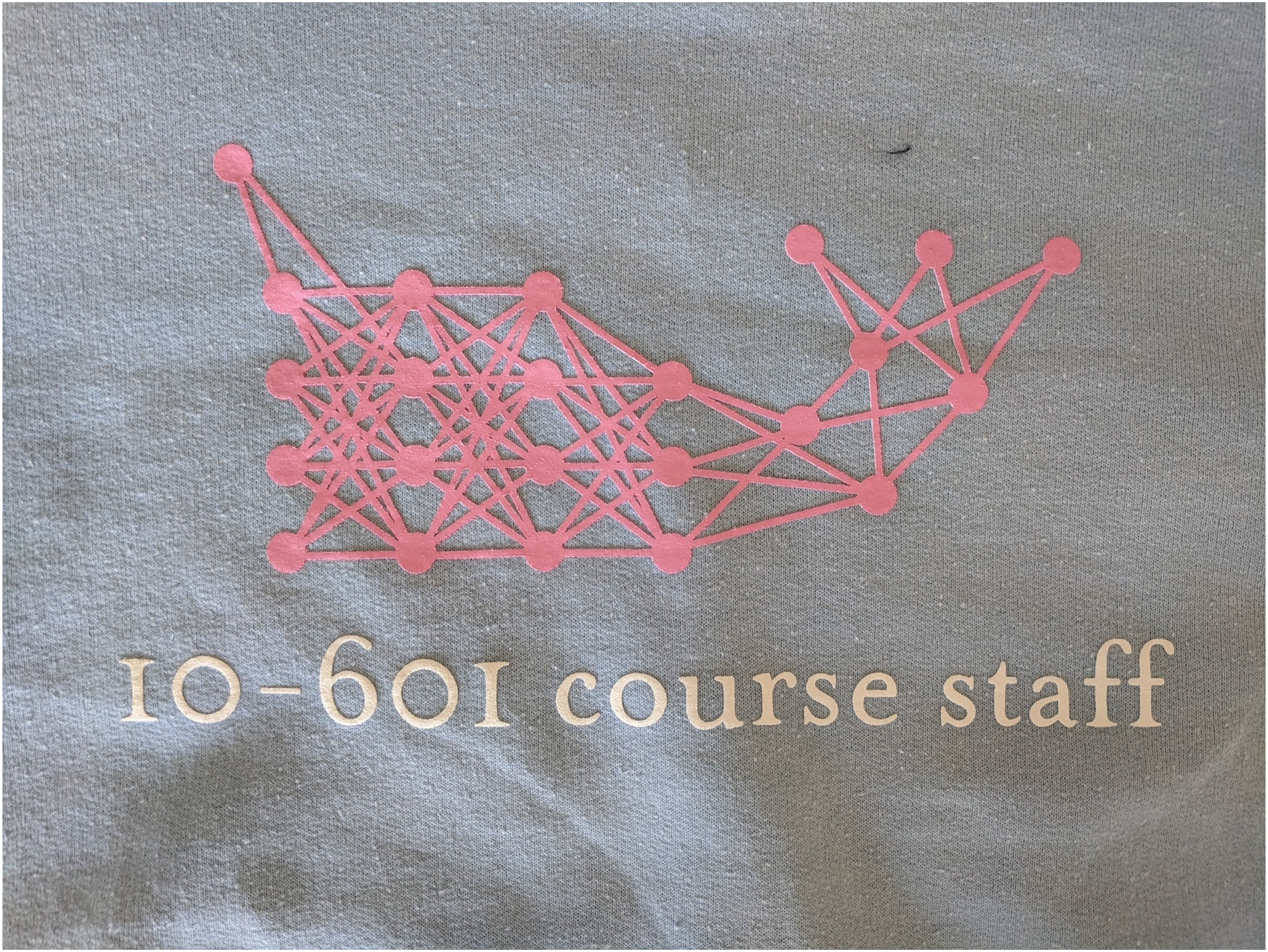
- Has won numerous pattern recognition competitions
- Does so with minimal feature engineering

This wasn't always the case!

Since 1980s: Form of models hasn't changed much, but lots of new tricks…

- More hidden units
- Better (online) optimization
- New nonlinear functions (ReLUs)
- Faster computers (CPUs and GPUs)

FIRST EXAMPLE OF A DEEP NETWORK


10-601 course staff

BACKGROUND: HUMAN LANGUAGE TECHNOLOGIES

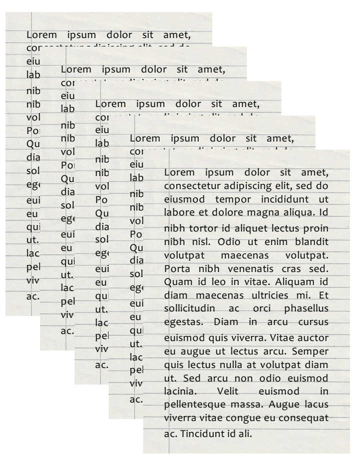
Human Language Technologies

Speech Recognition

Machine Translation

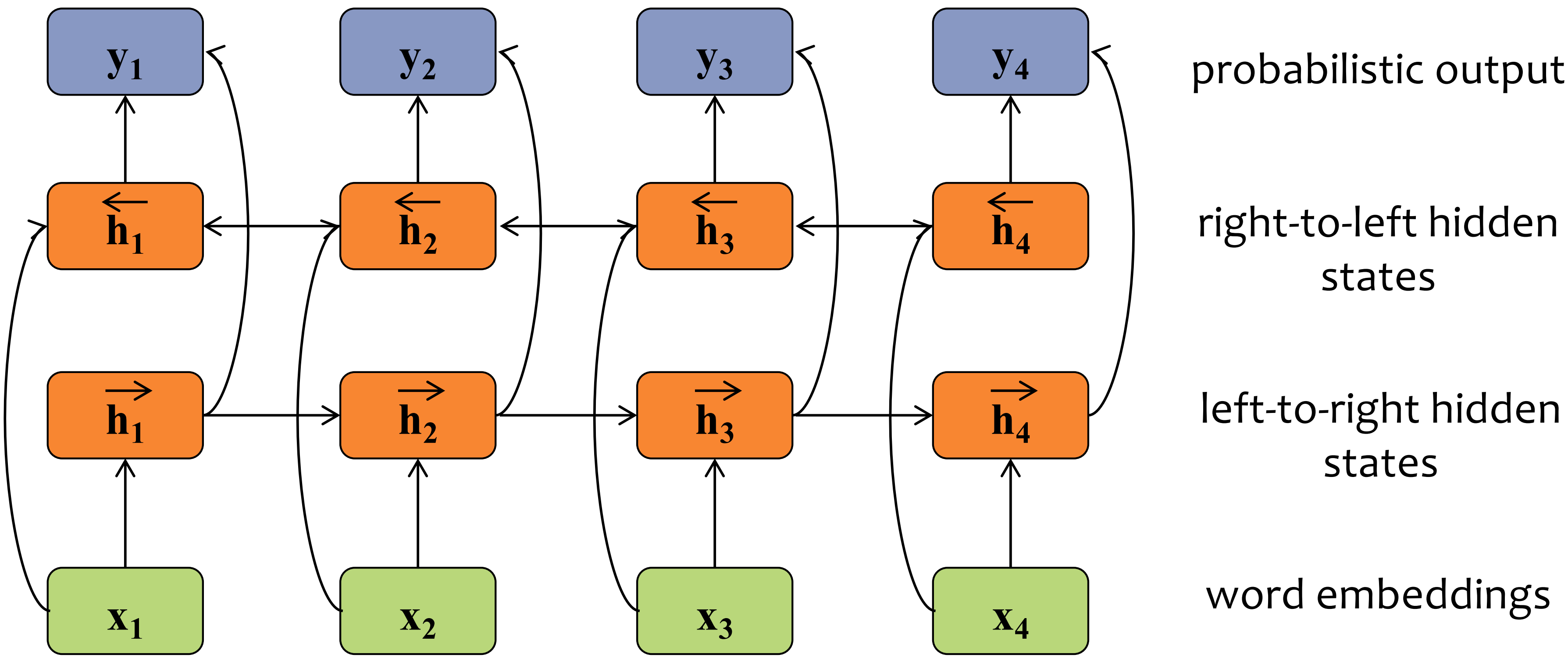
기계 번역은 특히 영어와 한국어와 같은 언어 쌍의 경우 매우 어렵습니다.

