

**TTIC 31230, Fundamentals of Deep Learning, Winter 2019**

David McAllester

**The Fundamental Equations of Deep Learning**

## Early History

**1943:** McCulloch and Pitts introduced the linear threshold “neuron”.

**1962:** Rosenblatt applies a “Hebbian” learning rule. Novikoff proved the perceptron convergence theorem.

**1969:** Minsky and Papert publish the book *Perceptrons*.

The Perceptrons book greatly discourages work in artificial neural networks. Symbolic methods dominate AI research through the 1970s.

## 80s Renaissance

**1980:** Fukushima introduces the neocognitron (a form of CNN)

**1984:** Valiant defines PAC learnability and stimulates learning theory.  
Wins Turing Award in 2010.

**1985:** Hinton and Sejnowski introduce the Boltzman machine

**1986:** Rumelhart, Hinton and Williams demonstrate empirical success with backpropagation (itself dating back to 1961).

## 90s and 00s: Research In the Shadows

**1997:** Schmidhuber et al. introduce LSTMs

**1998:** LeCun introduces convolutional neural networks (CNNs) (LeNet).

**2003:** Bengio introduces neural language modeling.

## Current Era

**2012:** Alexnet dominates the Imagenet computer vision challenge.

Google speech recognition converts to deep learning.

Both developments come out of Hinton's group in Toronto.

**2013:** Refinement of AlexNet continues to dramatically improve computer vision.

**2014:** Neural machine translation appears (Seq2Seq models).

Variational auto-encoders (VAEs) appear.

Graph networks for molecular property prediction appear.

Dramatic improvement in computer vision and speech recognition continues.

## Current Era

**2015:** Google converts to neural machine translation leading to dramatic improvements.

ResNet appears. This makes yet another dramatic improvement in computer vision.

Generative Adversarial Networks (GANs) appear.

**2016:** Alphago defeats Lee Sedol.

## Current Era

**2017:** AlphaZero learns both go and chess at super-human levels in a matter of hours entirely from self-play and advances computer go far beyond human abilities.

Unsupervised machine translation is demonstrated.

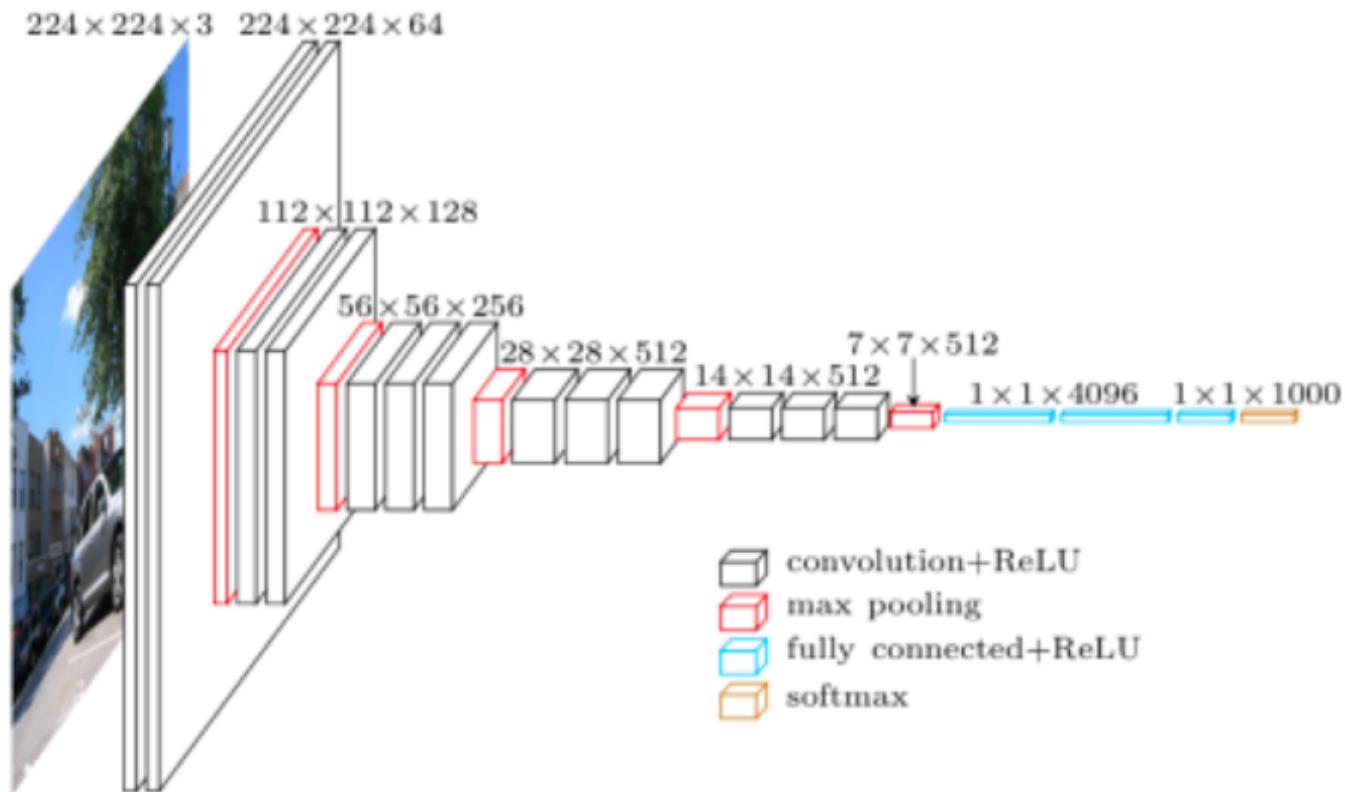
Progressive GANs.

