SPRING 2023

CS 378: INTRO TO SPEECH AND AUDIO PROCESSING

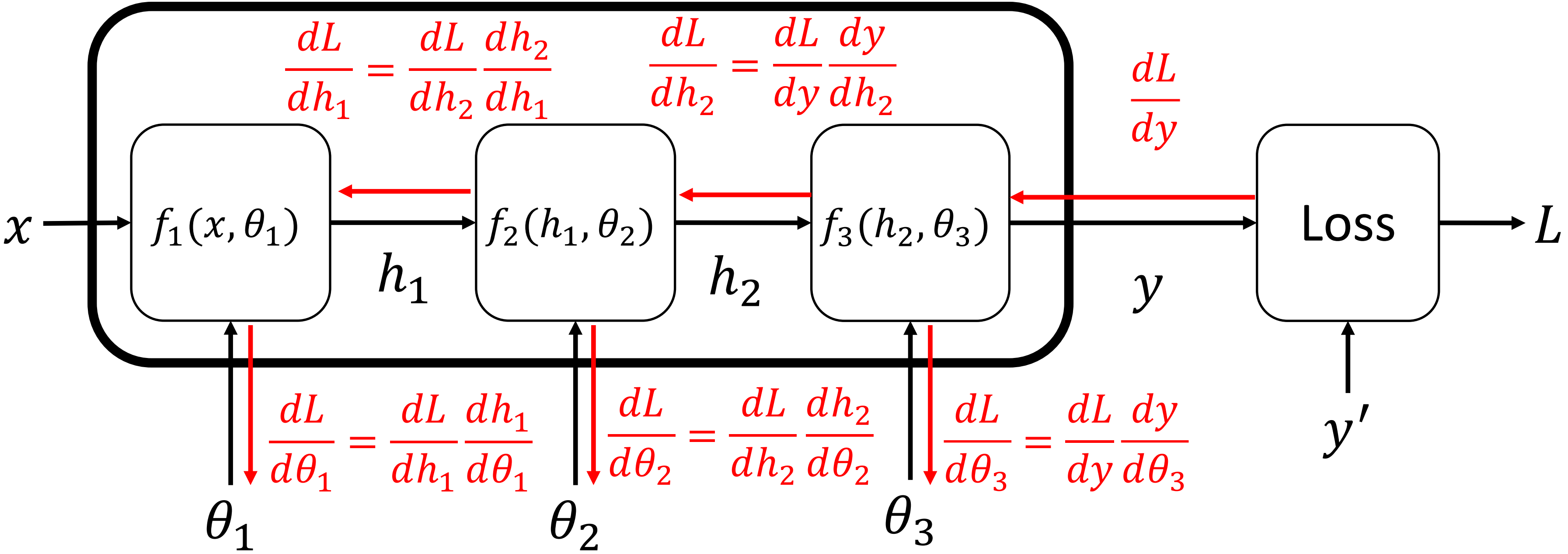
Neural Network Acoustic Models 2

DAVID HARWATH
Assistant Professor, UTCS

The University of Texas at Austin
Department of Computer Science
College of Natural Sciences

Miscellaneous Neural Net Tricks

- Xavier/Kaiming initialization: in deep networks, it's easy for gradients to explode to ∞ or collapse to 0



Miscellaneous Neural Net Tricks

- Xavier/Kaiming initialization: try to avoid exploding or vanishing gradients by careful weight initialization
- Idea: set initial weights so that the weighted sum of all inputs to a neuron will be zero mean, unit variance

$$w \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_i + n_o}}, \frac{\sqrt{6}}{\sqrt{n_i + n_o}}\right]$$

Xavier (for sigmoid/tanh)

$$w \sim \mathcal{N}\left[0, \frac{\sqrt{2}}{\sqrt{n_i}}\right]$$

Kaiming (for ReLU)

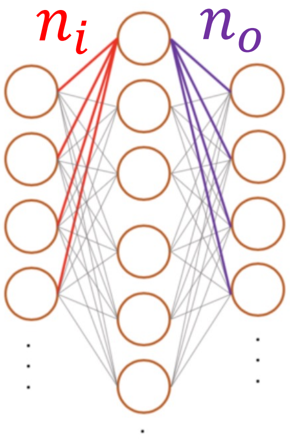


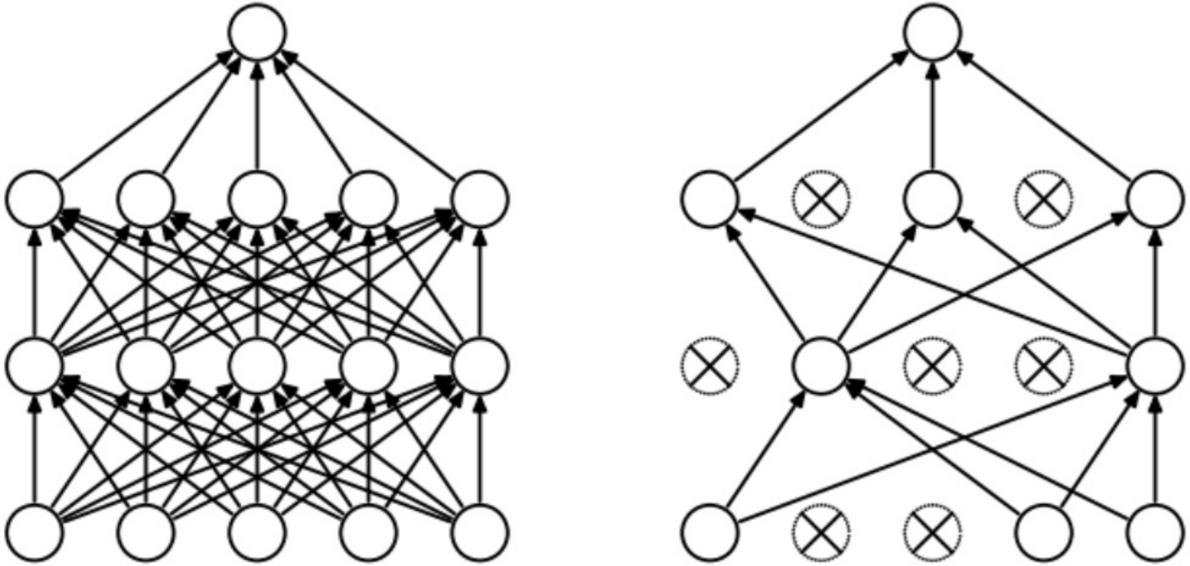
Illustration by Gideon Mendels

Glorot and Bengio, "Understanding the difficulty of training deep feedforward neural networks," AISTATS 2010

He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," ICCV 2015

Miscellaneous Neural Net Tricks

Dropout: At each minibatch, randomly choose some neurons to ignore

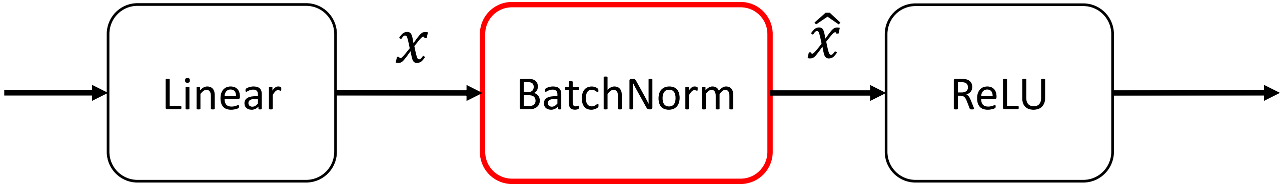


(a) Standard Neural Net (b) After applying dropout.

