Superconvergence

Kazem Safari and Wilfredo Molina

University of Houston

April 22, 2020

Outline

The Task

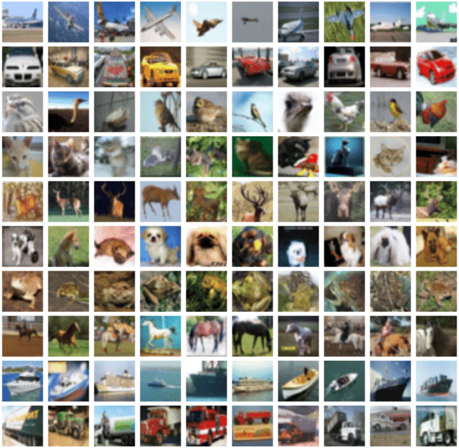
The Dataset

The Model

The *One Cycle Learning Rate* (OCLR)

The Results

Data



The Task: Image Classification

Dataset: Images and labels

Model: A mapping from images to labels

Train: Fit parameters of our model to training set

Test: Validate *generalizability* of the model on test set

Our Goal

Improve the test classification performance using the superconvergence technique.

Quick, Draw! Doodle Recognition Challenge

Quick, Draw! Doodle Recognition Challenge

50 million training images

112 thousand test images

340 categories (classes)

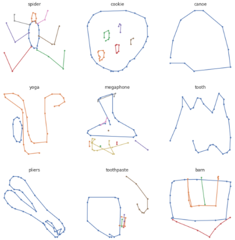
Challenges

Noisy Labels

Datapoints are not images!!! (per se)

Tensorflow: Takes more than 2TB of space.

Pytorch: Drawn on the fly as 128×128 RGB images.



Goodbye Tensorflow!!!

Challenges

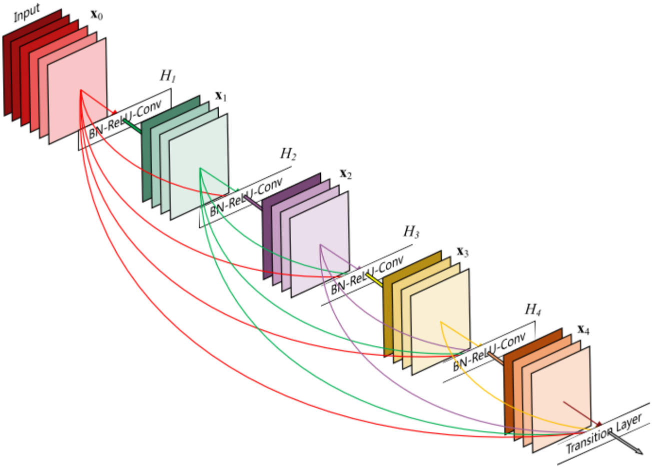
(Until 2 month ago), training time $\approx$ 2 days

Now, training time $\approx$ 3.5 hours

How did you do it?!!!

Magic

Neural Network (NN)



Neural Network (NN)

A composition of two functions:

1) A feature map:

Maps data points to a high-dimensional space (Kernel).

2) A classifier:

Maps the extracted features to labels (SVM).

However, they are trained simultaneously.

Neural Network (NN)

$$\phi = \psi \circ \phi_L \circ ... \circ \phi_1 : \mathbb{R}^{n_1} \to \mathbb{R}^{n_L}$$

n_1, n_L	input and output dimensions
L	number of layers
ψ	classifier

$$\phi_{k+1} = R_k(B_k(w_k * \phi_k + b_k))$$

$$\phi_1 \in \mathbb{R}^{n_1}$$

$$w_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}, \text{ weight matrix}$$

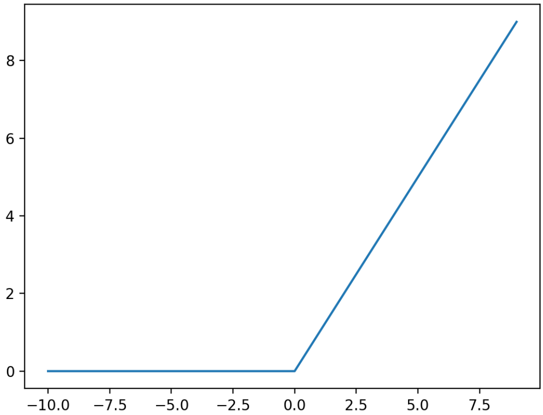
$$b_k \in \mathbb{R}^{n_{k+1}}, \text{ bias}$$

$$R_k : \mathbb{R} \to \mathbb{R}, \quad \text{ReLU } = \max(0, \mathrm{id})$$

$$B_k : \text{batch normalization}$$

$$* : \text{ operator}$$

Rectified Linear Unit (ReLU)



Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

From the original batch-norm paper

The * Operator

Convolution:

$$x * w[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} x[u,v].w[i-u, j-v]$$

Cross-Correlation:

$$x * w[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} x[u,v].w[i+u, j+v]$$

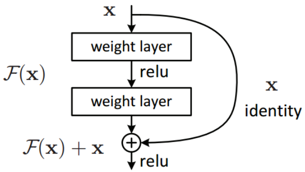
where k is the number of rows of square matrix x.

Convolutional Neural Network (CNN)

An NN where the $*$ operator is the convolution
(cross-correlation in practice).

Residual Neural Network (ResNet)

skip connections:

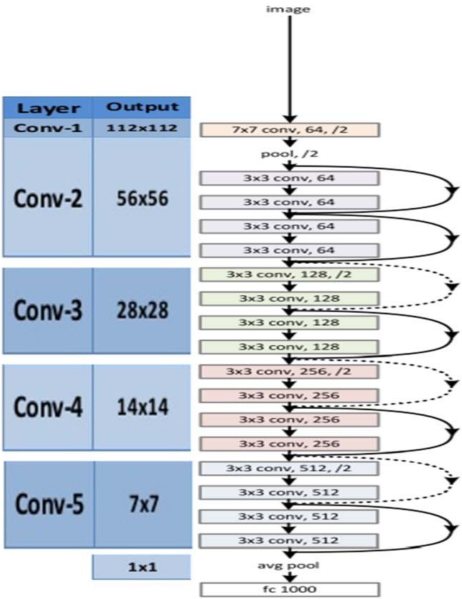


Mathematically

$$\phi_{k+1} = R_k(w_k * \phi_k + b_k) + \phi_{k-1}.$$

It mitigates vanishing gradients.

Our Model: ResNet18



Our Training Objective

To minimize the average cross entropy loss:

$$J(x, y; \theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} y_{i,j} log(p_{i,j}),$$

θ: Trainable parameters

(x, y): (Input images, Input labels)

$p_{i,j}$: Model's prediction probability of input image x_i belonging to class j

N: Total number of inputs

L: Total number of classes (340)

Batchwise Training

Memory Divide dataset into smaller pieces.

Batch Each such piece.

Train-step Feeding a batch to the model.

Epoch Feeding the entire dataset once to the model

A Training Step ($\boldsymbol{j}$)

Given a batch of examples (x_i, y_i) $i = 1, ..., m$

Feed-forward:

$$J = \frac{1}{m} \sum_{i=1}^{m} J(x_i, \theta)$$

Backprop:

$$\nabla J_\theta = \frac{1}{m} \sum_{i=1}^{m} \nabla J_\theta(x_i, \theta)$$

Update Rule (Gradient Descent):

$$\theta^{(\boldsymbol{j}+1)} \leftarrow \theta^{(\boldsymbol{j})} - \boldsymbol{\eta} \nabla_\theta J$$

Learning Rate (η)

How can we tweak η to improve performance?

Literature on Learning Rate (η)

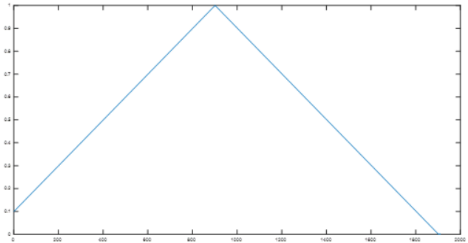
Exponential or Linear Decay

Cyclic Learning Rate (CLR) *(Leslie N. Smith)*

One Cycle Learning Rate (OCLR) *(Leslie N. Smith)*

OCLR and Superconvergence

Leslie Leslie N. Smith



I am sorry for the confusion on the 1cycle policy. It is one cycle but I let the cycle end a little bit before the end of training (and keep the learning rate constant at the smallest value) to allow the weights to settle into the local minima.

η during training

The Metric

Mean Average Precision@3 (MAP@3):

$$MAP@3 = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{3} P_i(k)$$

N : the number of test data

$P_i = \{P_i(k)\}_{k=1}^{3}$: model's top 3 predictions for test data point i

Quick, Draw!

	Private Score	
	Conventional	OCLR
epoch=1	.81595	**.90641**
epoch=2	.85421	**.92928**

References

Thank you!