Summarization



Bidirectional RNN

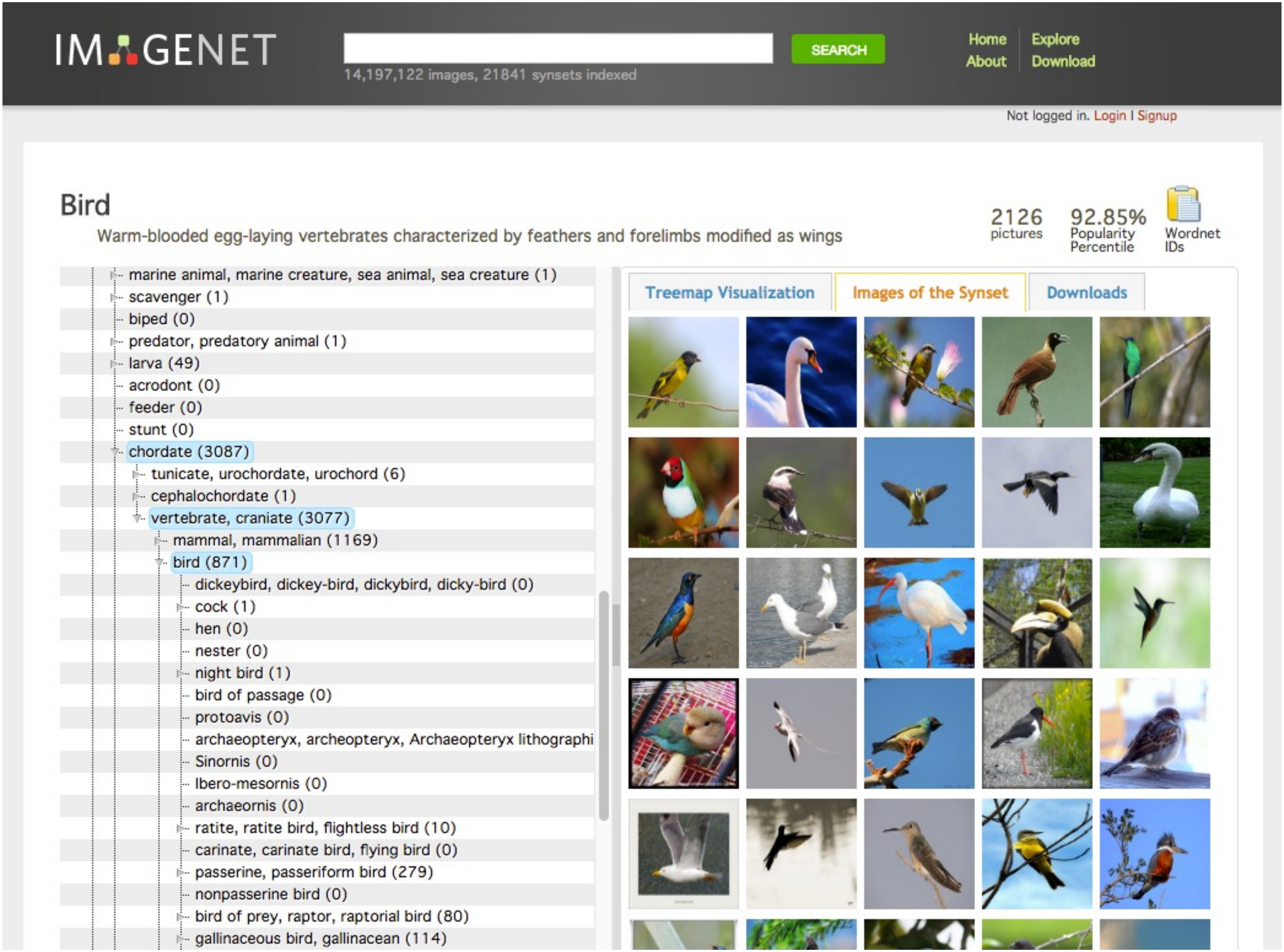
RNNs are a now commonplace backbone in deep learning approaches to natural language processing

