Regularization of Neural Networks using DropConnect

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, Rob Fergus

Dept. of Computer Science, Courant Institute of Mathematical Science, New York University

June 17, 2013

NEW YORK UNIVERSITY

Introduction

- Neural Networks are good at classifying large labeled datasets
- Large capacity is essential: more layers and more units
- But without regularization, model with millions or billions of parameters can easily overfit
- Existing regularization methods:
  - ℓ_1 or ℓ_2 penalty
  - Bayesian methods
  - Early stopping of training
  - DropOut network [Hinton et al. 2012]

Introduction

- Neural Networks are good at classifying large labeled datasets
- Large capacity is essential: more layers and more units
- But without regularization, model with millions or billions of parameters can easily overfit
- Existing regularization methods:
    - ℓ_1 or ℓ_2 penalty
    - Bayesian methods
    - Early stopping of training
    - DropOut network [Hinton et al. 2012]
- We introduce a new form of regularization: DropConnect

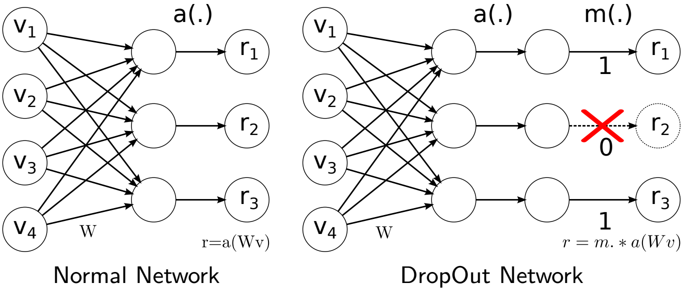
Overview

Review of DropOut Network [Hinton et al. 2012]

- Stochastic dropping of units
- Each element of a layer's output is kept with probability p, otherwise being set to 0 with probability $(1 - p)$
- Input v, weights W, activation function $a(.)$, output r and DropOut mask m:

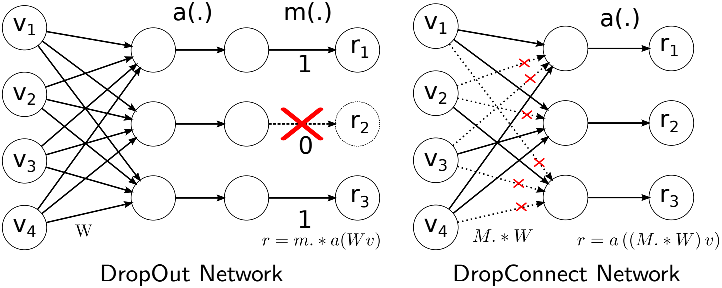
$$r = m \,.\!*\, a(Wv)$$

- For *every* training example at *every* epoch has different mask m



DropConnect Network

- Applies only to fully-connected layers
- Randomly drop *connections* in network, with probability $1 - p$
- Generalization of Dropout: $r = a((M .* W)v)$ (for $a(0) = 0$, e.g. *relu*)



DropConnect Network: Training and Inference

Training

- For *every* training example at *every* epoch has different binary mask matrix M
- Backward-prop gradient uses the same M as forward-prop, for each example
- Use SGD with mini-batch
- Efficient implementation requires care

DropConnect Network: Training and Inference

Training

- For *every* training example at *every* epoch has different binary mask matrix M
- Backward-prop gradient uses the same M as forward-prop, for each example
- Use SGD with mini-batch
- Efficient implementation requires care

Inference

- Exact solution is intractable
- Approximate neuron activation by Gaussian distribution

DropConnect Network: Inference

DropConnect Network Inference

Exact solution requires sum over all possible $2^{|M|}$ masks M:

$$r = \mathbf{E}_M[a((M .* W)v)] = \sum_M p(M)a((M .* W)v)$$

DropConnect Network: Inference

DropConnect Network Inference

Exact solution requires sum over all possible $2^{|M|}$ masks M:

$$r = \mathbf{E}_M[a((M .* W)v)] = \sum_M p(M)a((M .* W)v)$$

Inference Approximation

A single neuron u_i before activation function: $u_i = \sum_j (W_{ij}v_j)M_{ij}$. Approximate u_i by a Gaussian distribution via moment matching.