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Miscellaneous Neural Net Tricks

- Batch normalization: explicitly normalize the inputs to the layer to be zero mean, unit variance

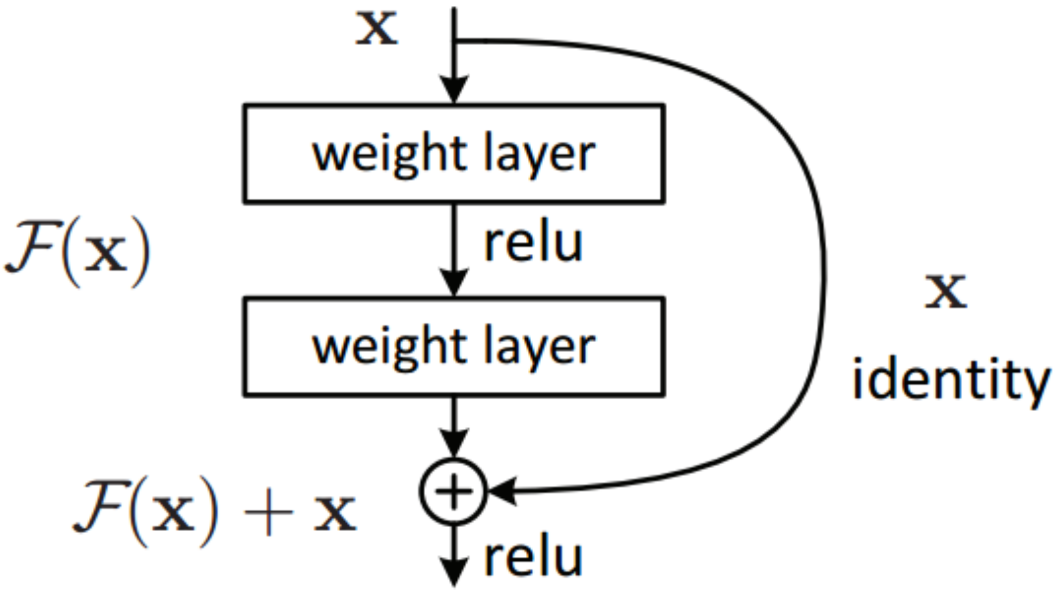


$$\mu_j = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} x_j^{(i)} \qquad \sigma_i^2 = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} (x_j^{(i)} - \mu_i)^2$$