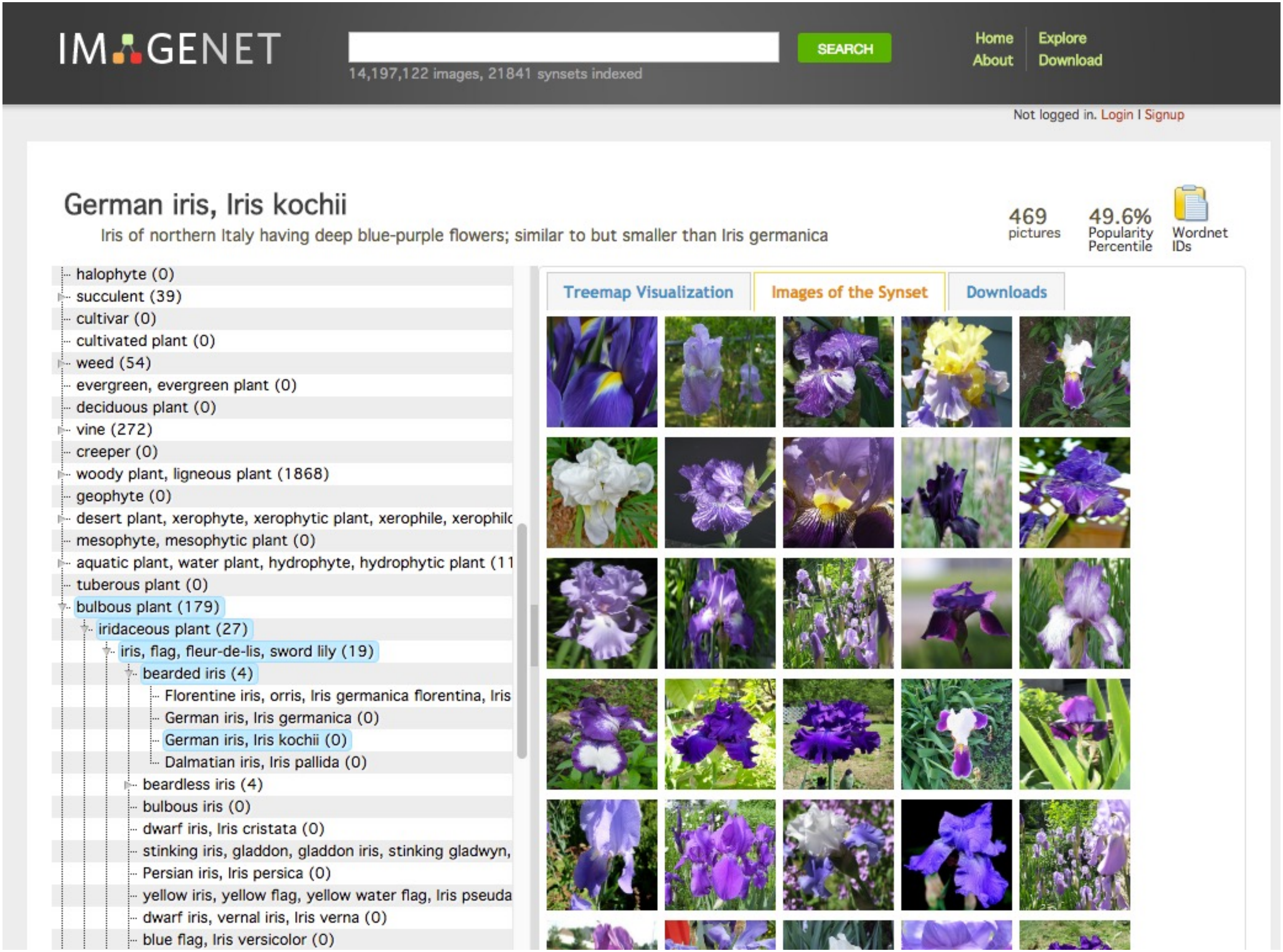


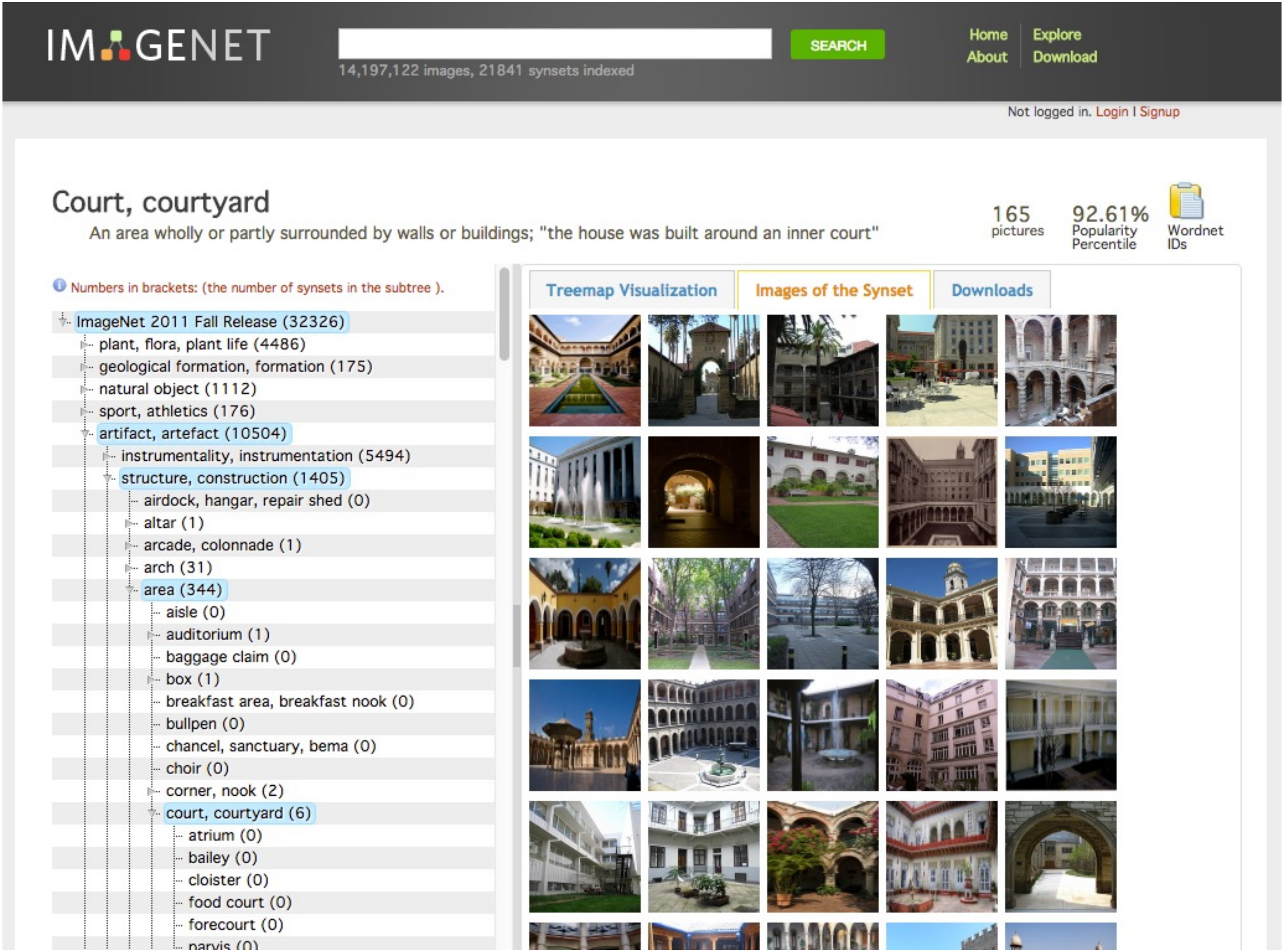
BACKGROUND: COMPUTER VISION

Example: Image Classification

- ImageNet LSVRC-2011 contest:
  - **Dataset:** 1.2 million labeled images, 1000 classes
  - **Task:** Given a new image, label it with the correct class
  - **Multiclass** classification problem
- Examples from http://image-net.org/


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
Bird
Warm-blooded egg-laying vertebrates characterized by feathers and forelimbs modified as wings
2126 pictures
92.85% Popularity Percentile
Wordnet IDs
marine animal, marine creature, sea animal, sea creature (1)
scavenger (1)
biped (0)
predator, predatory animal (1)
larva (49)
acrodont (0)
feeder (0)
stunt (0)
chordate (3087)
tunicate, urochordate, urochord (6)
cephalochordate (1)
vertebrate, craniate (3077)
mammal, mammalian (1169)
bird (871)
dickeybird, dickey-bird, dickybird, dicky-bird (0)
cock (1)
hen (0)
nester (0)
night bird (1)
bird of passage (0)
protoavis (0)
archaeopteryx, archeopteryx, Archaeopteryx lithographi
Sinornis (0)
Ibero-mesornis (0)
archaeornis (0)
ratite, ratite bird, flightless bird (10)
carinate, carinate bird, flying bird (0)
passerine, passeriform bird (279)
nonpasserine bird (0)
bird of prey, raptor, raptorial bird (80)
gallinaceous bird, gallinacean (114)
Treemap Visualization
Images of the Synset
Downloads

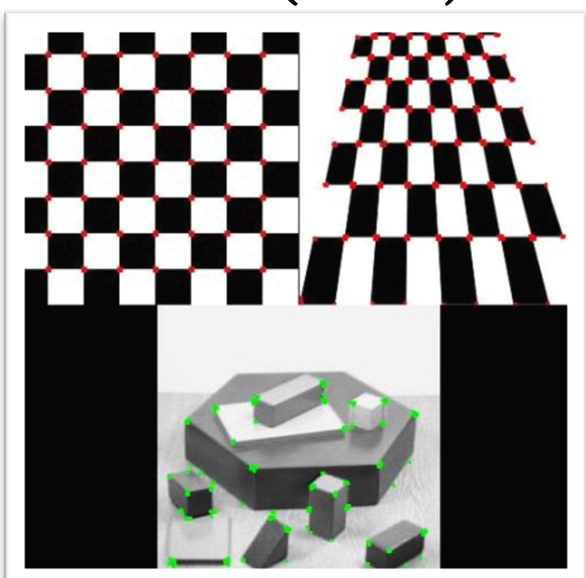

IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login | Signup
German iris, Iris kochii
Iris of northern Italy having deep blue-purple flowers; similar to but smaller than Iris germanica
469 pictures
49.6% Popularity Percentile
Wordnet IDs
halophyte (0)
succulent (39)
cultivar (0)
cultivated plant (0)
weed (54)
evergreen, evergreen plant (0)
deciduous plant (0)
vine (272)
creeper (0)
woody plant, ligneous plant (1868)
geophyte (0)
desert plant, xerophyte, xerophytic plant, xerophile, xerophilc
mesophyte, mesophytic plant (0)
aquatic plant, water plant, hydrophyte, hydrophytic plant (11
tuberous plant (0)
bulbous plant (179)
iridaceous plant (27)
iris, flag, fleur-de-lis, sword lily (19)
bearded iris (4)
Florentine iris, orris, Iris germanica florentina, Iris
German iris, Iris germanica (0)
German iris, Iris kochii (0)
Dalmatian iris, Iris pallida (0)
beardless iris (4)
bulbous iris (0)
dwarf iris, Iris cristata (0)
stinking iris, gladdon, gladdon iris, stinking gladwyn,
Persian iris, Iris persica (0)
yellow iris, yellow flag, yellow water flag, Iris pseuda
dwarf iris, vernal iris, Iris verna (0)
blue flag, Iris versicolor (0)
Treemap Visualization
Images of the Synset
Downloads


IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login I Signup
Court, courtyard
An area wholly or partly surrounded by walls or buildings; "the house was built around an inner court"
165
pictures
92.61%
Popularity Percentile
Wordnet IDs
Numbers in brackets: (the number of synsets in the subtree).
ImageNet 2011 Fall Release (32326)
plant, flora, plant life (4486)
geological formation, formation (175)
natural object (1112)
sport, athletics (176)
artifact, artefact (10504)
instrumentality, instrumentation (5494)
structure, construction (1405)
airdock, hangar, repair shed (0)
altar (1)
arcade, colonnade (1)
arch (31)
area (344)
aisle (0)
auditorium (1)
baggage claim (0)
box (1)
breakfast area, breakfast nook (0)
bullpen (0)
chancel, sanctuary, bema (0)
choir (0)
corner, nook (2)
court, courtyard (6)
atrium (0)
bailey (0)
cloister (0)
food court (0)
forecourt (0)
Treemap Visualization
Images of the Synset
Downloads

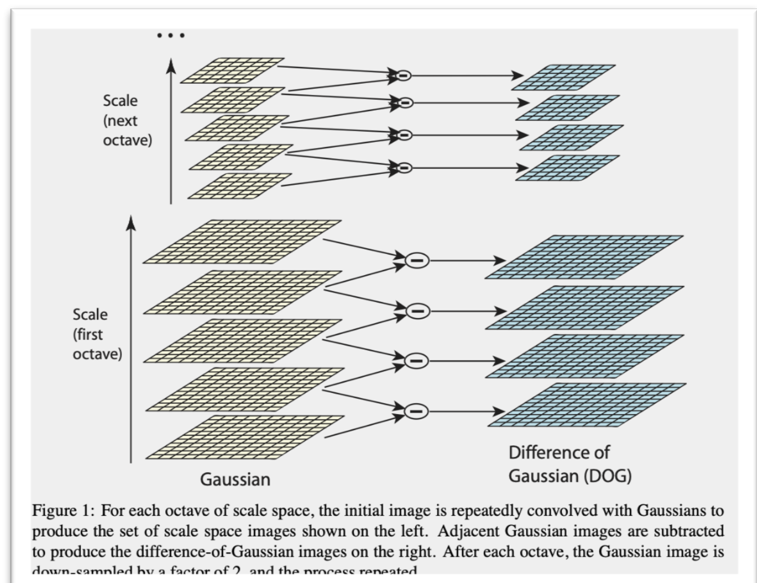
Feature Engineering for CV

Edge detection (Canny)

Corner Detection (Harris)



Scale Invariant Feature Transform (SIFT)



Figures from http://opencv.org

Figure from Lowe (1999) and Lowe (2004)