$$u \sim \mathcal{N}(pWv, p(1-p)(W .* W)(v .* v))$$

where $1 - p$ is drop connect rate

DropConnect Network: Inference

DropConnect Network Inference

Exact solution requires sum over all possible $2^{|M|}$ masks M:

$$r = \mathbf{E}_M[a((M .* W) v)] = \sum_M p(M) a((M .* W) v)$$

Inference Approximation

A single neuron u_i before activation function: $u_i = \sum_j (W_{ij} v_j) M_{ij}$. Approximate u_i by a Gaussian distribution via moment matching.

$$u \sim \mathcal{N}(pWv, p(1-p)(W .* W)(v .* v))$$

where $1 - p$ is drop connect rate

Inference Algorithm

- Compute layer response at test time: $r \approx \mathbf{E}_u[a(u)]$ by sampling or numerical integration
- Each neuron activation sampled independently, thus very efficient

Inference comparison: DropOut v.s. DropConnect

DropOut Network Inference[Hinton et al. 2012]

Approximate by changing the order of expectation and neuron activation:

$$\mathbf{E}_M[a((M .* W)v] \approx a(\mathbf{E}_M(M .* W)v) = a(pWv)$$

Inference comparison: DropOut v.s. DropConnect

DropOut Network Inference[Hinton et al. 2012]

Approximate by changing the order of expectation and neuron activation:

$$\mathbf{E}_M[a((M .* W)v] \approx a(\mathbf{E}_M(M .* W)v) = a(pWv)$$

Failure Example

$u \sim \mathcal{N}(0,1)$ with $a(u) = max(u,0)$. $a(\mathbf{E}_M(u)) = 0$ while $\mathbf{E}_u(a(u)) = 1/\sqrt{2\pi} \approx 0.4$

Inference comparison: DropOut v.s. DropConnect

DropOut Network Inference[Hinton et al. 2012]

Approximate by changing the order of expectation and neuron activation:

$$\mathbf{E}_M[a((M .* W)v] \approx a(\mathbf{E}_M(M .* W)v) = a(pWv)$$

Failure Example

$u \sim \mathcal{N}(0,1)$ with $a(u) = max(u,0)$. $a(\mathbf{E}_M(u)) = 0$ while $\mathbf{E}_u(a(u)) = 1/\sqrt{2\pi} \approx 0.4$

DropConnect Network Inference

Approximate by Gaussian moment matching:

$$\mathbf{E}_M[a((M .* W)v] \approx \mathbf{E}_u\left[a(u)\right]$$

gives the right answer for the above example

Why DropConnect Regualize Network

DropConnect Network Model

Model averaging interpretation: a mixture model of $2^{|M|}$ classifiers

$$f(x;\theta) = \mathbf{E}_M\left[f_M(x;\theta,M)\right] = \sum_M p(M) f_M(x;\theta,M)$$

Why DropConnect Regualize Network

DropConnect Network Model

Model averaging interpretation: a mixture model of $2^{|M|}$ classifiers

$$f(x;\theta) = \mathbf{E}_M\left[f_M(x;\theta,M)\right] = \sum_M p(M) f_M(x;\theta,M)$$

Rademacher Complexity of Model

Let W_s be the weights of soft-max layer and W be the weights of DropConnect layer, where $max|W_s| \leq B_s$, $max|W| \leq B$. Define k is the number of classes, $\hat{R}_\ell(\mathcal{G})$ is the Rademacher complexity of the feature extractor (layers before DropConnect layer), n and d are the dimensionality of the input and output of the DropConnect layer respectively.

$$\hat{R}_\ell(\mathcal{F}) \leq p\left(2\sqrt{k}dB_s n\sqrt{d}B_h\right)\hat{R}_\ell(\mathcal{G})$$