$$\hat{x}_j = \gamma \frac{x_j - \mu_j}{\sigma + \epsilon} + \beta$$

Miscellaneous Neural Net Tricks

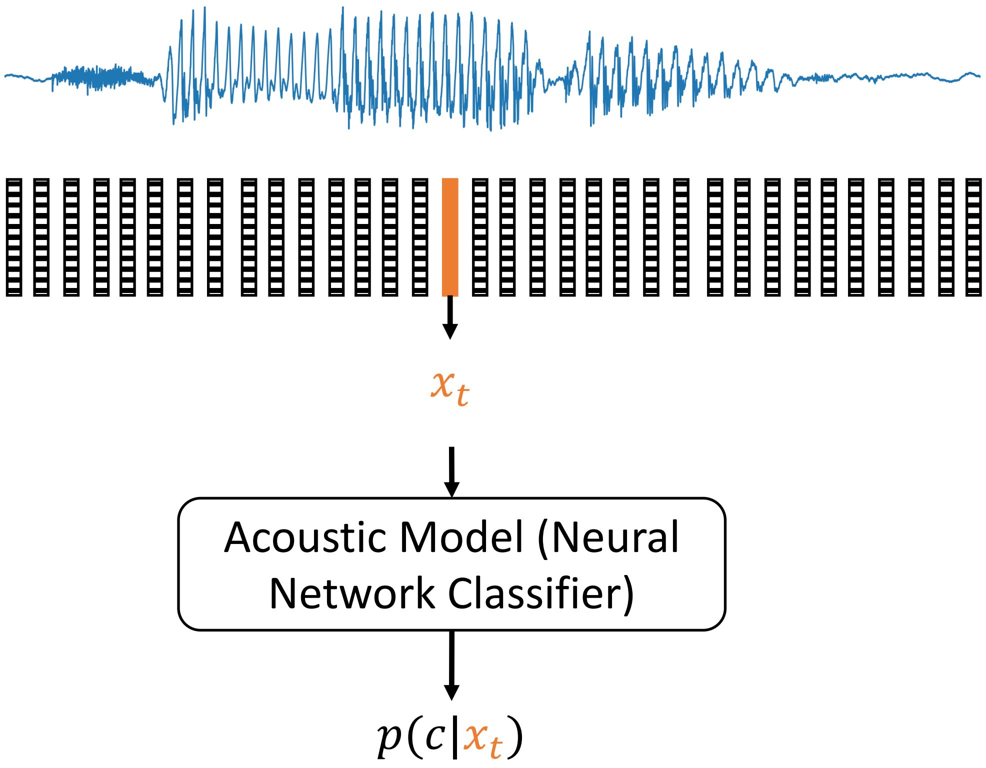
- Residual (skip) connections



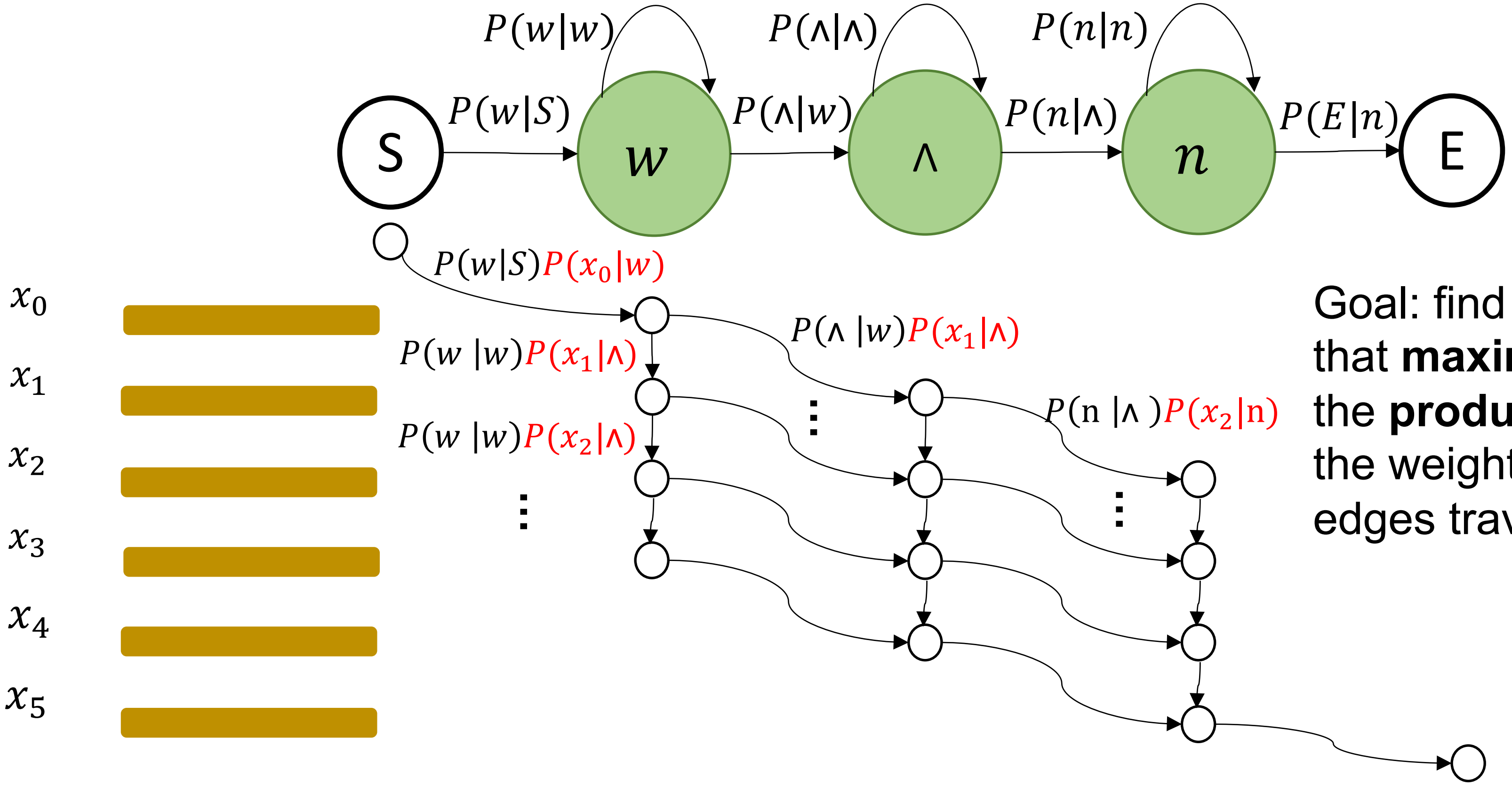
He et al., "Deep Residual Learning for Image Recognition," CVPR 2016

Recall: Acoustic Modeling for ASR

How does this actually get integrated into an ASR system?



HMM Decoding Graphs (Next Lecture..)



Goal: find path that **maximizes** the **product** of the weights of all edges traversed

ASR-specific Details

- Converting posteriors into scaled likelihoods

$$P(x|c) = \frac{P(c|x)}{P(c)}$$

- Output class targets: in the simplest case, phones. More generally, context-dependent sub-phone states

- Input features: Can be MFCCs, but nowadays more often log-Mel filterbanks with a large number of Mel filters (80)

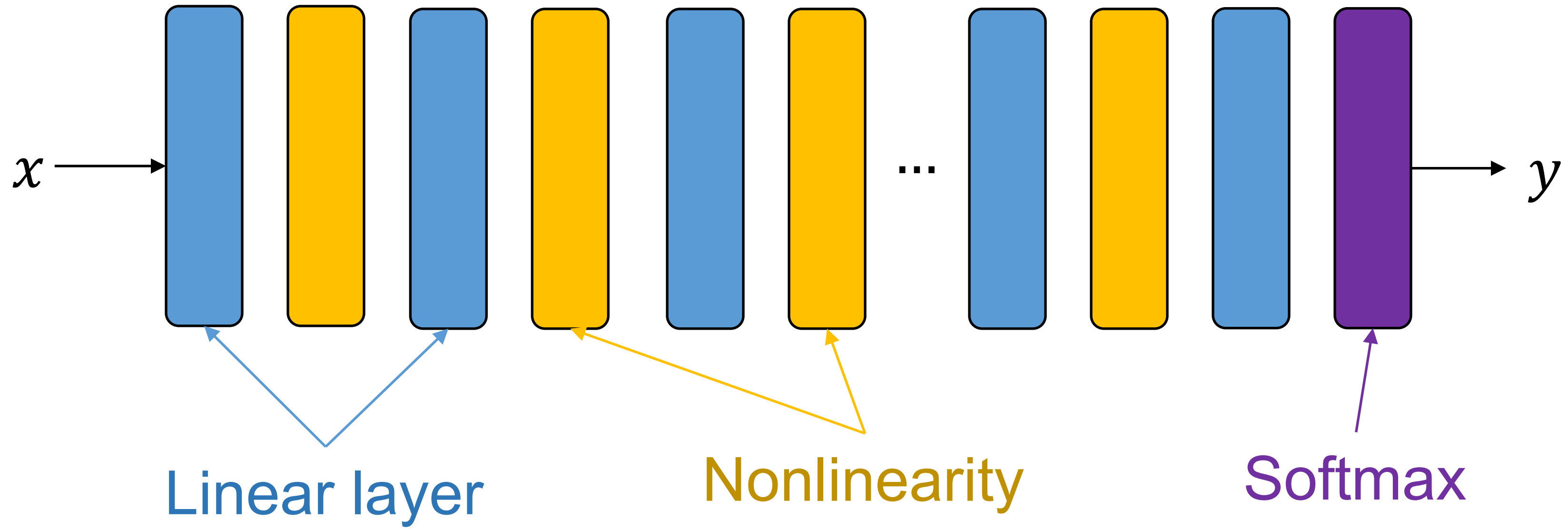
Neural Architectures for ASR

- All popular architectures have been explored extensively (feedforward, CNN, RNN, transformer), plus a few esoteric ones like TDNN
- I'll go over each of these along with some examples from the literature
- We'll discuss neural end-to-end ASR (including Transformers) in a later lecture