**2018:** Unsupervised pre-training significantly improves a broad range of NLP tasks including question answering (but dialogue remains unsolved).

AlphaFold revolutionizes protein structure prediction.

# What is a Deep Network?

VGG, Zisserman, 2014



Davi Frossard



## What is a Deep Network?

We assume some set  $\mathcal{X}$  of possible inputs, some set  $\mathcal{Y}$  of possible outputs, and a parameter vector  $\Phi \in \mathbb{R}^d$ .

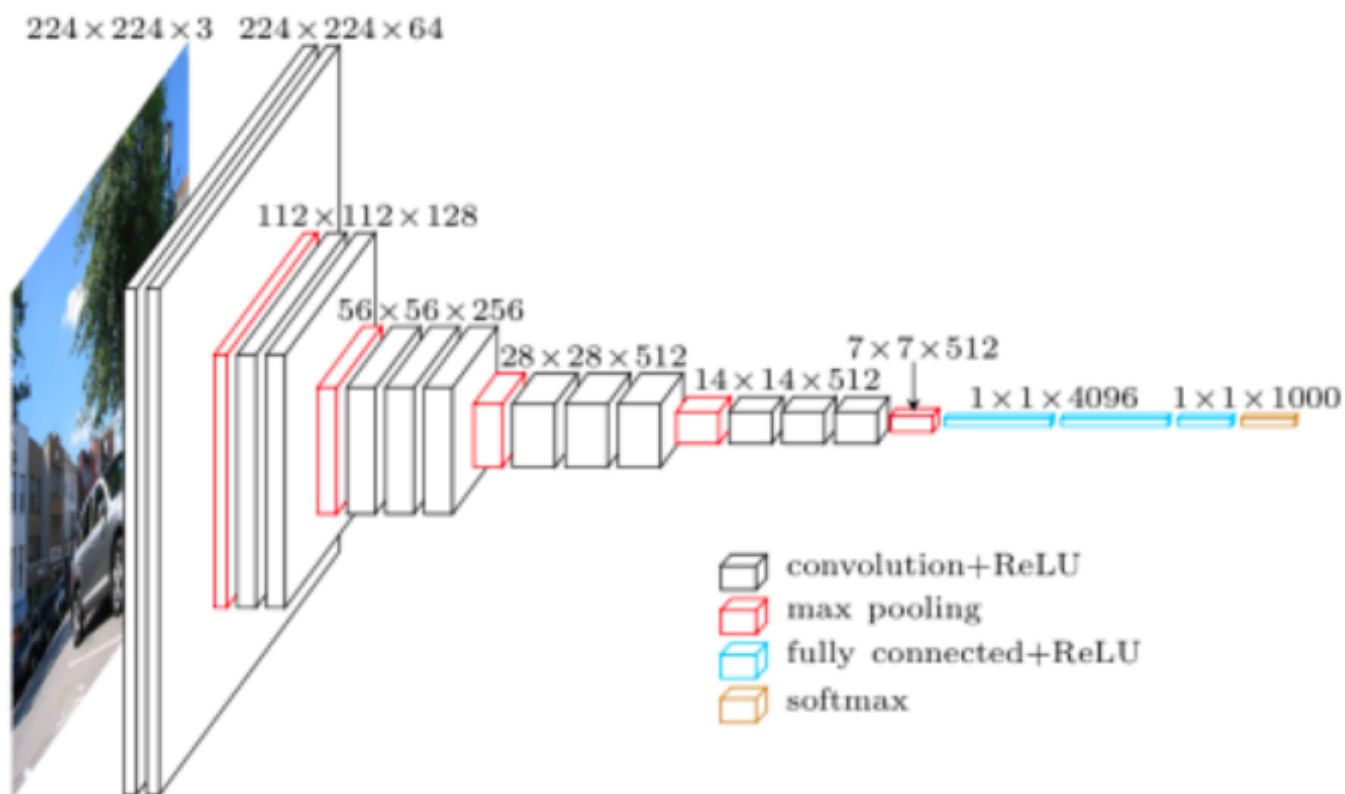
For a parameter vector  $\Phi$ , a given input  $x \in \mathcal{X}$ , and for each possible output  $y \in \mathcal{Y}$  a deep network computes a probability distribution  $P_\Phi(y|x)$  over the possible outputs  $y \in \mathcal{Y}$ .