Why DropConnect Regualize Network

DropConnect Network Model

Model averaging interpretation: a mixture model of $2^{|M|}$ classifiers

$$f(x;\theta) = \mathbf{E}_M\left[f_M(x;\theta,M)\right] = \sum_M p(M) f_M(x;\theta,M)$$

Rademacher Complexity of Model

Let W_s be the weights of soft-max layer and W be the weights of DropConnect layer, where $max|W_s| \leq B_s$, $max|W| \leq B$. Define k is the number of classes, $\hat{R}_\ell(\mathcal{G})$ is the Rademacher complexity of the feature extractor (layers before DropConnect layer), n and d are the dimensionality of the input and output of the DropConnect layer respectively.

$$\hat{R}_\ell(\mathcal{F}) \leq p\left(2\sqrt{k}dB_s n\sqrt{d}B_h\right)\hat{R}_\ell(\mathcal{G})$$

Special Cases of p

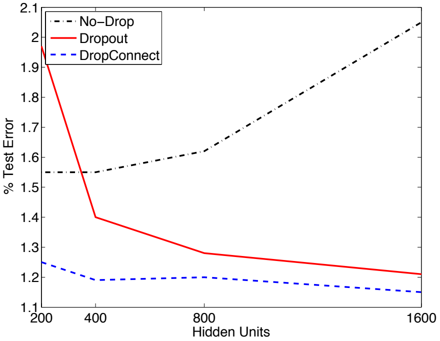
1. $p = 0$: the model complexity is zero, since the input has no influence on the output.
2. $p = 1$: it returns to the complexity of a standard model.
3. $p = 1/2$: all sub-models have equal preference.

Experiments overview

- DataSet: MNIST, CIFAR-10, and SVNH
- Kept Rate $p = 0.5$ for both DropOut and DropcConnect network (except where explicitly stated)
- Normal Network, DropOut Network and DropConnect Network have:
  - exactly the same architecture
  - exactly the same model/training parameters
  - exactly the same data augmentation algorithm
- We use:
  - Data Augmentation: Translation, Rotation and Scaling
  - Aggregating multiple models: 5 or more

Varying Size of Network

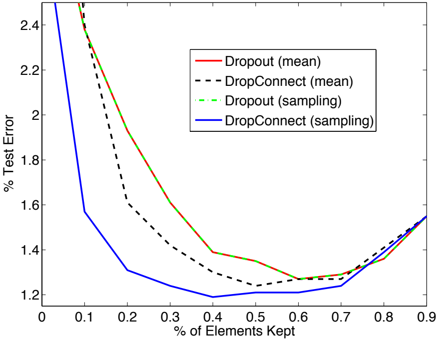
MNIST test error with two hidden layer network ($p = 0.5$)



- No-Drop Network overfits with a large number of neurons
- Both DropConnect and DropOut regularize model nicely
- DropConnect performance better than DropOut and No-Drop

Varying Kept Rate p

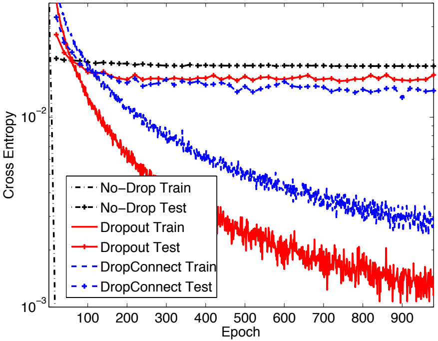
MNIST test error with two hidden layer network with 400 neurons each



- Optimal value of p is roughly at 0.5 for both DropOut and DropConnect
- DropConnect: Sampling inference works better than mean inference

Comparison of Convergence Rates

MNIST test error with two hidden layer network with 400 neurons each



- Cross Entropy, continuous measure of error
- DropConnect slower than DropOut but better test error

MNIST Results(1)