Feedforward

The most “vanilla” neural network architecture

Hyperparameters: # layers, # neurons/layer, activation type…



Feedforward Neural Nets

2011: first large-scale (300h) demonstration of significant gains over GMM system on conversational speech

Conversational Speech Transcription Using Context-Dependent Deep Neural Networks

Frank Seide[1], *Gang Li*,[1] *and Dong Yu*[2]

[1]Microsoft Research Asia, Beijing, P.R.C.
[2]Microsoft Research, Redmond, USA
{fseide,ganl,dongyu}@microsoft.com

Architecture:

- PLP input features (center frame +/- 5 context frames)
- 7 layers x 2048 neurons
- Sigmoid activations
- 9304 output targets (context-dependent phone states)

Table 3: *Comparing different influence factors on CD-DNN-HMM accuracy. 'nz' means 'non-zero.' Word-error rates in % for Hub5'00 SWB.*

acoustic model	#params	WER (r. chg.)
GMM 40 mix, BMMI	29.4M	23.6
CD-DNN 1 layer×4634 nodes	43.6M	26.0 (+10%)
+ 2×5 neighbor frames	45.1M	22.4 (-14%)
CD-DNN 7 layers×2048 nodes	45.1M	17.1 (-24%)
+ updated state alignment	45.1M	16.4 (-4%)
+ sparsification 66%	15.2M nz	16.1 (-2%)

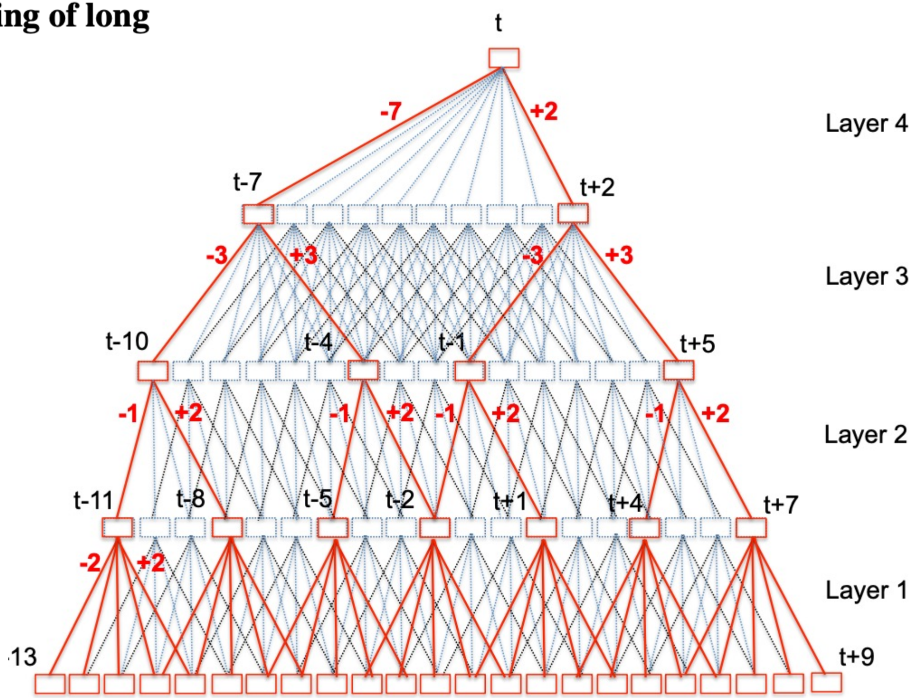
Time-Delay Neural Net (TDNN)

A time delay neural network architecture for efficient modeling of long temporal contexts

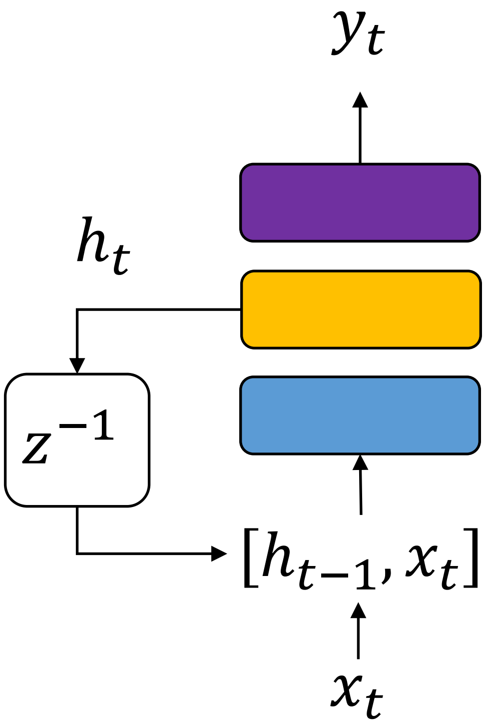
Vijayaditya Peddinti[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing &
[2]Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD 21218, USA
vijay.p,khudanpur@jhu.edu, dpovey@gmail.com