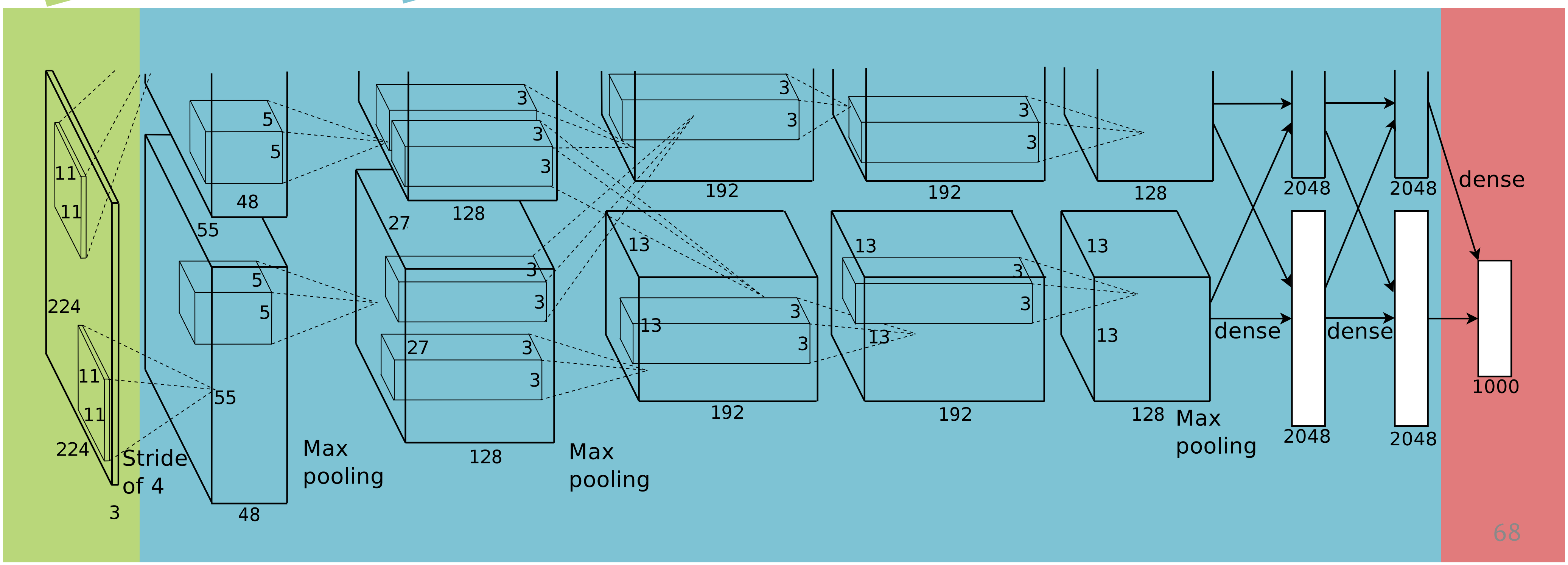
Example: Image Classification

CNN for Image Classification
(Krizhevsky, Sutskever & Hinton, 2012)
15.3% error on ImageNet LSVRC-2012 contest

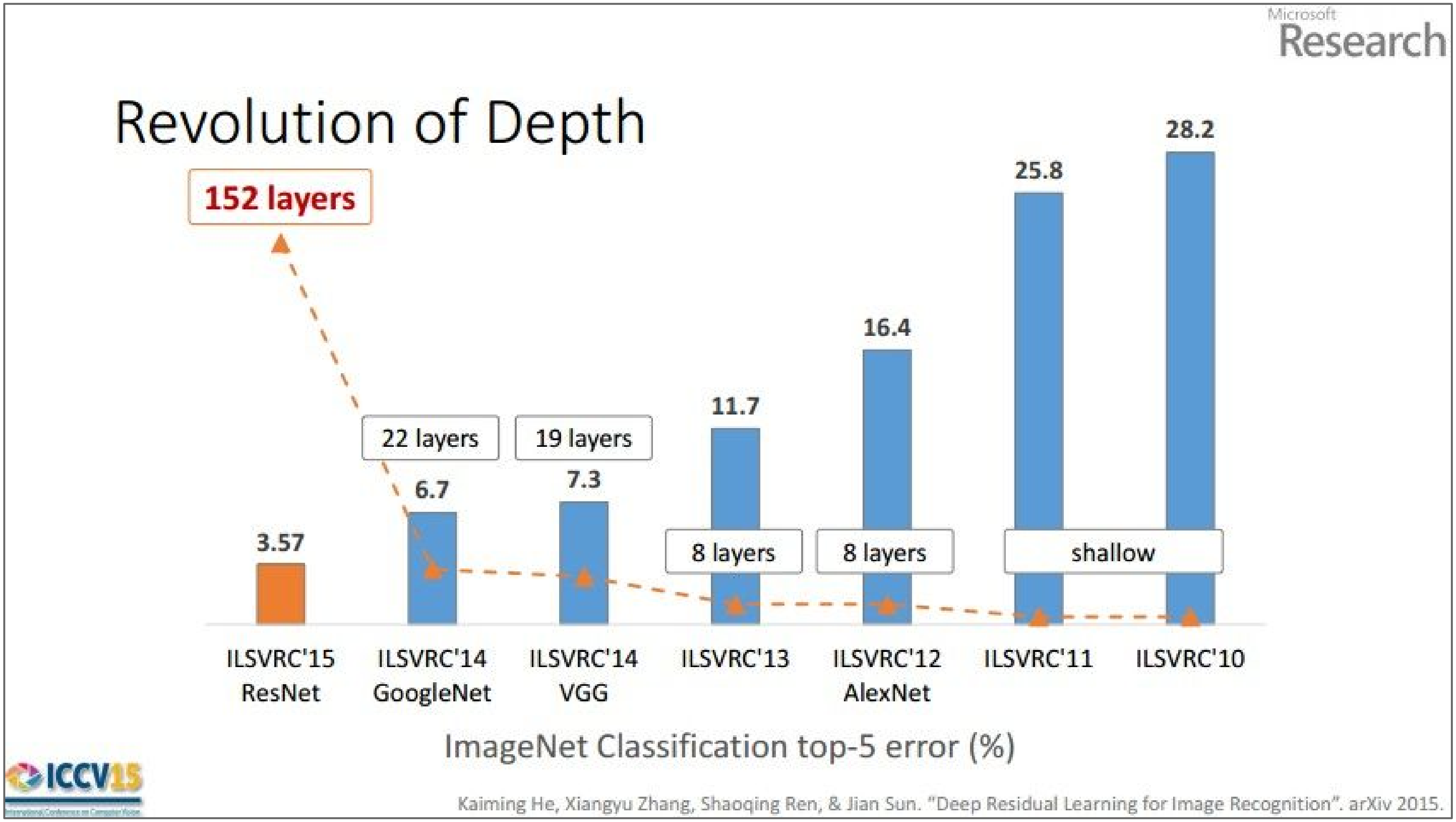
Input image (pixels)

- Five convolutional layers (w/max-pooling)
- Three fully connected layers

1000-way softmax



CNNs for Image Recognition



Slide from Kaiming He

Backpropagation and Deep Learning

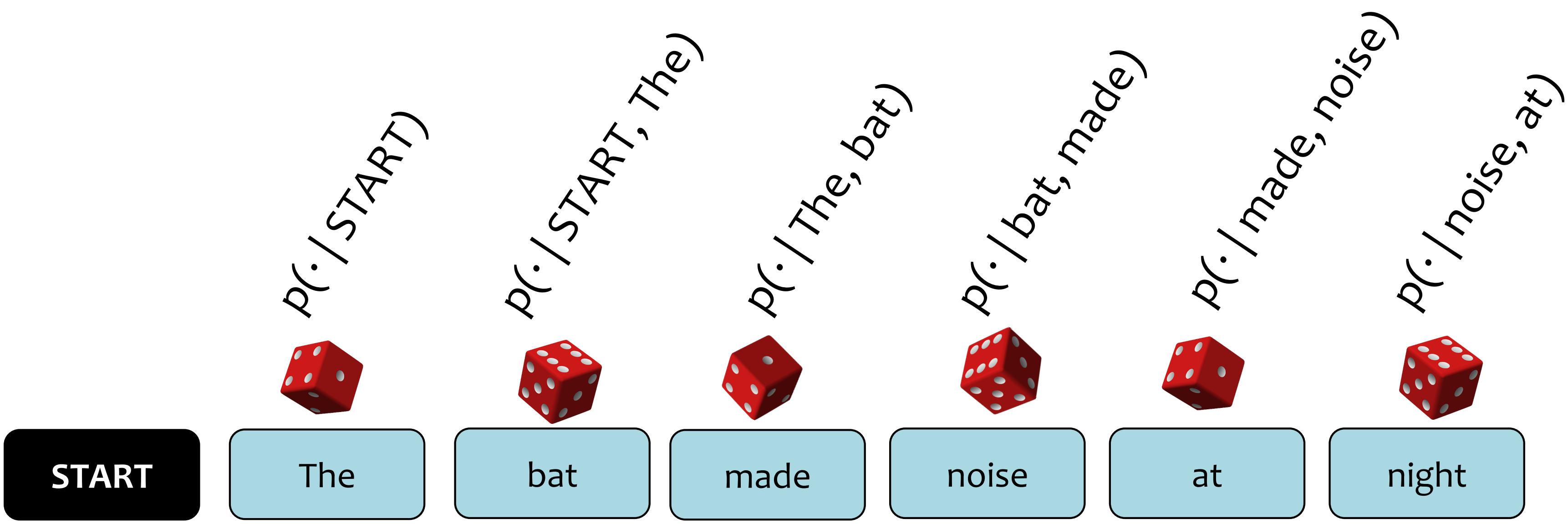
Convolutional neural networks (CNNs) and **recurrent neural networks** (RNNs) are simply fancy computation graphs (aka. hypotheses or decision functions).

