Exploring Complexity Reduction in Deep Learning

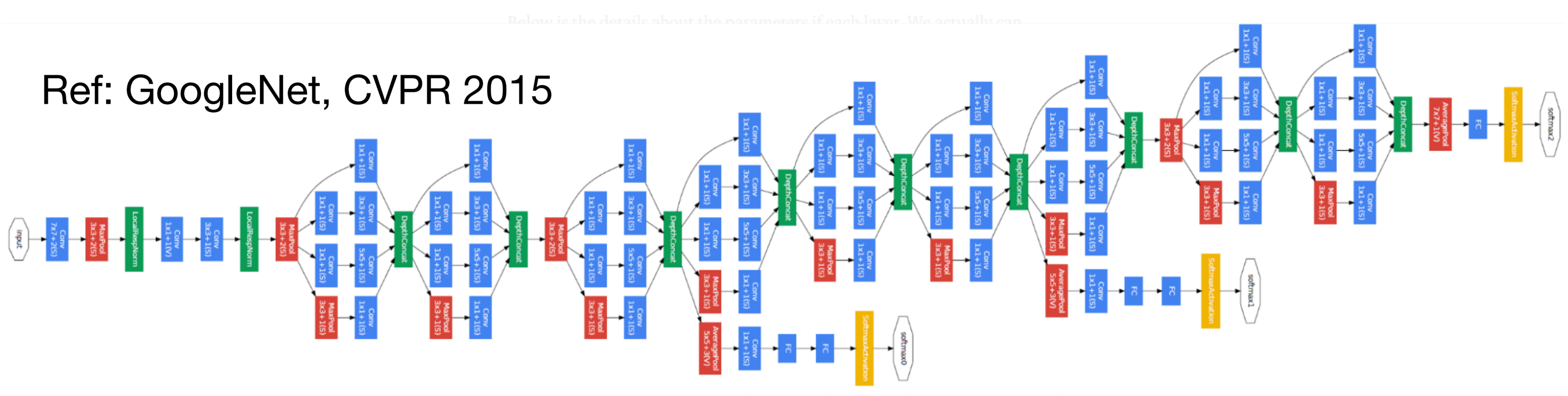
Sourya Dey, Peter Beerel, Keith Chugg

(1) Problem Statement

Neural networks *need a lot of manual tuning*
- Architecture, layers (discrete)
- Hyperparameter values (continuous)

Neural networks have *massive complexity*

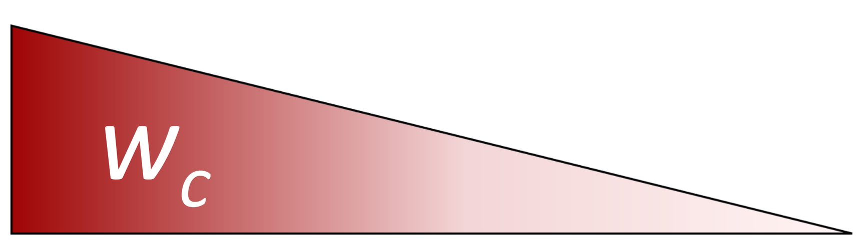
Ref: GoogleNet, CVPR 2015



Our research goal: ***Automate the search for low complexity networks which give good performance***

Optimization objective:

$f = f_p(\text{Performance}) + w_c * f_c(\text{Complexity})$



Quick to train
Bad performance

Good performance
Too long to train

Current focus: CNNs

(2) Approaches

Search space is both continuous and discrete
Each point ***x*** is a neural network to be trained
Evaluating f is expensive and noisy!

Potential approaches
- Simulated annealing
- ***Bayesian optimization***
- Evolutionary / genetic algorithms

Sample $f(\cdot)$ and model via a ***Gaussian process***

$$f\left(\boldsymbol{X}_{1:n}\right) \sim \mathcal{N}\left(\underset{n\times 1}{\boldsymbol{\mu}}, \underset{n\times n}{\boldsymbol{\Sigma}}\right) \qquad \boldsymbol{\Sigma} = \begin{bmatrix} k(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1,\boldsymbol{x}_n) \\ \vdots & \text{Covariance kernel } k & \vdots \\ k(\boldsymbol{x}_n,\boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n,\boldsymbol{x}_n) \end{bmatrix}$$

Get potential new networks via ***expected improvement***
- Expensive f evaluations are minimized
- Kernel can model noise

$$EI(\boldsymbol{x}) = (f^* - \mu)P\left(\frac{f^* - \mu}{\sigma}\right) + \sigma p\left(\frac{f^* - \mu}{\sigma}\right)$$

f^* = Current optimal value

(3) Research Methodology

Bayesian optimization can fail if search space is too big
Given a problem, ***divide search space into levels***:

- # convolutional layers
- # channels in each
- Downsampling (strides/pooling)

Level 1: Basic structure

- Kernel sizes
- Batch normalization (yes/no)
- Grouped convolutions

Level 2: Parameter adjustments

- # classification layers
- Densities
- Weight decay coefficients

Level 3: Classifier

- Learning rate
- Learning rate decay
- Batch size

Level 4: Training hyperparameters

Our prior work on pre-defined sparsity

S. Dey, K. Huang, P. A. Beerel and K. M. Chugg, "Pre-Defined Sparse Neural Networks with Hardware Acceleration," in *IEEE JETCAS*, vol. 9, no. 2, pp. 332-345, June 2019.