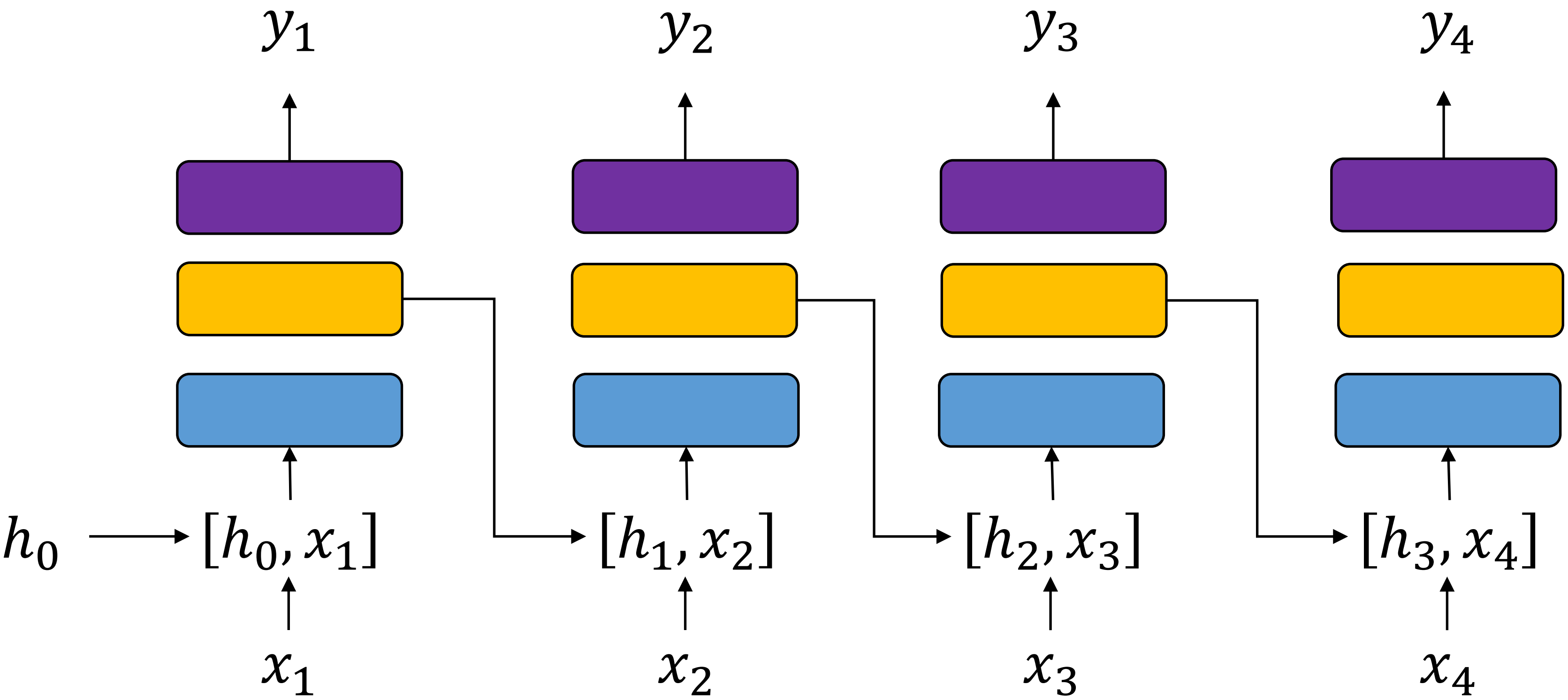
Database	Size	WER		Rel.
		DNN	TDNN	Change
Res. Management	3h hrs	2.27	2.30	-1.3
Wall Street Journal	80 hrs	6.57	6.22	5.3
Tedlium	118 hrs	19.3	17.9	7.2
Switchboard	300 hrs	15.5	14.0	9.6
Librispeech	960 hrs	5.19	4.83	6.9
Fisher English	1800 hrs	22.24	21.03	5.4



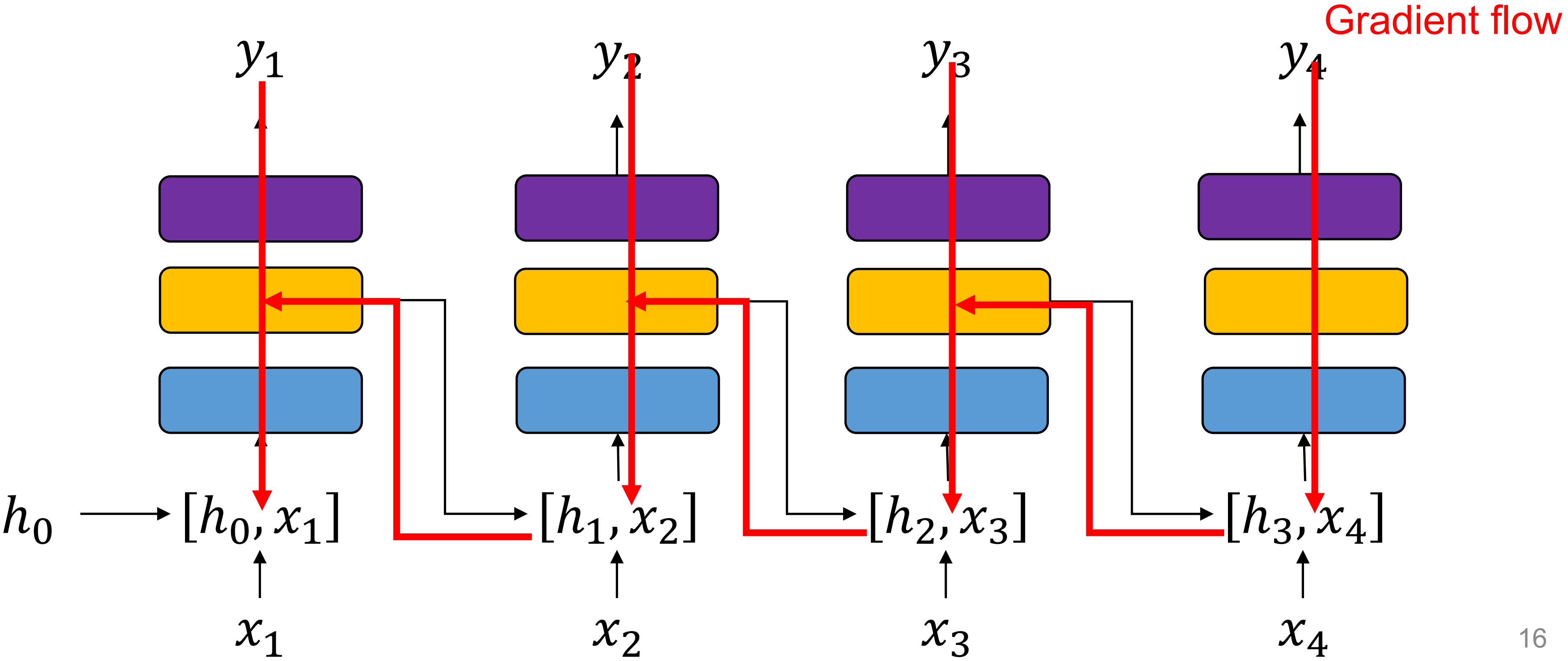
Recurrent Neural Nets (RNNs)



Unrolling RNNs



Backpropagation through time



RNNs and Vanishing Gradients

- L layer network applied over T timesteps $\rightarrow$ LT effective layers between first input and last output

- For a 10 layer RNN acoustic model to look back 0.5 seconds (~2 syllables) in time, the gradient needs to flow through 10 layers * 50 frames = 500 multiplicative terms

- Very easy for product of 500 numbers to vanish to 0 or blow up to infinity. This is a problem for **all** deep neural nets, but RNNs are especially vulnerable

Long Short-Term Memory (LSTM)

LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. Our experiments with artificial data involve local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms.

