

# **TTIC 31230, Fundamentals of Deep Learning**

David McAllester, Winter 2019

## **The Fundamental Equations of Deep Learning**

## Early History

**1943:** McCulloch and Pitts introduced the linear threshold “neuron”. Words in red are important — we will discuss them in detail in this class.

**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 convolutional neural Network or CNN.

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

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

**1986:** Rummelhart, 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 (a form of recurrent neural network or RNN).

**1998:** LeCunn draws attention to 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.

## Current Era

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

Variational auto-encoders (VAEs) appear.

Generative Adversarial Networks (GANs) appear.

Graph neural networks appear (GNNs) revolutionizing the prediction of molecular properties.

Dramatic improvement in computer vision and speech recognition continues.

## Current Era

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

Batch Normalization appears improving the performance of image classification.

Residual Connections appear. This makes yet another dramatic improvement in computer vision.

**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 demonstrate high resolution realistic face generation.

The **Transformer** appears greatly improving language modeling.

## Current Era

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

AlphaFold revolutionizes protein structure prediction.

**2019:** Vector quantized VAEs (VQ-VAE) demonstrate that VAEs can be competitive with GANs for high-resolution image generation.

Super-human performance is achieved on the GLUE natural language understanding benchmark.

# 2019: Natural Language Understanding

## GLUE: General Language Understanding Evaluation

ArXiv 1804.07461

Corpus	Train	Test	Task	Metrics	Domain
Single-Sentence Tasks					
CoLA	8.5k	<b>1k</b>	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
Similarity and Paraphrase Tasks					
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	<b>391k</b>	paraphrase	acc./F1	social QA questions
Inference Tasks					
MNLI	393k	<b>20k</b>	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	<b>146</b>	coreference/NLI	acc.	fiction books

Table 1: Task descriptions and statistics. All tasks are single sentence or sentence pair classification, except STS-B, which is a regression task. MNLI has three classes; all other classification tasks have two. Test sets shown in bold use labels that have never been made public in any form.

# BERT and GLUE

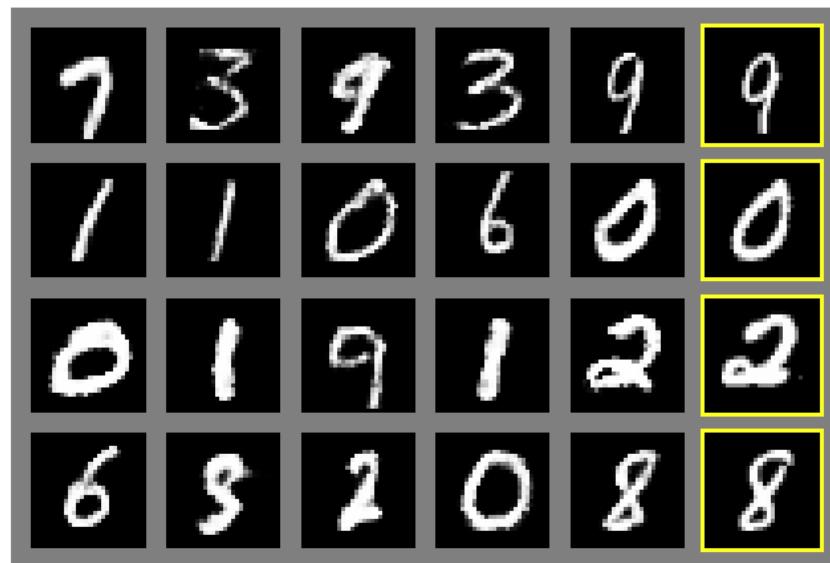
Rank	Name	Model	URL	Score
1	T5 Team - Google	T5		90.3
2	ERNIE Team - Baidu	ERNIE		90.1
3	Microsoft D365 AI & MSR AI & GATECH MT-DNN-SMART			89.9
4	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.7
5	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4
6	Junjie Yang	HIRE-RoBERTa		88.3
7	Facebook AI	RoBERTa		88.1
8	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6
9	GLUE Human Baselines	GLUE Human Baselines		87.1

## BERT and SuperGLUE

Rank	Name	Model	URL	Score
1	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8
2	T5 Team - Google	T5		89.3
3	Zhuiyi Technology	RoBERTa-mtl-adv		85.7
4	Facebook AI	RoBERTa		84.6
5	IBM Research AI	BERT-mtl		73.5