Our recipe also applies to these models and (again) relies on the **backpropagation algorithm** to compute the necessary gradients.

BACKGROUND:
N-GRAM LANGUAGE MODELS

n-Gram Language Model

- *Goal*: Generate realistic looking sentences in a human language
- *Key Idea*: condition on the last n-1 words to sample the n^{th} word



n-Gram Language Model

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=2) $$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1})$$

$$p(w_1, w_2, w_3, \ldots, w_6) =$$

The	$p(w_1)$
The bat	$p(w_2 \mid w_1)$
bat made	$p(w_3 \mid w_2)$
made noise	$p(w_4 \mid w_3)$
noise at	$p(w_5 \mid w_4)$
at night	$p(w_6 \mid w_5)$

n-Gram Language Model

<u>Question</u>: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

n-Gram Model (n=3)

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$$

$p(w_1, w_2, w_3, \ldots, w_6) =$

The						$p(w_1)$
The	bat					$p(w_2 \mid w_1)$
The	bat	made				$p(w_3 \mid w_2, w_1)$
	bat	made	noise			$p(w_4 \mid w_3, w_2)$
		made	noise	at		$p(w_5 \mid w_4, w_3)$
			noise	at	night	$p(w_6 \mid w_5, w_4)$

n-Gram Language Model

Question: How can we **define** a probability distribution over a sequence of length T?

The bat made noise at night

w_1 w_2 w_3 w_4 w_5 w_6

n-Gram Model (n=3)

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, w_{t-2})$$

$p(w_1, w_2, w_3, \ldots, w_6) =$

The

The bat

The

$p(w_1)$

$p(w_2 \mid w_1)$

Note: This is called a **model** because we made some **assumptions** about how many previous words to condition on (i.e. only n-1 words)

Learning an n-Gram Model

Question: How do we **learn** the probabilities for the n-Gram Model?

$p(w_t \mid w_{t-2} = \text{The}, w_{t-1} = \text{bat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
ate	0.015
…	
flies	0.046
…	
zebra	0.000

$p(w_t \mid w_{t-2} = \text{made}, w_{t-1} = \text{noise})$

w_t	$p(\cdot \mid \cdot, \cdot)$
at	0.020
…	
pollution	0.030
…	
zebra	0.000

$p(w_t \mid w_{t-2} = \text{cows}, w_{t-1} = \text{eat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
corn	0.420
…	
grass	0.510
…	
zebra	0.000

Learning an n-Gram Model

Question: How do we **learn** the probabilities for the n-Gram Model?
Answer: From data! Just **count** n-gram frequencies

…the **cows eat grass**…
…our **cows eat hay** daily…
…factory-farm **cows eat corn**…
…on an organic farm, **cows eat hay** and…
…do your **cows eat grass** or corn?...
…what do **cows eat if** they have…
…**cows eat corn** when there is no…
…which **cows eat which** foods depends…
…if **cows eat grass**…
…when **cows eat corn** their stomachs…
…should we let **cows eat corn**?...

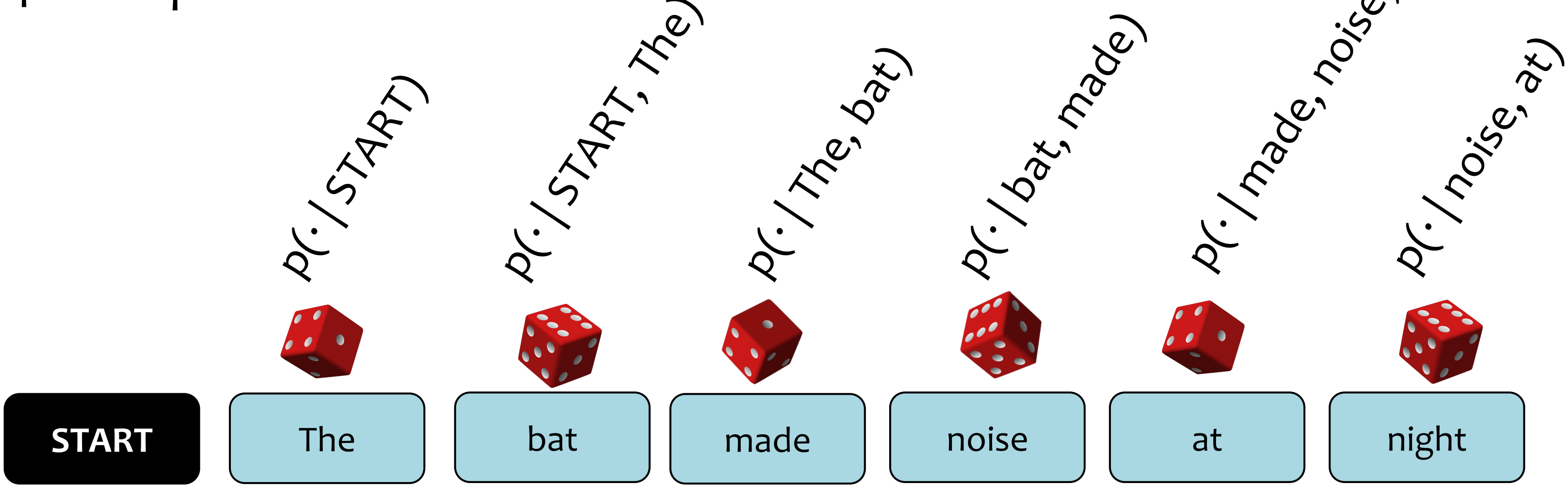
$p(w_t \mid w_{t-2} = \text{cows}, w_{t-1} = \text{eat})$

w_t	$p(\cdot \mid \cdot, \cdot)$
corn	4/11
grass	3/11
hay	2/11
if	1/11
which	1/11

Sampling from a Language Model

Question: How do we sample from a Language Model?
Answer:

1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat



Sampling from a Language Model

Question: How do we sample from a Language Model?
Answer:
1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat

Training Data (Shakespeaere)	**5-Gram Model**
I tell you, friends, most charitable care ave the patricians of you. For your wants, Your suffering in this dearth, you may as well Strike at the heaven with your staves as lift them Against the Roman state, whose course will on The way it takes, cracking ten thousand curbs Of more strong link asunder than can ever Appear in your impediment. For the dearth, The gods, not the patricians, make it, and Your knees to them, not arms, must help.	Approacheth, denay. dungy Thither! Julius think: grant,--O Yead linens, sheep's Ancient, Agreed: Petrarch plaguy Resolved pear! observingly honourest adulteries wherever scabbard guess; affirmation--his monsieur; died. jealousy, chequins me. Daphne building. weakness: sun-rise, cannot stays carry't, unpurposed. prophet-like drink; back-return 'gainst surmise Bridget ships? wane; interim? She's striving wet;

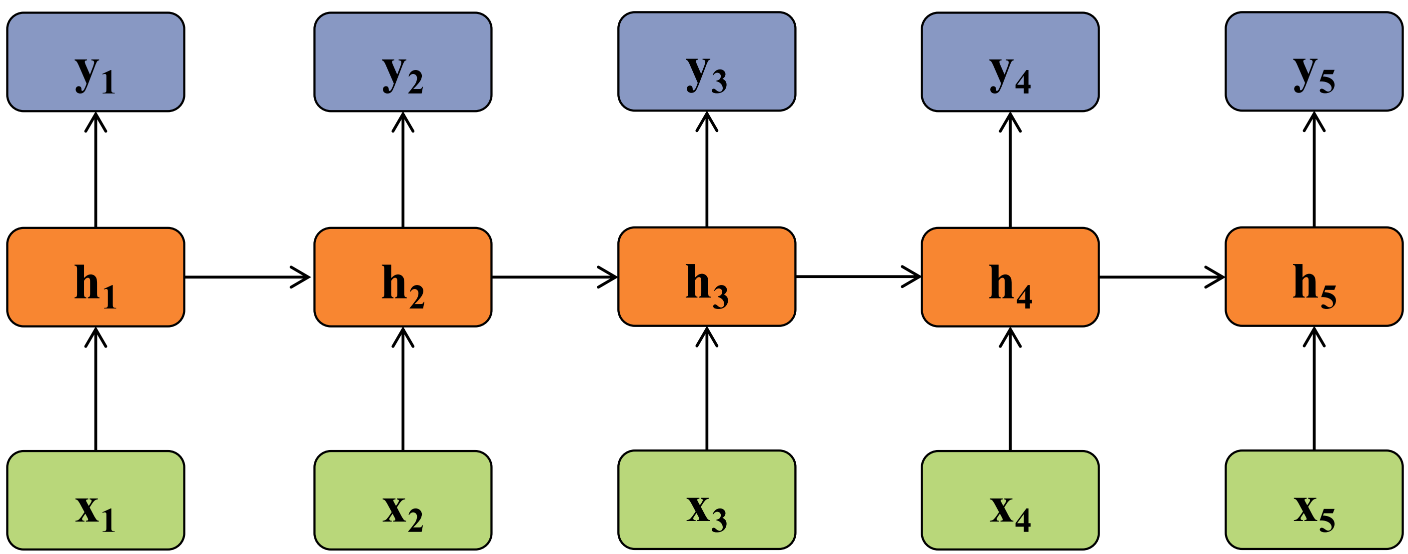
RECURRENT NEURAL NETWORK (RNN) LANGUAGE MODELS

Recurrent Neural Networks (RNNs)

$$\text{inputs: } \mathbf{x} = (x_1, x_2, \ldots, x_T), x_i \in \mathcal{R}^I$$
$$\text{hidden units: } \mathbf{h} = (h_1, h_2, \ldots, h_T), h_i \in \mathcal{R}^J$$
$$\text{outputs: } \mathbf{y} = (y_1, y_2, \ldots, y_T), y_i \in \mathcal{R}^K$$
$$\text{nonlinearity: } \mathcal{H}$$

Definition of the RNN:

$$h_t = \mathcal{H}\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right)$$
$$y_t = W_{hy}h_t + b_y$$



Recall...