RNN's problem: naïve recurrence is unstable to train because of vanishing/exploding gradients

LSTM's solution: Instead of using a feedback loop (from output to input), us an explicit read/write/reset memory register to pass information across timesteps

Long Short-Term Memory (LSTM)

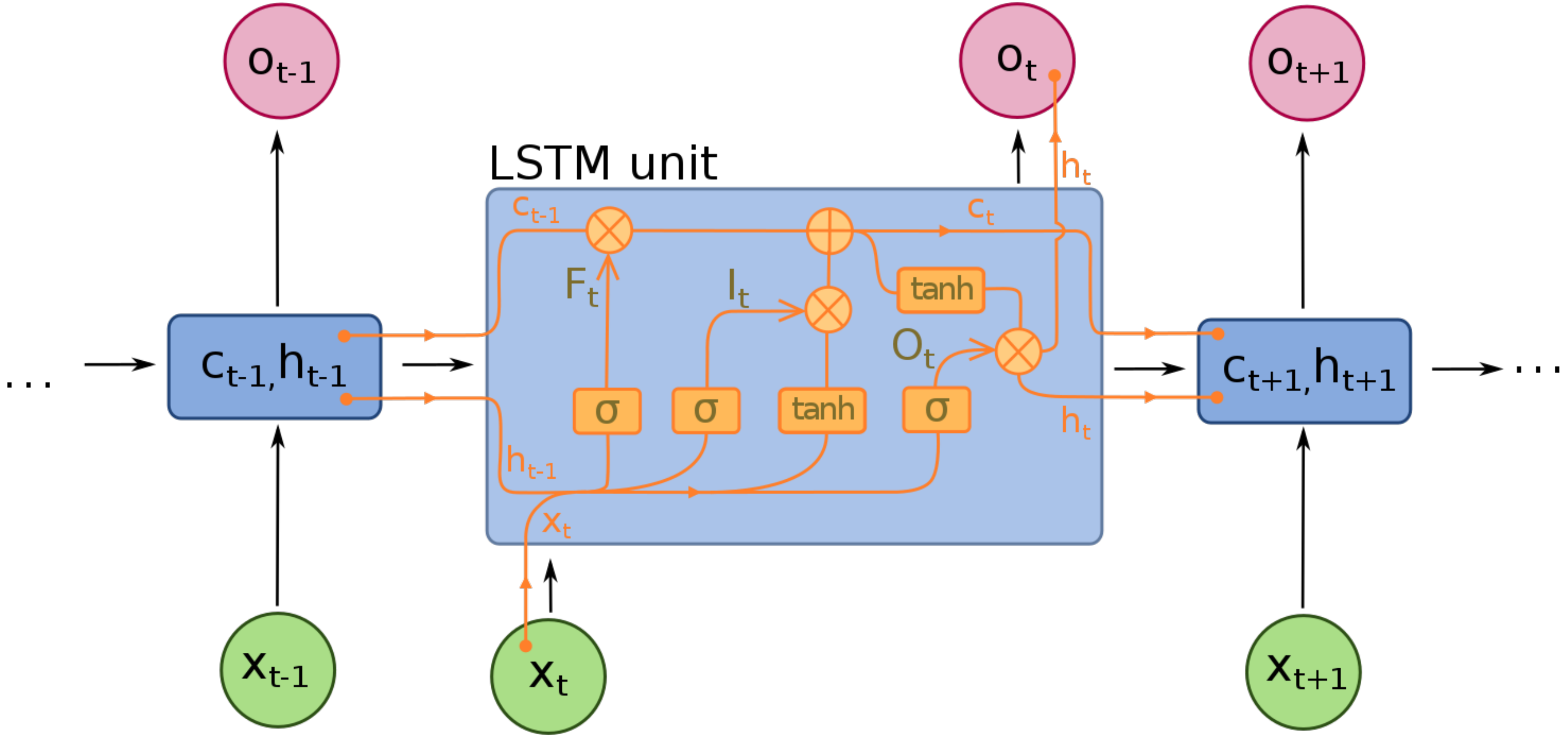
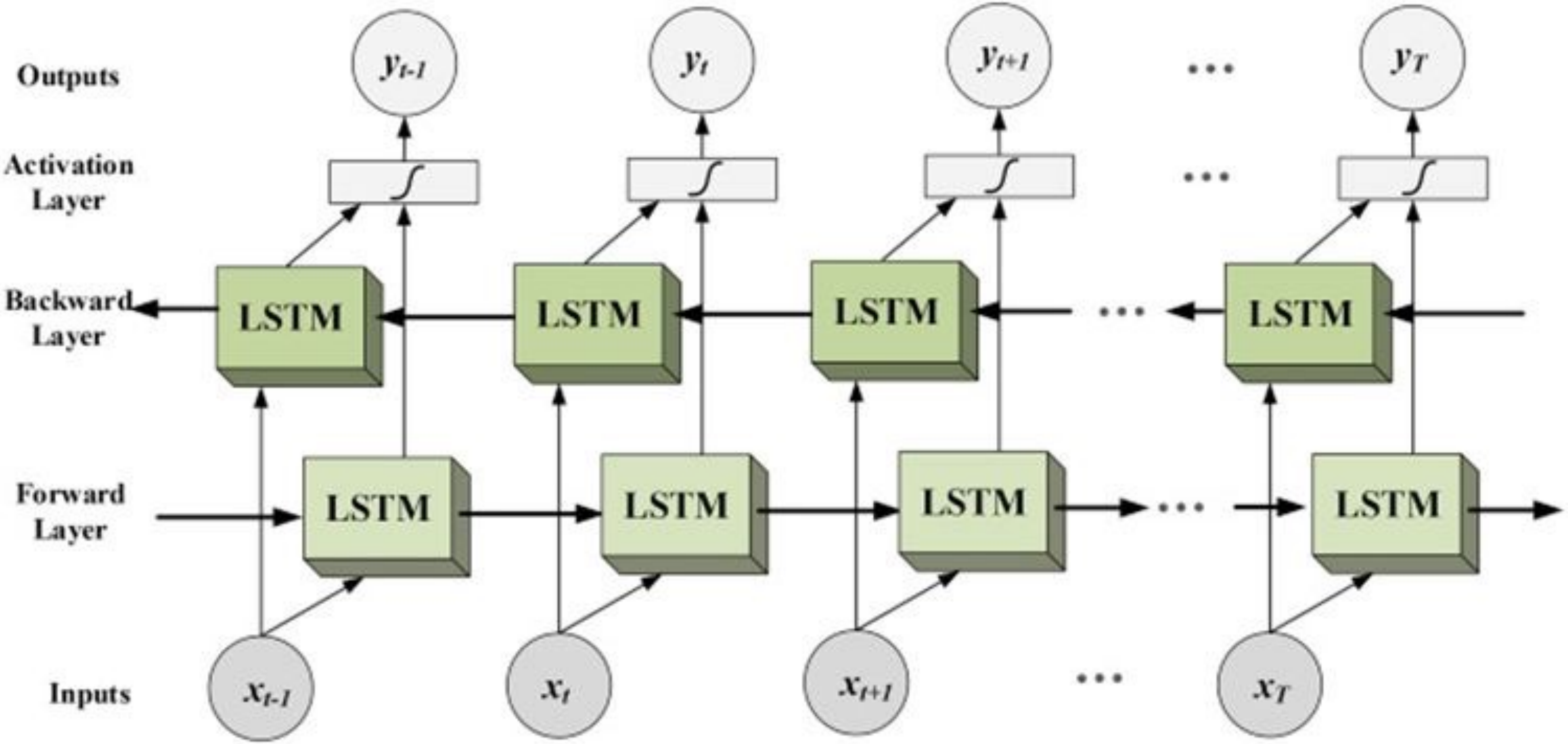