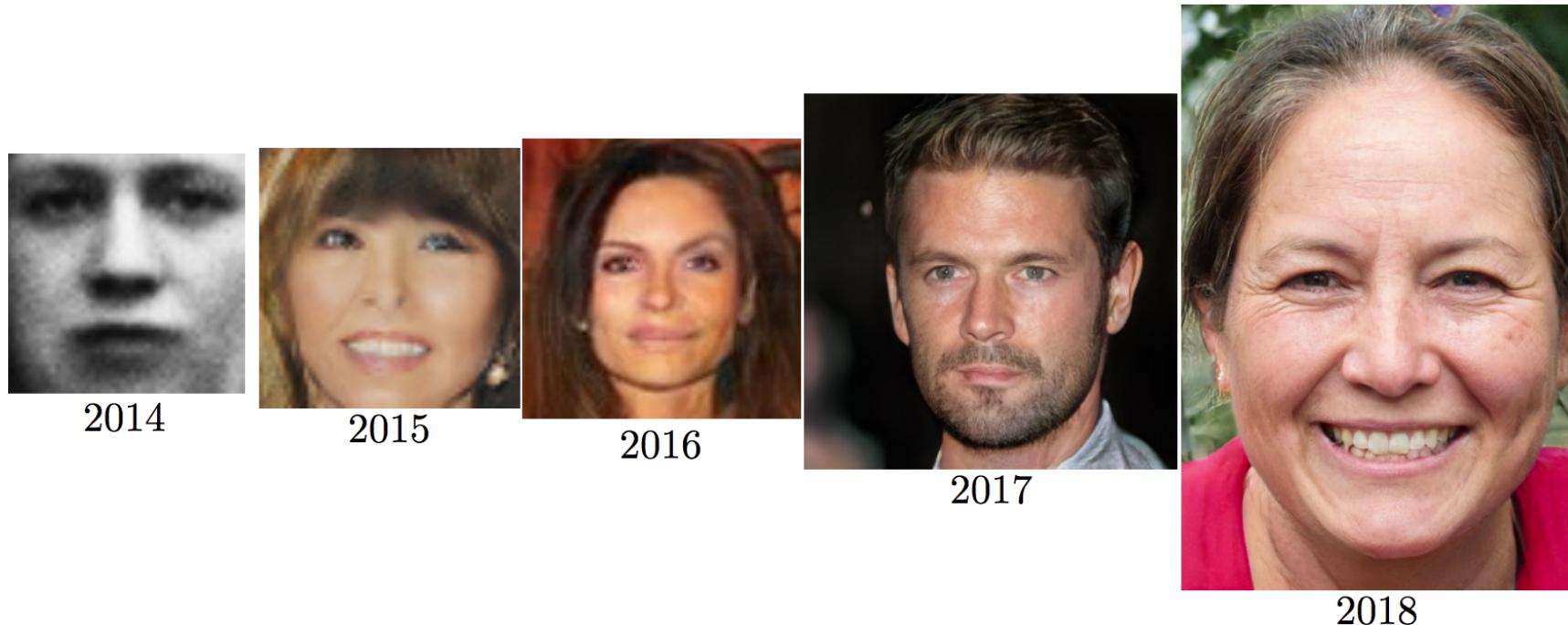
# Generative Adversarial Nets (GANs)

Goodfellow et al., 2014



# Moore's Law of AI

## 4.5 years of progress on faces



(Goodfellow 2019)

ArXiv 1406.2661, 1511.06434, 1607.07536, 1710.10196, 1812.04948  
Goodfellow, ICLR 2019 Invited Talk

# GANs for Imagenet



Odena et al  
2016



Miyato et al  
2017



Zhang et al  
2018



Brock et al  
2018

(Odena 2018)

# BigGANs, Brock et al., 2018



Figure 1: Class-conditional samples generated by our model.