## Softmax: Converting Scores to Probabilities

We start from a “score” function  $s_{\Phi}(y|x) \in \mathbb{R}$ .

$$\begin{aligned} P_{\Phi}(y|x) &= \frac{1}{Z} e^{s_{\Phi}(y|x)}; \quad Z = \sum_y e^{s_{\Phi}(y|x)} \\ &= \operatorname{softmax}_y s_{\Phi}(y|x) \end{aligned}$$

## Note the Final Softmax Layer



Davi Frossard

## The Fundamental Equation of Deep Learning

We assume a “population” probability distribution  $\text{Pop}$  on pairs  $(x, y)$ .

$$\begin{aligned}\Phi^* &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} \mathcal{L}(x, y, \Phi) \\ &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} -\ln P_{\Phi}(y|x)\end{aligned}$$

This loss function  $\mathcal{L}(x, y, \Phi) = -\ln P_{\Phi}(y|x)$  is called **cross entropy loss**.

## Binary Classification

We have a population distribution over  $(x, y)$  with  $y \in \{-1, 1\}$ .

We compute a single score  $s_\Phi(x)$  where

for  $s_\Phi(x) \geq 0$  predict  $y = 1$

for  $s_\Phi(x) < 0$  predict  $y = -1$

## Softmax for Binary Classification

$$\begin{aligned} P_{\Phi}(y|x) &= \frac{1}{Z} e^{ys(x)} \\ &= \frac{e^{ys(x)}}{e^{ys(x)} + e^{-ys(x)}} \\ &= \frac{1}{1 + e^{-2ys(x)}} \\ &= \frac{1}{1 + e^{-m(y)}} \quad m(y|x) = 2ys(x) \text{ is the margin} \end{aligned}$$

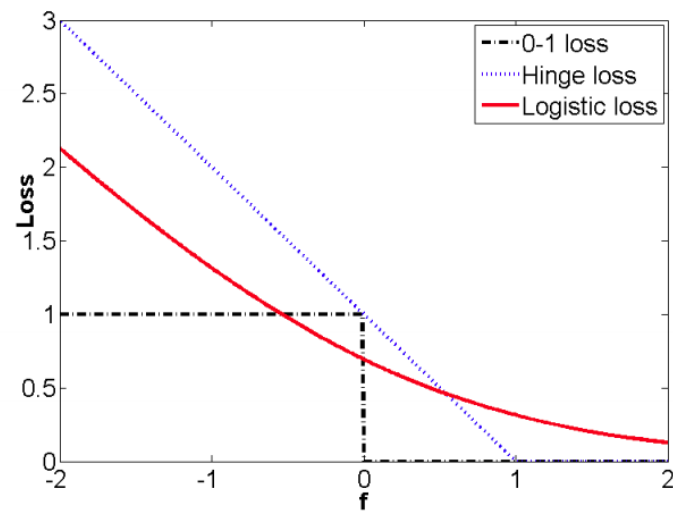
## Logistic Regression for Binary Classification

$$\begin{aligned}\Phi^* &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} \mathcal{L}(x, y, \Phi) \\ &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} -\ln P_{\Phi}(y|x) \\ &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} \ln \left( 1 + e^{-m(y|x)} \right)\end{aligned}$$

$$\ln \left( 1 + e^{-m(y|x)} \right) \approx 0 \quad \text{for } m(y|x) \gg 1$$

$$\ln \left( 1 + e^{-m(y|x)} \right) \approx -m(y|x) \quad \text{for } -m(y|x) \gg 1$$

## Log Loss vs. Hinge Loss (SVM loss)





## Image Classification (Multiclass Classification)

We have a population distribution over  $(x, y)$  with  $y \in \{y_1, \dots, y_k\}$ .

$$P_{\Phi}(y|x) = \underset{y}{\text{softmax}} \ s_{\Phi}(y|x)$$

$$\begin{aligned} \Phi^* &= \underset{\Phi}{\operatorname{argmin}} \ E_{(x,y) \sim \text{Pop}} \ \mathcal{L}(x, y, \Phi) \\ &= \underset{\Phi}{\operatorname{argmin}} \ E_{(x,y) \sim \text{Pop}} \ -\ln P_{\Phi}(y|x) \end{aligned}$$

## Machine Translation (Structured Labeling)

We have a population of translation pairs  $(x, y)$  with  $x \in V_x^*$  and  $y \in V_y^*$  where  $V_x$  and  $V_y$  are source and target vocabularies respectively.