Figure credit: François Deloche

Bidirectional RNNs



ASR Results with LSTMs

HYBRID SPEECH RECOGNITION WITH DEEP BIDIRECTIONAL LSTM

Alex Graves, Navdeep Jaitly and Abdel-rahman Mohamed

University of Toronto
Department of Computer Science
6 King's College Rd. Toronto, M5S 3G4, Canada

Table 3. WSJ Results. All results recorded on the dev93 evaluation set. 'WER' is word error rate, 'FER' is frame error rate and 'CE' is cross entropy error in nats per frame.

SYSTEM	WER	FER	CE
DBLSTM	**11.7**	30.0	1.15
DBLSTM (NOISE)	12.0	**28.2**	**1.12**
DNN	12.3	44.6	1.68
SGMM [20]	13.1	–	–

Purely sequence-trained neural networks for ASR based on lattice-free MMI

Daniel Povey[1,2], Vijayaditya Peddinti[1], Daniel Galvez[3], Pegah Ghahrmani[1], Vimal Manohar[1], Xingyu Na[4], Yiming Wang[1], Sanjeev Khudanpur[1,2]

Table 3: Performance of LF-MMI with different models on the Hub5 '00 eval set, using SWBD-300 Hr data

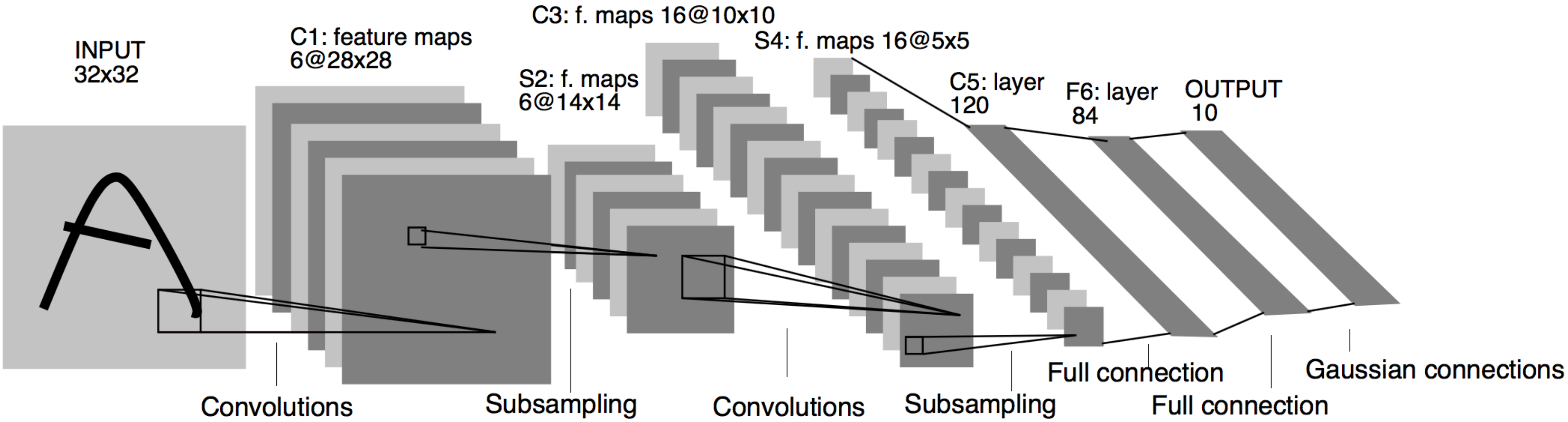
Model	WER	
	Total	SWBD
TDNN-C + CE	18.2	12.5
TDNN-C + LF-MMI	15.5	10.2
LSTM + CE	16.5	11.6
LSTM + LF-MMI	15.6	10.3
BLSTM + CE	14.9	10.3
BLSTM + LF-MMI	14.5	9.6

Convolutional Neural Nets (CNNs)

PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

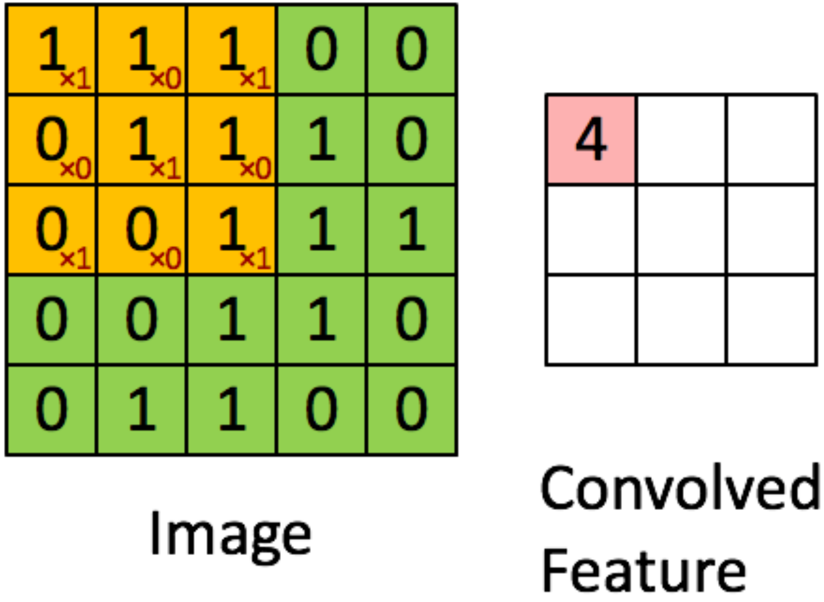


Convolution Layers

Essentially a sliding linear layer that only sees part of its input at a time.