## Variational Auto Encoders (VAEs, 2015)



[Alec Radford, 2015]

## VAEs in 2019



VQ-VAE-2, Razavi et al. June, 2019

## VAEs in 2019

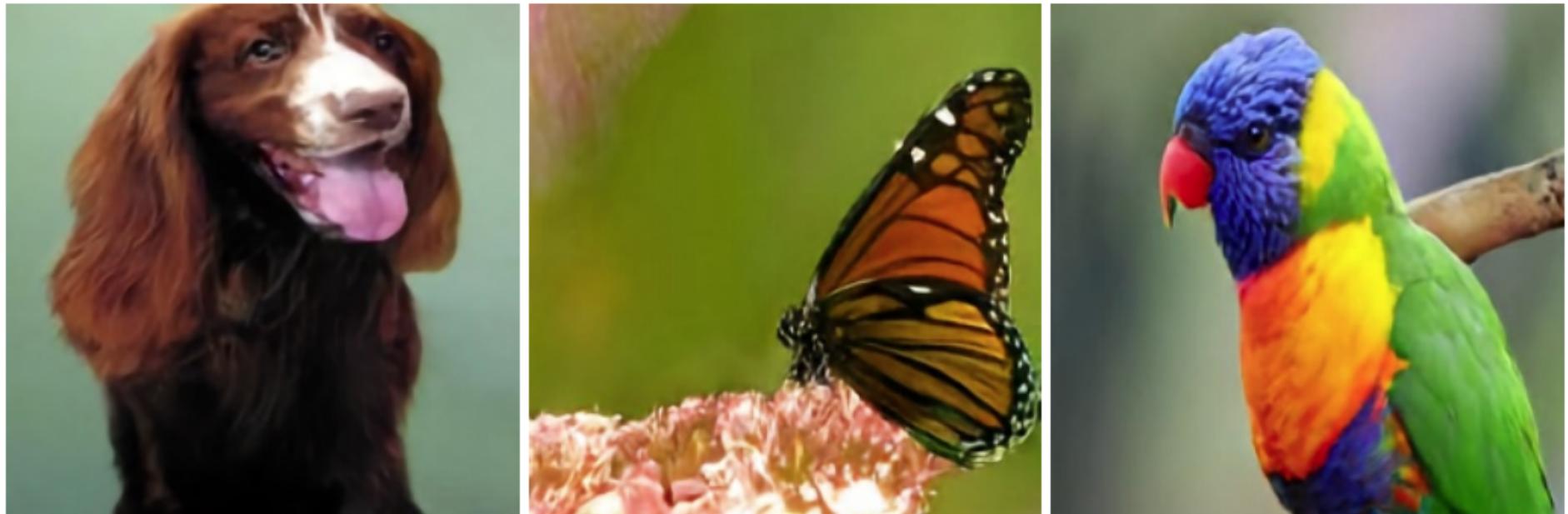
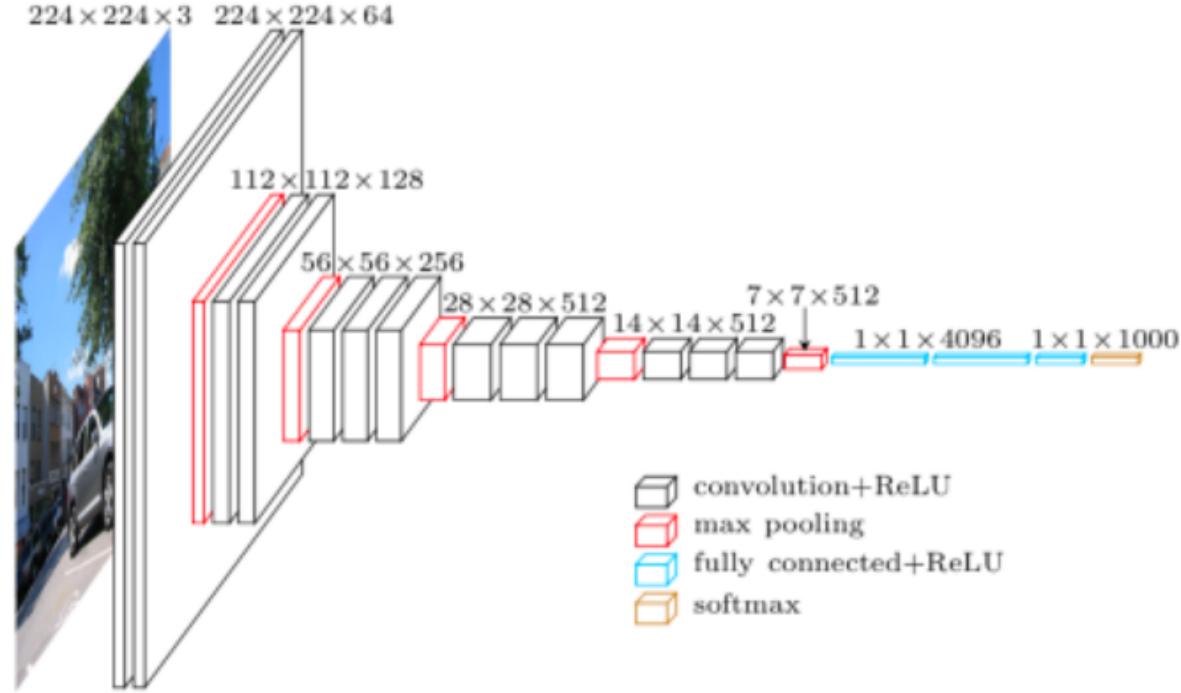


Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

VQ-VAE-2, Razavi et al. June, 2019

# What is a Deep Network?

## VGG, Zisserman, 2014



Davi Frossard

138 Million Parameters

## 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  $\Phi \in \mathbb{R}^d$  and  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$  a deep network computes a probability  $P_\Phi(y|x)$ .

# The Fundamental Equation of Deep Learning

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

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

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

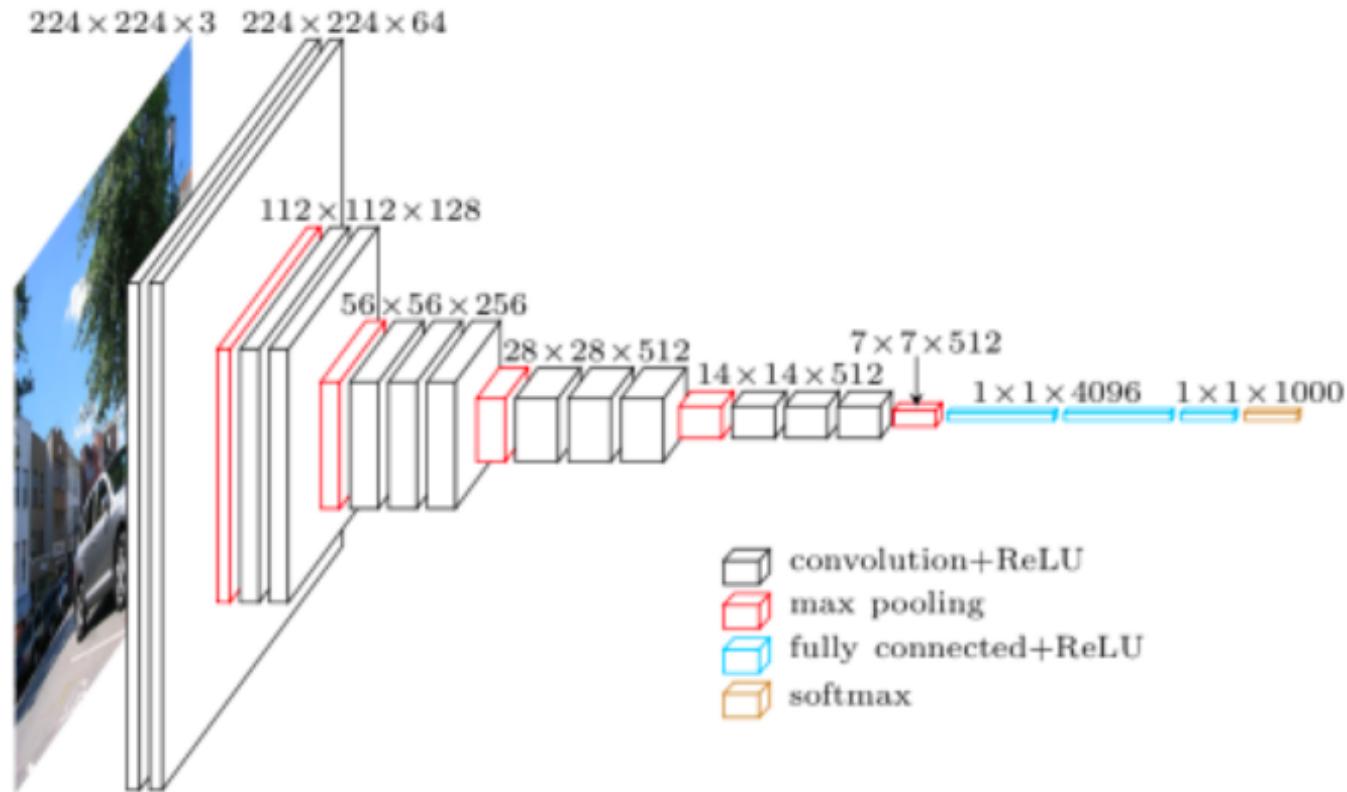
## A Second Fundamental Equation

### Softmax: Converting Scores to Probabilities

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

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

## Note the Final Softmax Layer



Davi Frossard

## How Many Possibilities

We have  $y \in \mathcal{Y}$  where  $\mathcal{Y}$  is some set of “possibilities”.

Binary:  $Y = \{-1, 1\}$

Multiclass:  $Y = \{y_1, \dots, y_k\}$   $k$  manageable.

Structured:  $y$  is a “structured object” like a sentence. Here  $|Y|$  is unmanageable.

## 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

$$\Phi^* = \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} \mathcal{L}(x, y, \Phi)$$

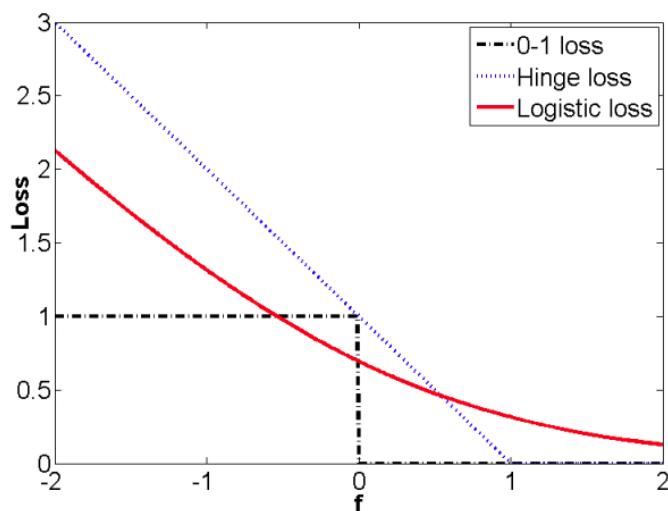
$$= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} - \ln P_{\Phi}(y|x)$$

$$= \operatorname{argmin}_{\Phi} E_{(x,y) \sim \text{Pop}} \ln \left( 1 + e^{-m(y|x)} \right)$$

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