$$P_{\Phi}(w_{t+1}|x, w_1, \dots, w_t) = \operatorname{softmax}_{w \in V_y \cup \langle \text{EOS} \rangle} s_{\Phi}(w \mid x, w_1, \dots, w_t)$$

$$P_{\Phi}(y|x) = \prod_{t=0}^{|y|} P_{\Phi}(y_{t+1} \mid x, y_1, \dots, y_t)$$

$$\begin{aligned} \Phi^* &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} \mathcal{L}(x, y, \Phi) \\ &= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} -\ln P_{\Phi}(y|x) \end{aligned}$$

## Entropy, Cross Entropy and KL Divergence

Let  $P$  and  $Q$  be two probability distributions on the same set  $\mathcal{Y}$ .

$$\text{Entropy :} \quad H(P) = E_{y \sim P} - \ln P(y)$$

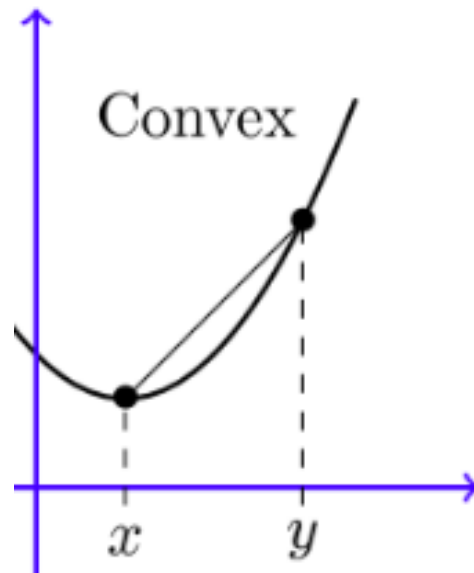
$$\text{CrossEntropy :} \quad H(P, Q) = E_{y \sim P} - \ln Q(y)$$

$$\text{KL Divergence :} \quad KL(P, Q) = E_{y \sim P} \ln \frac{P(y)}{Q(y)}$$

Cross Entropy Loss:

$$E_{(x,y) \sim \text{Pop}} - \ln P_{\Phi}(y|x) = E_{x \sim \text{Pop}} H(\text{Pop}(y|x), P_{\Phi}(y|x))$$

## Jensen's Inequality



For  $f$  convex (upward curving) we have

$$E[f(x)] \geq f(E[x])$$

## KL Divergence

$$KL(P, Q) \geq 0$$

Proof:

$$\begin{aligned} KL(P, Q) &= E_{y \sim P} - \log \frac{Q(y)}{P(y)} \\ &\geq -\log E_{x \sim P} \frac{Q(y)}{P(y)} \\ &= -\log \sum_y P(y) \frac{Q(y)}{P(y)} \\ &= -\log \sum_y Q(y) \\ &= 0 \end{aligned}$$

## Fundamental Equations

$$KL(P, Q) \geq 0$$

$$H(P, Q) = H(P) + KL(P, Q)$$

$$\operatorname{argmin}_Q H(P, Q) = P$$

$$\begin{aligned} \operatorname{argmin}_{Q(y|x)} E_{(x,y) \sim \text{Pop}} - \ln Q(y|x) &= \operatorname{argmin}_{Q(y|x)} E_{x \sim \text{Pop}} H(\text{Pop}(y|x), Q(y|x)) \\ &= \text{Pop}(y|x) \end{aligned}$$

## Asymmetry of Cross Entropy

Consider

$$\Phi^* = \operatorname{argmin}_{\Phi} H(P, Q_{\Phi}) \quad (1)$$

$$\Phi^* = \operatorname{argmin}_{\Phi} H(Q_{\Phi}, P) \quad (2)$$

For (1)  $Q_{\Phi}$  must cover all of the support of  $P$ .

For (2)  $Q_{\Phi}$  concentrates all mass on the point maximizing  $P$ .

## Asymmetry of KL Divergence

Consider

$$\begin{aligned}\Phi^* &= \operatorname{argmin}_{\Phi} KL(P, Q_{\Phi}) \\ &= \operatorname{argmin}_{\Phi} H(P, Q_{\Phi})\end{aligned}\tag{1}$$

$$\begin{aligned}\Phi^* &= \operatorname{argmin}_{\Phi} KL(Q_{\Phi}, P) \\ &= \operatorname{argmin}_{\Phi} H(Q_{\Phi}, P) - H(Q_{\Phi})\end{aligned}\tag{2}$$

If  $Q_{\Phi}$  is not universally expressive we have that (1) still forces  $Q_{\Phi}$  to cover all of  $P$  (or else the KL divergence is infinite) while (2) allows  $Q_{\Phi}$  to be restricted to a single mode of  $P$  (a common outcome).



**END**