Far fewer parameters than a full linear layer, plus reflects a shift-invariant prior over learned features

$$a[n] * b[n] = \sum_{m=-N}^{N} a[n-m]b[m]$$



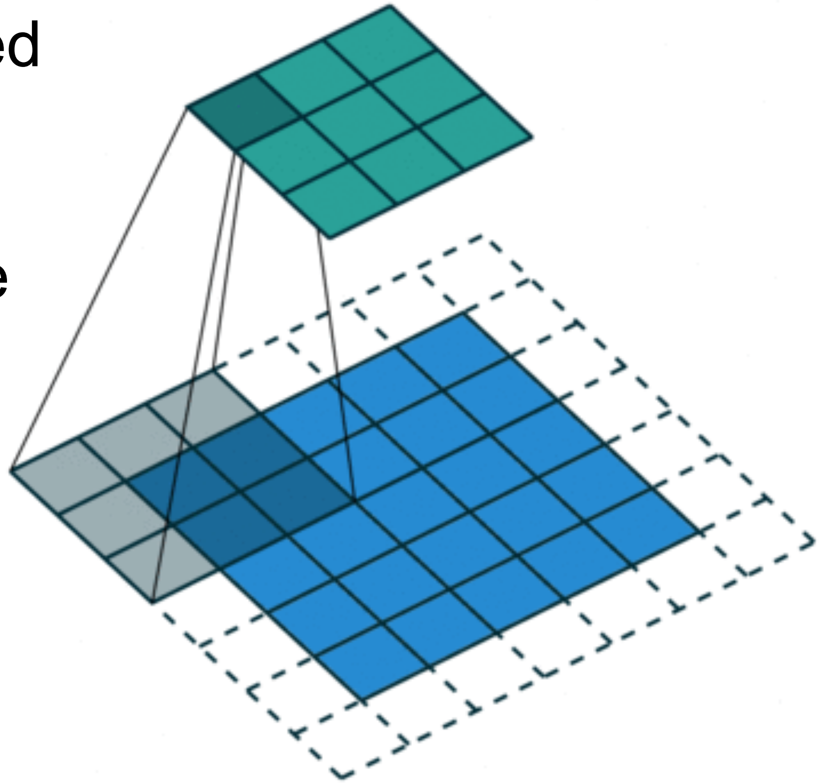
Convolution Layers

Each convolutional filter is called a *kernel*

Width/height (for 2D) determine size

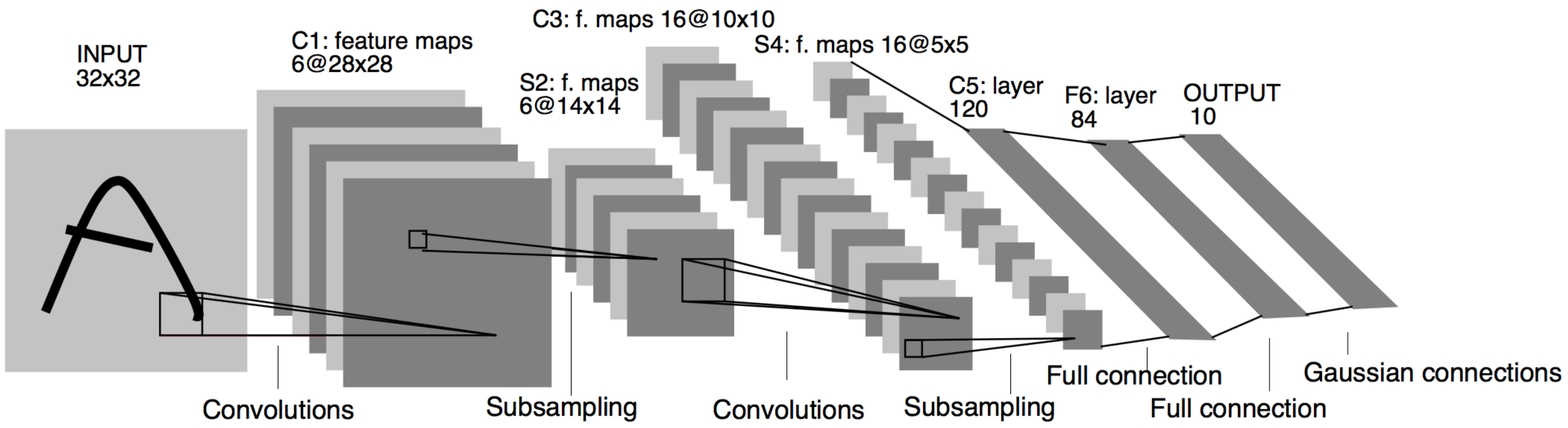
Shift (stride) determines downsampling ratio

Zero-padding used at edges



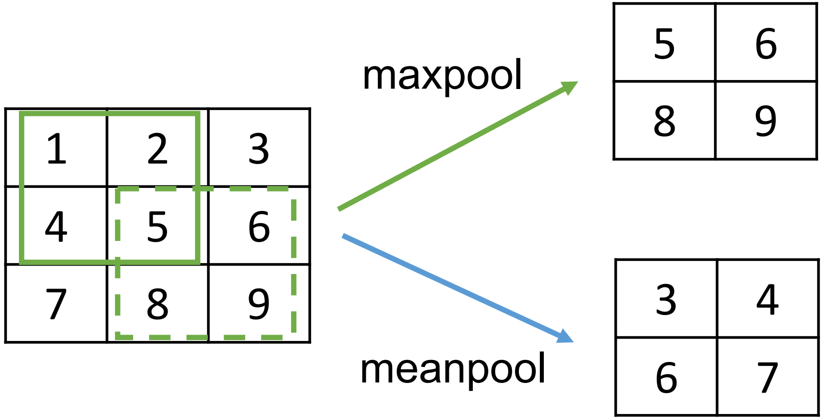
Convolution Layers

Each convolutional layer usually uses multiple convolutional filters in parallel – call these *channels*



Pooling layers

- Similar to convolution in the sense that we slide a fixed-size window over a feature map
- No parameters – instead, we use some function like mean() or max() applied within the window



ASR Results with CNNs (2013)

DEEP CONVOLUTIONAL NEURAL NETWORKS FOR LVCSR

Tara N. Sainath[1], *Abdel-rahman Mohamed*[2], *Brian Kingsbury*[1], *Bhuvana Ramabhadran*[1]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A.
[2]Department of Computer Science, University of Toronto, Canada
[1]{tsainath, bedk, bhuvana}@us.ibm.com, [2]asamir@cs.toronto.edu

model	dev04f	rt04
Baseline GMM/HMM	18.8	18.1
Hybrid DNN	16.3	15.8
DNN-based Features	16.7	16.0
Hybrid CNN	15.8	15.0
CNN-based Features	**15.2**	**15.0**

Table 5. WER for NN Hybrid and Feature-Based Systems

(Trained on 50h from Broadcast News)

Hybrid model: Use DNN or CNN senone likelihoods for HMM states as input to FST decoder

Tandem model: extract embeddings from DNN or CNN, then use them instead of MFCCs in a regular GMM-HMM system

	Hub5'00	rt03	
model	SWB	FSH	SWB
Baseline GMM/HMM	14.5	17.0	25.2
Hybrid DNN	12.2	14.9	23.5
CNN-based Features	**11.5**	**14.3**	**21.9**

Table 7. WER on Switchboard, 300 hrs