

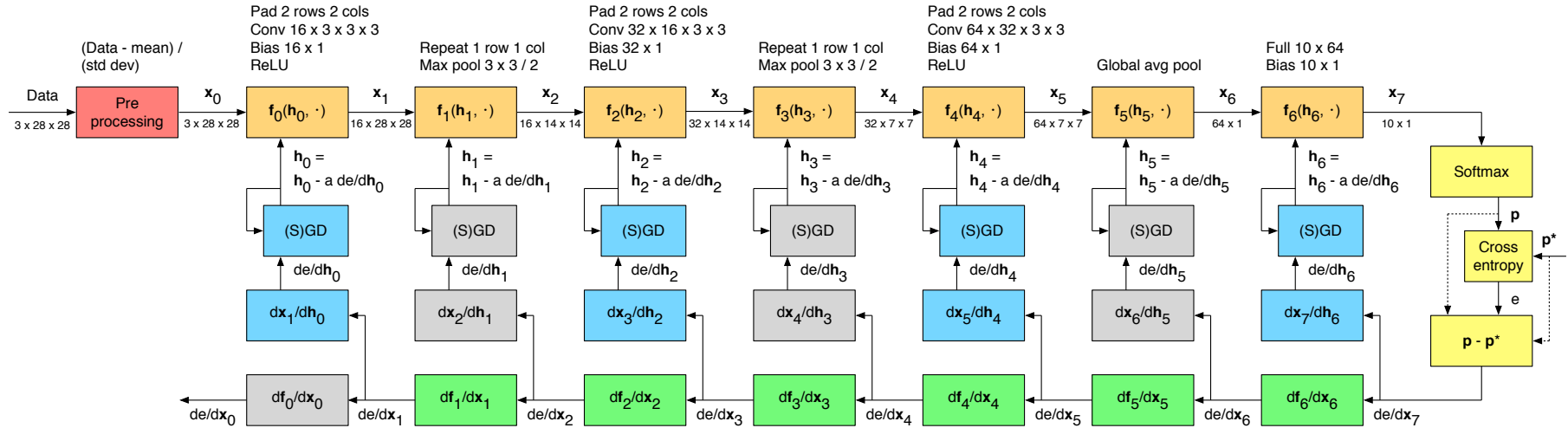
Test Study Guide

Arthur J. Redfern

axr180074@utdallas.edu

Sep 26, 2018

Example Network



Introduction

- 4 steps in an information extraction framework (data to information)
 - Pre processing
 - Feature extraction
 - Prediction
 - Post processing
- Pre and post processing
 - Tend to be application dependent
- Feature extraction and prediction
 - Can use CNNs for feature extraction and prediction for many applications
 - Use machine learning to learn CNN parameters from data

Linear Algebra

- CNN style 2D convolution

- With an input size of $N_i \times L_r \times L_c$
- With an input 0 pad of $(F_r - 1)$ rows and $(F_c - 1)$ cols
- With a filter size of $N_o \times N_i \times F_r \times F_c$
- What is the output size (note zero padding of $F - 1$)?
- What is the equivalent matrix problem size?
- How many MACs not taking advantage of 0s in pad?

$$N_o \times L_r \times L_c$$

$$M_{\text{BLAS}} = N_o, N_{\text{BLAS}} = L_r L_c, K_{\text{BLAS}} = N_i F_r F_c$$

$$(M_{\text{BLAS}} \times N_{\text{BLAS}}) = (M_{\text{BLAS}} \times K_{\text{BLAS}}) * (K_{\text{BLAS}} \times N_{\text{BLAS}})$$

$$N_o N_i F_r F_c L_r L_c$$

- Matrix vector multiplication

- With an input size of $N_i \times 1$
- With a filter size of $N_o \times N_i$
- What is the output size?
- Can you use an inner product to write output m ?
- How many weights?

$$N_o \times 1$$

$$\langle \mathbf{H}(m, :)^T, \mathbf{x} \rangle, \text{ strength } ||\mathbf{H}(m, :)||_2, \text{ alignment } \theta$$

$$N_o N_i$$

Calculus

- Gradient

- Start with a scalar function of multiple variables $y = f(\mathbf{x}) = f(x_0, x_1, \dots, x_{K-1})$
- Gradient is the partial derivative with respect to each variable
- Points in direction of maximum change of function
- $\nabla f(\mathbf{x}) = [(\partial f / \partial x_0) (\partial f / \partial x_1) \dots (\partial f / \partial x_{K-1})]^T$
- Compute the gradient of the error with respect to the final output