The Chain Rule of Probability

Question: How can we **define** a probability distribution over a sequence of length T?

The	bat	made	noise	at	night
w_1	w_2	w_3	w_4	w_5	w_6

Chain rule of probability: $$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_{t-1}, \ldots, w_1)$$

$p(w_1, w_2, w_3, \ldots, w_6) =$

The — $p(w_1)$

The — $p(w_2 \mid w_1)$

The

The

The — $\ldots w_1)$

The — $p(w_6 \mid w_5, w_4, w_3, w_2, w_1)$

Note: This is called the chain **rule** because it is **always** true for every probability distribution

RNN Language Model

RNN Language Model: $p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid f_{\boldsymbol{\theta}}(w_{t-1}, \ldots, w_1))$

$p(w_1, w_2, w_3, \ldots, w_6) =$

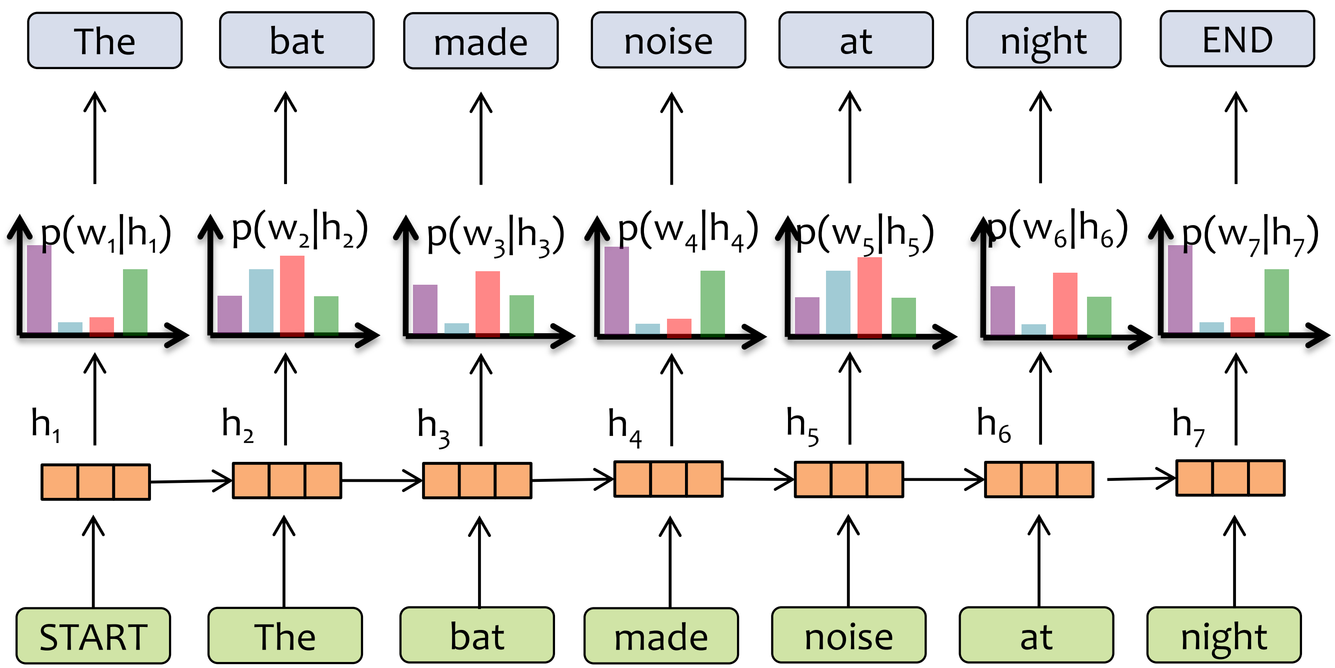
The						$p(w_1)$
The	bat					$p(w_2 \mid f_\theta(w_1))$
The	bat	made				$p(w_3 \mid f_\theta(w_2, w_1))$
The	bat	made	noise			$p(w_4 \mid f_\theta(w_3, w_2, w_1))$
The	bat	made	noise	at		$p(w_5 \mid f_\theta(w_4, w_3, w_2, w_1))$
The	bat	made	noise	at	night	$p(w_6 \mid f_\theta(w_5, w_4, w_3, w_2, w_1))$

Key Idea:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector

RNN Language Model

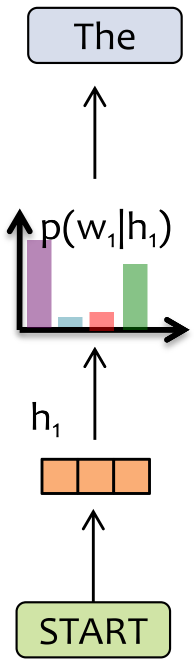


Key Idea:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

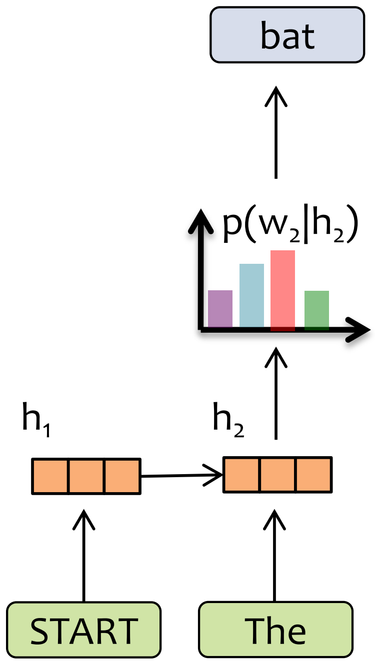


__Key Idea__:

(1) convert all previous words to a **fixed length vector**

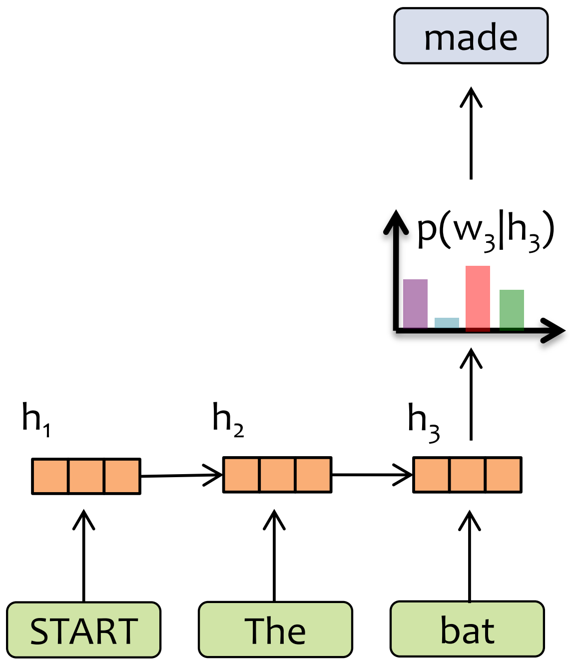
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