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

## 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.

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

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

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

## Fuundamental Equation: Unconditional Form

$$\Phi^* = \operatorname{argmin}_{\Phi} E_{y \sim \text{Pop}} - \ln P_{\Phi}(y)$$

## Entropy of a Distribution

The entropy of a distribution  $P$  is defined by

$$H(P) = E_{y \sim \text{Pop}} - \ln P(y) \text{ in units of "nats"}$$

$$H_2(P) = E_{y \sim \text{Pop}} - \log_2 P(y) \text{ in units of bits}$$

Example: Let  $Q$  be a uniform distribution on 256 values.

$$E_{y \sim Q} - \log_2 Q(y) = -\log_2 \frac{1}{256} = \log_2 256 = 8 \text{ bits} = 1 \text{ byte}$$

$$1 \text{ nat} = \frac{1}{\ln 2} \text{ bits} \approx 1.44 \text{ bits}$$

## The Coding Interpretation of Entropy

We can interpret  $H_2(Q)$  as the number of bits required on average to represent items drawn from distribution  $Q$ .

We want to use fewer bits for common items.

There exists a representation where, for all  $y$ , the number of bits used to represent  $y$  is no larger than  $-\log_2 y + 1$  (Shannon's source coding theorem).

$$H(Q) = \frac{1}{\ln 2} H_2(Q) \approx 1.44 H_2(Q)$$

## Cross Entropy

Let  $P$  and  $Q$  be two distribution on the same set.

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

$$\Phi^* = \operatorname{argmin}_{\Phi} H(\text{Pop}, P_\Phi)$$

$H(P, Q)$  also has a data compression interpretation.