- Error gradient propagation

- How do we propagate? Reverse mode automatic differentiation / chain rule from calculus
- What does it do? Constructs a graph that propagates the error gradient from the end to the beginning
- How does it work? $\partial e / \partial \mathbf{x}_d = (\partial \mathbf{x}_{d+1} / \partial \mathbf{x}_d) (\partial e / \partial \mathbf{x}_{d+1}) = (\partial \mathbf{f}_d / \partial \mathbf{x}_d) (\partial e / \partial \mathbf{x}_{d+1})$

- Parameter update

- How do we update? Gradient descent (later, many variants)
- What does it do? Update the parameters in a small step in the opposite direction of the error gradient
- How does it work? $\partial e / \partial \mathbf{h}_d = (\partial \mathbf{x}_{d+1} / \partial \mathbf{h}_d) (\partial e / \partial \mathbf{x}_{d+1}), \mathbf{h}_d \leftarrow \mathbf{h}_d - \alpha \partial e / \partial \mathbf{h}_d$

Probability

- Input normalization

- Assume the input \mathbf{x} has mean μ and std dev σ
- How can you normalize \mathbf{x} to 0 mean unit variance?

$$\mathbf{x} \leftarrow (\mathbf{x} - \mu) / \sigma$$

- Soft max cross entropy error

- What does soft max do?
- What does cross entropy do for a 1 hot input?
- What is the soft max cross entropy error gradient?

Converts network output to a \sim PMF

$$\mathbf{p} = f(\mathbf{x}) = (1/(\sum_k e^{x(k)})) [e^{x(0)} e^{x(1)} \dots e^{x(K-1)}]^T$$

Divergence between network PMF and true PMF \mathbf{p}^*

$$e = f(\mathbf{p}^*, \mathbf{p}) = -\sum_k p^*(k) \log(p(k))$$

$\mathbf{p} - \mathbf{p}^*$ where \mathbf{p} is the network PMF and \mathbf{p}^* is the true PMF

- Feature map compression

- Assume feature map elements are independent
- And all have the same PMF and 8 bit quantization
- $p_x(0) = 0.5, p_x(!= 0) = 0.5 / 255$
- What is the entropy bound for compression?

$$H(X) = -(0.5 \log_2(0.5)) - ((255) (0.5/255) \log_2(0.5/255)) \\ \sim 5 \text{ bits per element}$$

Algorithms

- Pooling

- What does it do? Reduces the spatial resolution of the feature maps
- What else? Increases the receptive field size

- Sequential comparison sort

- For unknown input require $O(N \log_2(N))$ comparisons
- Can you outline a short proof of this bound?
- Proof outline
 - There are $N!$ possible arrangements of a sequence of length N
 - View the arrangements as a random variable $X(s)$
 - The probability of each arrangement is $1/N!$
 - Uniform probability mass function with support of size $N!$
 - The entropy (information) of a realization of this random variable
 - $H(X(s)) = -\sum (1/N!) \log_2(1/N!) = \log_2(N!)$
 - Each comparison in a comparison sort gives at most 1 bit of information
 - To reduce the entropy to 0 with C comparisons need $\log_2(N!) - C \leq 0$
 - $C \geq \log_2(N!) \approx O(N \log_2(N))$ via Stirling's approximation

Design

- Tail

- What is a common tail design?

64 x 3 x 7 x 7 / 2 conv, 3 x 3 / 2 max pool

- Body

- How is CNN convolution commonly split?
 - Why?
 - How does a residual building block work?
 - Why?

Standard convolution (spatial), 1x1 CNN convolution (channel)

Save computation and memory, still get spatial and channel mixing

$$\mathbf{x}_{d+1} = \mathbf{f}_d(\mathbf{x}_d) = \mathbf{x}_d + \mathbf{h}_d(\mathbf{x}_d)$$

Error gradient propagates as identity + perturbation, allows deeper nets

- Head

- What is a common head design?
 - What is assumed in this?

Global avg pool, matrix vector mult, bias add, soft max or arg max

Output classes are linearly (affine) separable from final features

- Receptive field size

- What is the receptive field size at \mathbf{x}_5 ?
 - How was that calculated?

$((1 + 2) * 2 + 2 + 2) * 2 + 2 + 2 = 24$ pixels in the original input

Start at end with 1, filter adds $F - 1$ and down sampling multiplies