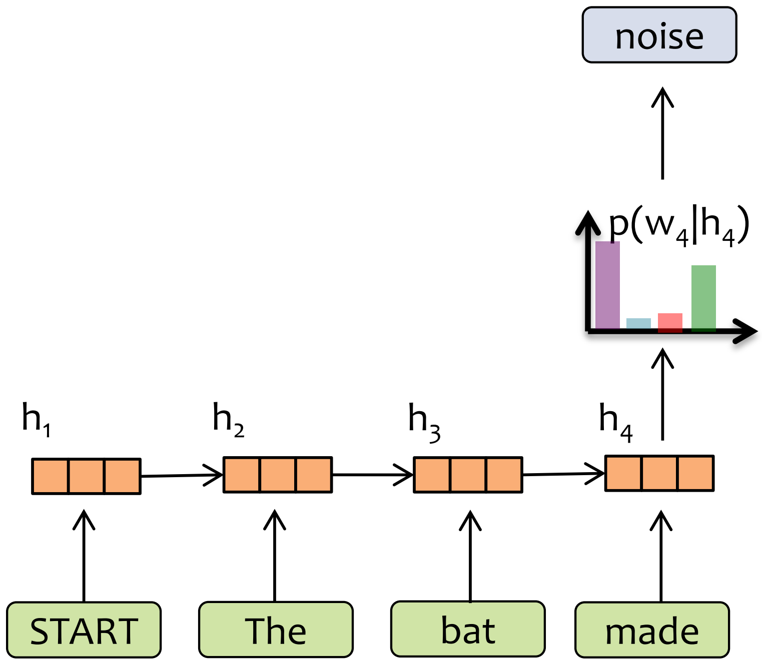


<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

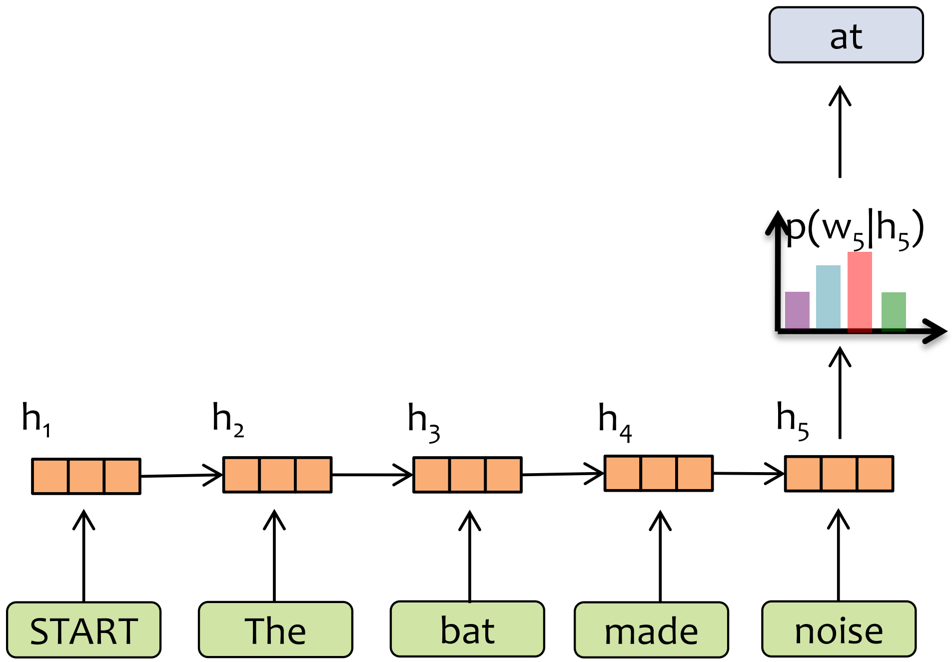
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



<u>*Key Idea*</u>:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

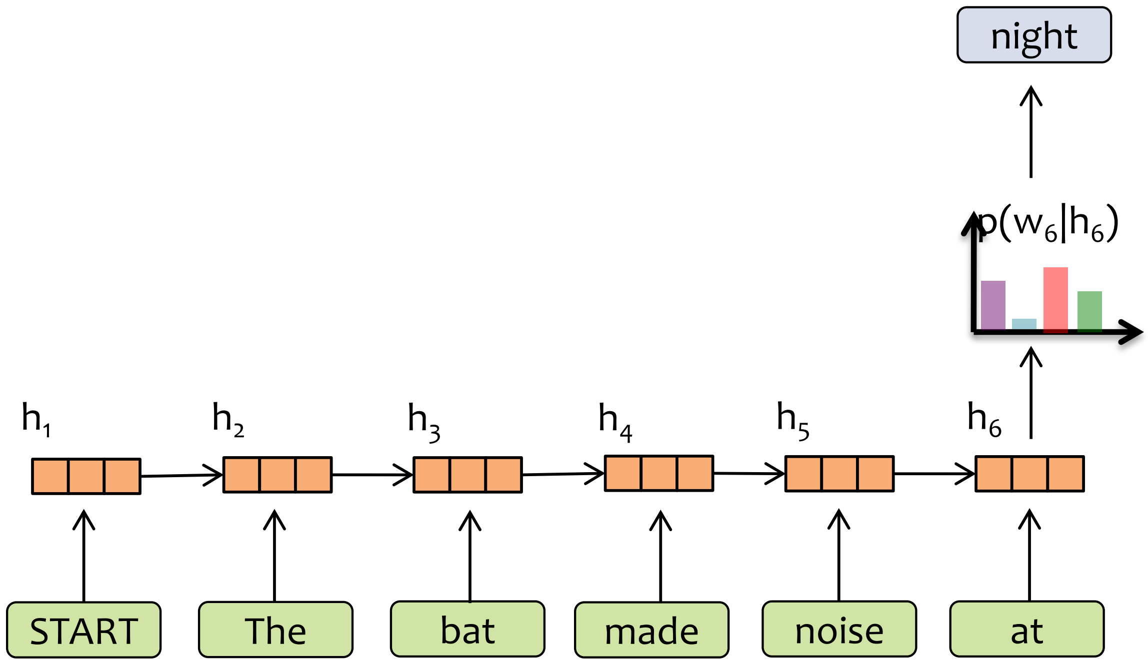


Key Idea:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

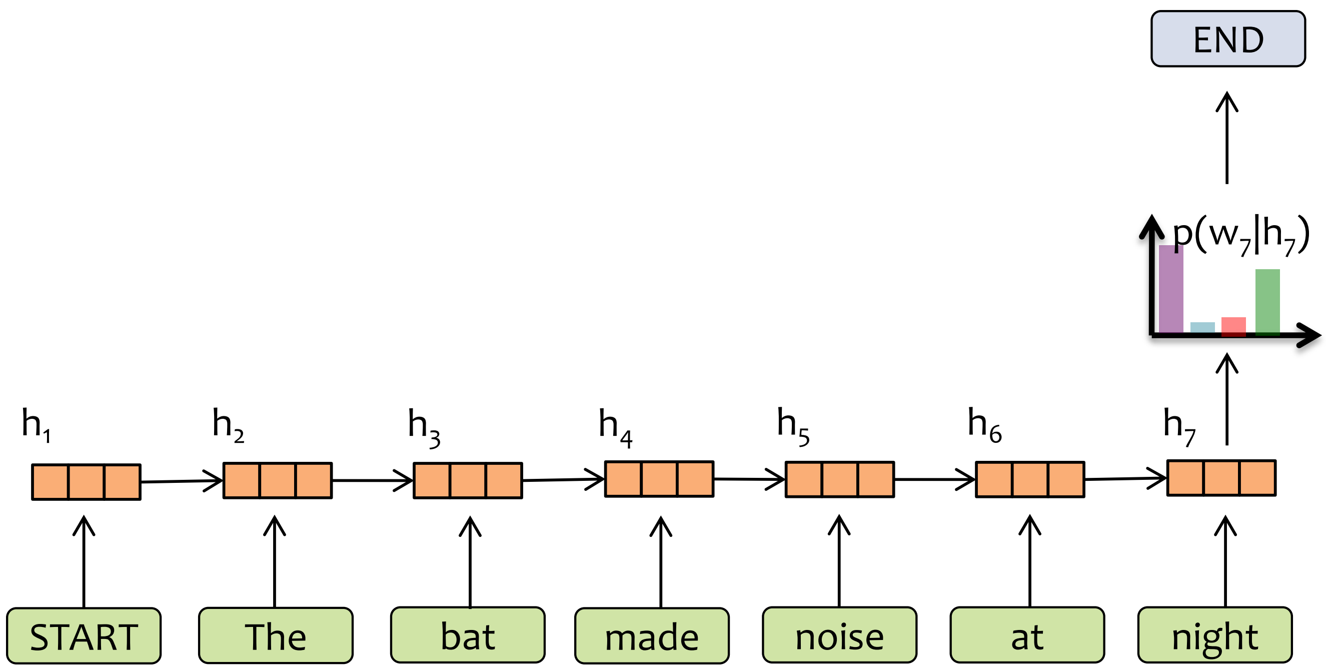


Key Idea:

(1) convert all previous words to a **fixed length vector**

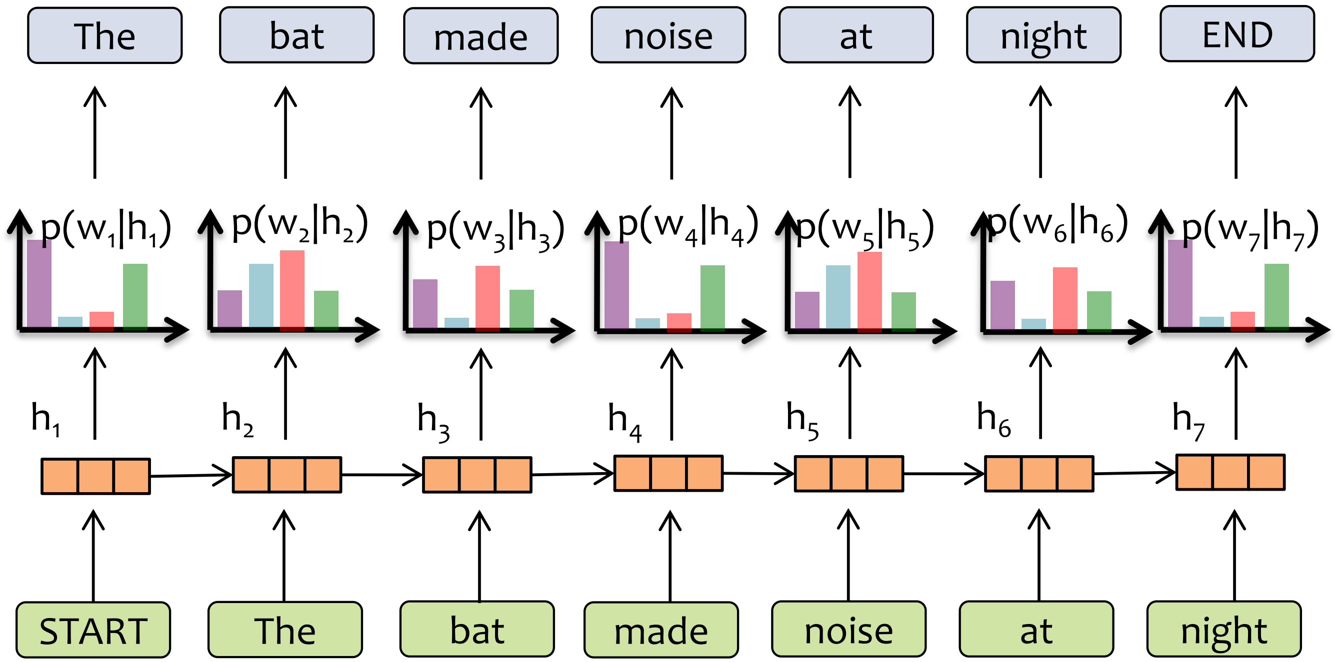
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

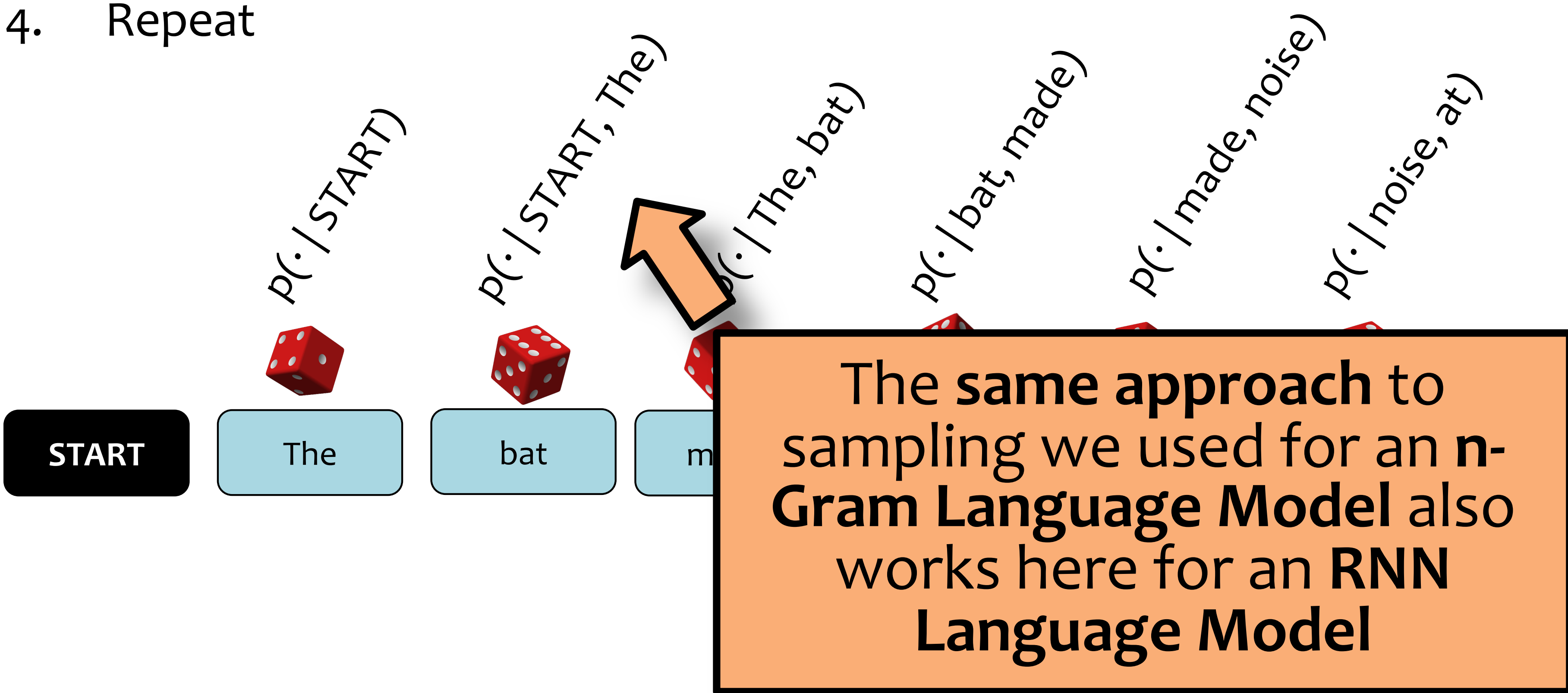


$$p(w_1, w_2, w_3, \ldots , w_T) = p(w_1 \mid h_1)\, p(w_2 \mid h_2) \ldots\ p(w_2 \mid h_T)$$

Sampling from a Language Model

Question: How do we sample from a Language Model?
Answer:

1. Treat each probability distribution like a (50k-sided) weighted die
2. Pick the die corresponding to $p(w_t \mid w_{t-2}, w_{t-1})$
3. Roll that die and generate whichever word w_t lands face up
4. Repeat



Sampling from an RNN-LM

??

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered a… master's ready there My powe… so much as hell: Some service i… bondman here, Would show hi…

KING LEAR: O, if you we… feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

??

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him … is but young and tender; and, …uld be loath to foil him, as I … honour, if he come in: …y love to you, I came hither to acquaint you wi… that either you might stay him from his int…ent or brook such disgrace well as he sha…n into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Which is the real Shakespeare?!

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sampling from an RNN-LM

Shakespeare's As You Like It

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

RNN-LM Sample

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sampling from an RNN-LM

RNN-LM Sample

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell: Some service in the noble bondman here, Would show him to her wine.

KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

Shakespeare's As You Like It

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him well. Your brother is but young and tender; and, for your love, I would be loath to foil him, as I must, for my own honour, if he come in: therefore, out of my love to you, I came hither to acquaint you withal, that either you might stay him from his intendment or brook such disgrace well as he shall run into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Sampling from an RNN-LM

??

VIOLA: Why, Salisbury must find his flesh and thought That which I am not aps, not a man and in fire, To show the reining of the raven and the wars To grace my hand reproach within, and not a fair are hand, That Caesar and my goodly father's world; When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered a … master's ready there My powe … so much as hell: Some service i … bondman here, Would show hi …

KING LEAR: O, if you we… feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my hands are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.

??

CHARLES: Marry, do I, sir; and I came to acquaint you with a matter. I am given, sir, secretly to understand that your younger brother Orlando hath a disposition to come in disguised against me to try a fall. To-morrow, sir, I wrestle for my credit; and he that escapes me without some broken limb shall acquit him … is but young and tender; and, … uld be loath to foil him, as I … honour, if he come in: … ny love to you, I came hither to acquaint you wi… that either you might stay him from his int…ent or brook such disgrace well as he sha…n into, in that it is a thing of his own search and altogether against my will.

TOUCHSTONE: For my part, I had rather bear with you than bear you; yet I should bear no cross if I did bear you, for I think you have no money in your purse.

Which is the real Shakespeare?!

Example from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

SEQUENCE TO SEQUENCE MODELS

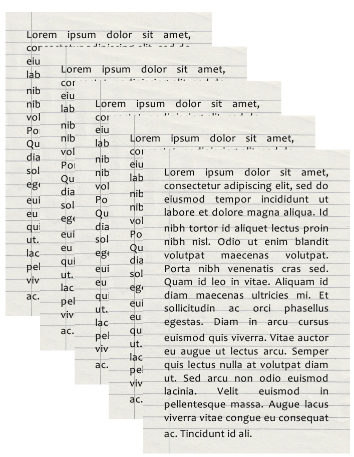
Sequence to Sequence Model

Speech Recognition

Machine Translation

기계 번역은 특히 영어와 한국어와 같은 언어 쌍의 경우 매우 어렵습니다.

Summarization



Sequence to Sequence Model

Now suppose you want generate a sequence conditioned on another input

Key Idea:

1. Use an **encoder** model to generate a vector representation of the **input**
2. Feed the output of the encoder to a **decoder** which will generate the **output**

Applications:

- translation:
  Spanish → English
- summarization:
  article → summary
- speech recognition:
  speech signal → transcription