MNIST 784-800-800-10 network classification error rate without data augmentation:

neuron	model	error(%) 5 network	voting error(%)
relu	No-Drop	1.62 ± 0.037	1.40
	Dropout	1.28 ± 0.040	1.20
	DropConnect	1.20 ± 0.034	1.12
sigmoid	No-Drop	1.78 ± 0.037	1.74
	Dropout	1.38 ± 0.039	1.36
	DropConnect	1.55 ± 0.046	1.48
tanh	No-Drop	1.65 ± 0.026	1.49
	Dropout	1.58 ± 0.053	1.55
	DropConnect	1.36 ± 0.054	1.35

- *relu* neuron is always performs better than other neuron
- Error Rate: DropConnect $<$ DropOut $<$ No-Drop for *relu* and *tanh*
- DropOut works best for *sigmod* which does not have $a(0) = 0$.

MNIST Results(2)

MNIST classification error

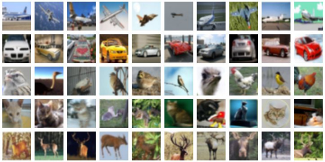
crop	rotation scaling	model	error(%) 5 network	voting error(%)
no	no	No-Drop	0.77 ± 0.051	0.67
		Dropout	0.59 ± 0.039	0.52
		DropConnect	0.63 ± 0.035	0.57
yes	no	No-Drop	0.50 ± 0.098	0.38
		Dropout	0.39 ± 0.039	0.35
		DropConnect	0.39 ± 0.047	0.32
yes	yes	No-Drop	0.30 ± 0.035	0.21
		Dropout	0.28 ± 0.016	0.27
		DropConnect	0.28 ± 0.032	0.21

0.21% is the new state-of-the-art

Previous state-of-the-art is:

- *0.45*% for a single model without elastic distortions [Goodfellow et al. 2013]
- *0.23*% with elastic distortions and voting [Ciresan et al. 2012]

CIFAR-10 Results



- 10 classes of 32 $\times$ 32 colored images
- 50K train/class
- 10K test/class

CIFAR-10 classification error

model	error(%) 5 network	voting error(%)
No-Drop	11.18± 0.13	10.22
Dropout	11.52± 0.18	9.83
DropConnect	11.10± 0.13	9.41

Voting with 12 DropConnect networks produces a new state-of-the-art of 9.32%

Previous state-of-the-art is:

- *11.21*% [Ciresan et al. 2012]
- *9.5*% [Snoek et al. 2012]
- *9.38*% [Goodfellow et al. 2013]

SVHN Results

- 10 classes of 32 × 32 colored image
- 604,388 training images (both training set and extra set)
- 26,032 testing images
- large variety of colors and brightness

SVHN classification error

model	error(%) 5 network	voting error(%)
No-Drop	2.26 ± 0.072	1.94
Dropout	2.25 ± 0.034	1.96
DropConnect	2.23 ± 0.039	1.94

1.94% is the new state-of-the-art

Previous state-of-the-art is:

- *2.8*% stochastic pooling[Zeiler et al. 2013]
- *2.47*% maxout network[Goodfellow et al. 2013]

How-to Implement DropConnect Layer

Implementation	Mask Weight	Time(ms)				Speedup
		fprop	bprop acts	bprop weights	total	
CPU	float	480.2	1228.6	1692.8	3401.6	1.0 ×
CPU	bit	392.3	679.1	759.7	1831.1	1.9 ×
GPU	float(global memory)	21.6	6.2	7.2	35.0	97.2 ×
GPU	float(tex1D memory)	15.1	6.1	6.0	27.2	126.0 ×
GPU	bit(tex2D aligned memory)	2.4	2.7	3.1	8.2	414.8 ×

- NVidia GTX580 GPU relative to a 2.67Ghz Intel Xeon (compiled with `-O3` flag).
- Input and output dimension: 1024 and mini-batch size: 128
- Tricks:
  - encode connection information with bits
  - bind mask weight matrix to 2D texture memory
- CUDA code avaiable at `http://cs.nyu.edu/~wanli/dropc/`

Conclusion

We introduced DropConnect Network

- A simple stochastic regularization algorithm for neural network
- Generalization of DropOut
- Only effective on fully-connected layers
- Set new state-of-the-art on three popular data sets