$H(P, Q)$  can be interpreted as 1.44 times the number of bits used to code draws from  $P$  when using the imperfect code defined by  $Q$ .

## Entropy, Cross Entropy and KL Divergence

Let  $P$  and  $Q$  be two distribution on the same set.

Entropy : 
$$H(P) = \mathbb{E}_{y \sim P} - \ln P(y)$$

CrossEntropy : 
$$H(P, Q) = \mathbb{E}_{y \sim P} - \ln Q(y)$$

KL Divergence : 
$$KL(P, Q) = H(P, Q) - H(P)$$

$$= \mathbb{E}_{y \sim P} \ln \frac{P(y)}{Q(y)}$$

We have  $H(P, Q) \geq H(P)$  or equivalently  $KL(P, Q) \geq 0$ .

## The Universality Assumption

$$\Phi^* = \operatorname{argmin}_{\Phi} H(\text{Pop}, P_{\Phi}) = \operatorname{argmin}_{\Phi} H(\text{Pop}) + KL(\text{Pop}, P_{\Phi})$$

Universality assumption:  $P_{\Phi}$  can represent any distribution and  $\Phi$  can be fully optimized.

This is clearly false for deep networks. But it gives important insights like:

$$P_{\Phi^*} = \text{Pop}$$

This is the motivation for the fundamental equation.

## 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

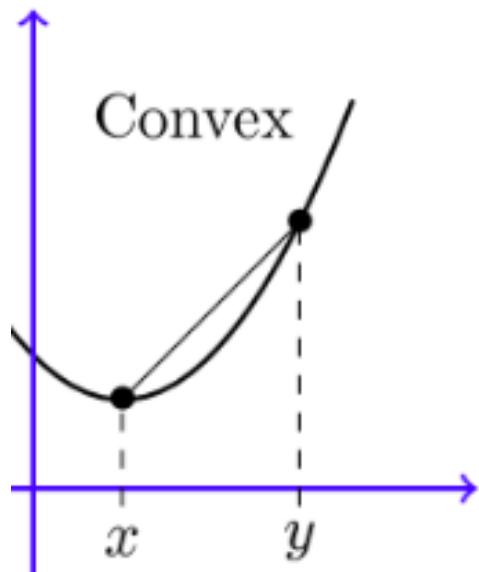
Consider

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

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

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).

## Proving $KL(P, Q) \geq 0$ : Jensen's Inequality



For  $f$  convex (upward curving) we have

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

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

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

## Summary

$\Phi^* = \operatorname{argmin}_\Phi H(\text{Pop}, P_\Phi)$  unconditional

$\Phi^* = \operatorname{argmin}_\Phi E_{x \sim \text{Pop}} H(\text{Pop}(y|x), P_\Phi(y|x))$  conditional

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

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

KL Divergence : 
$$KL(P, Q) = H(P, Q) - H(P)$$

$$= E_{y \sim P} \ln \frac{P(y)}{Q(y)}$$

$H(P, Q) \geq H(P), \quad KL(P, Q) \geq 0, \quad \operatorname{argmin}_Q H(P, Q) = P$

## Appendix: The Rearrangement Trick

$$\begin{aligned} KL(P, Q) &= E_{x \sim P} \ln \frac{P(x)}{Q(x)} \\ &= E_{x \sim P} [(-\ln Q(x)) - (-\ln P(x))] \\ &= (E_{x \sim P} - \ln Q(x)) - (E_{x \sim P} - \ln P(x)) \\ &= H(P, Q) - H(P) \end{aligned}$$

In general  $E_{x \sim P} \ln (\prod_i A_i) = E_{x \sim P} \sum_i \ln A_i$

## Appendix: The Rearrangement Trick

$$\text{ELBO} = E_{z \sim P_\Psi(z|y)} \ln \frac{P_\Phi(z, y)}{P_\Psi(z|y)}$$

$$= E_{z \sim P_\Psi(z|y)} \ln \frac{P_\Phi(z)P_\Phi(y|z)}{P_\Psi(z|y)}$$

$$= E_{z \sim P_\Psi(z|y)} \ln \frac{P_\Phi(y)P_\Phi(z|y)}{P_\Psi(z|y)}$$

Each of the last two expressions can be grouped three different ways leading to six ways of writing the ELBO.

**END**