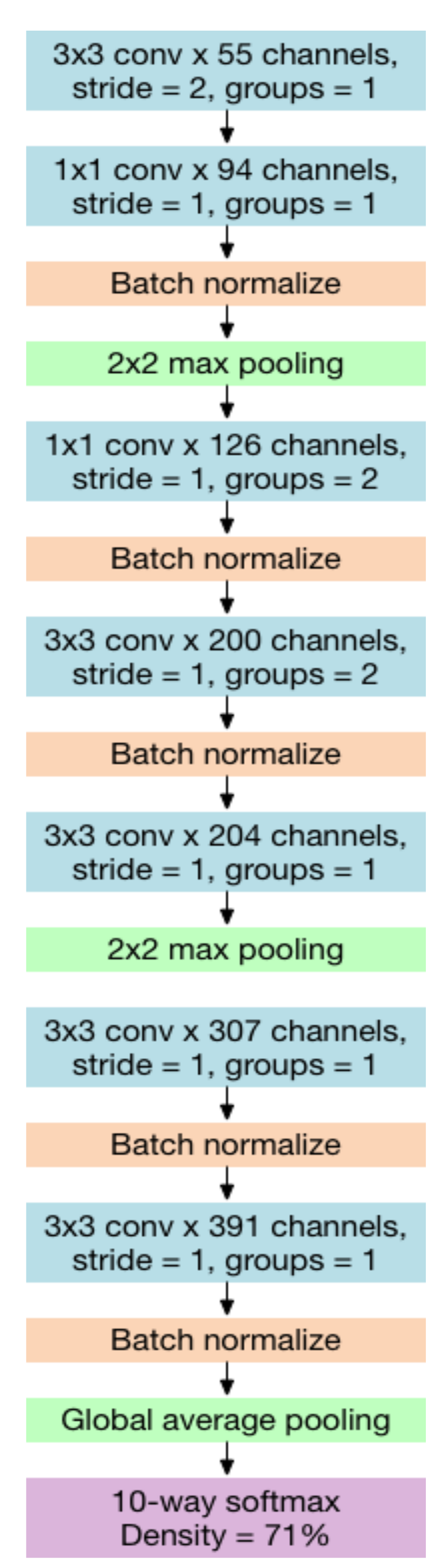
f_p(Performance) = 1 - Best validation acc

f_c(Complexity) = Normalized training time per epoch

Can set w_c according to desired tradeoff

(4) Results so far

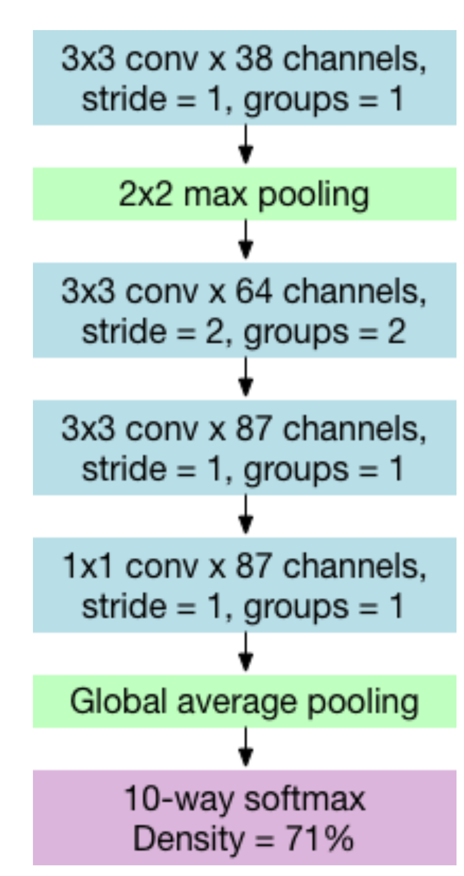
w_c = 0.1 (Balanced case)



Learning rate = 4.4e-3, Decay = 0.992
Weight decay = 2.3e-3, Batch size = 338

Best val acc = 82% in 30 eps

w_c = 1 (More focus on low complexity)



Learning rate = 4.3e-3, Decay = 0.999
Weight decay = 4e-4, Batch size = 501

Best val acc = 74% in 30 eps

Dataset used: CIFAR-10 images of size 32x32 x 3 channels (no aug)

Contact Information: souryade@usc.edu, pabeerel@usc.edu, chugg@usc.edu

This work is partly supported by National Science Foundation, USA, Grant #1763747, and by Defense Threat Reduction Agency, USA.

Information Sciences Institute

USC Viterbi School of Engineering