Dropout: A simple way to improve neural networks

Geoffrey Hinton & George Dahl
Department of Computer Science
University of Toronto

What has happened to neural nets since 1985

- Computers got faster.
- Labeled datasets got bigger.
- We found better ways to initialize the weights of a deep net using unlabeled data.
- As a result of all three factors, deep neural nets are now state of the art for tasks like speech recognition and object recognition.

Is there anything we cannot do with very big, deep neural networks?

- It appears to be hard to do massive model averaging:
  - Each net takes a long time to learn.
  - At test time we don't want to run lots of different large neural nets.

Averaging many models

- To win a machine learning competition (e.g. Netflix) you need to use many different types of model and then combine them to make predictions at test time.

- Decision trees are not very powerful models, but they are easy to fit to data and very fast at test time.
  - Averaging many decision trees works really well. Its called random forests.
  - We can make the individual trees different by giving them different training sets.

Two ways to average models

- We can combine models by averaging their output probabilities:

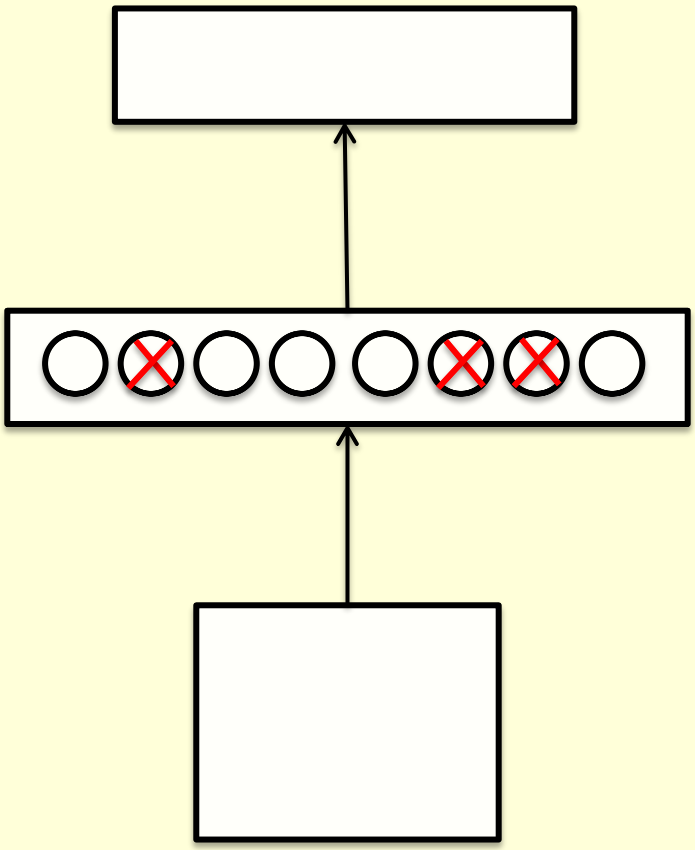
	class 1	class 2	class 3
Model A:	.3	.2	.5
Model B:	.1	.8	.1
Combined	.2	.5	.3

- We can combine models by taking the geometric means of their output probabilities:

Model A:	.3	.2	.5	
Model B:	.1	.8	.1	
Combined	$\sqrt{.03}$	$\sqrt{.16}$	$\sqrt{.05}$	/sum

Dropout: An efficient way to average many large neural nets.

- Consider a neural net with one hidden layer.
- Each time we present a training example, we randomly omit each hidden unit with probability 0.5.
- So we are randomly sampling from 2^H different architectures.
  - All architectures share weights.



Dropout as a form of model averaging

- We sample from 2^H models. So only a few of the models ever get trained, and they only get one training example.
- The sharing of the weights means that every model is very strongly regularized.
  - It’s a much better regularizer than L2 or L1 penalties that pull the weights towards zero.
  - It pulls the weights towards what other models want.

But what do we do at test time?

- We could sample many different architectures and take the geometric mean of their output distributions.
- Its faster to use all of the hidden units, but to halve their outgoing weights.
  - This exactly computes the geometric mean of the predictions of all 2^H models.

What if we have more hidden layers?

- Use dropout of 0.5 in every layer.

- At test time, use the “mean net” that has all the outgoing weights halved.

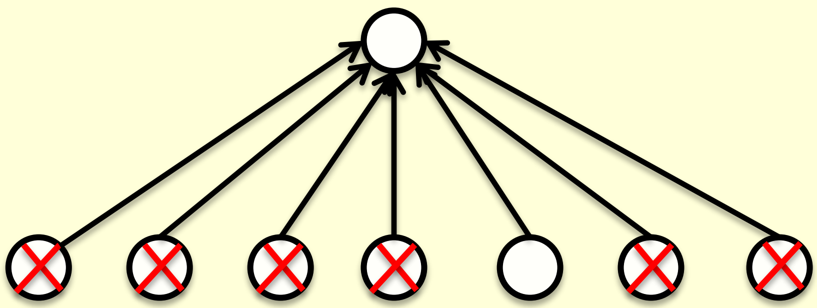
- This is not exactly the same as averaging all the separate dropped out models, but it’s a pretty good approximation, and its fast.

What about the input layer?

- It helps to use dropout there too, but with a higher probability of keeping an input unit.
  - This trick is already used by the "denoising autoencoders" developed in Yoshua Bengio's group.
  - It was derived by a different route.

A familiar example of dropout

- Do logistic regression, but for each training case, dropout all but one of the inputs.



- At test time, use all of the inputs.
  - Its better to divide the learned weights by the number of features, but if we just want the best class its unnecessary.
- This is called "Naïve Bayes".
  - Why keep just one input?

How well does dropout work?

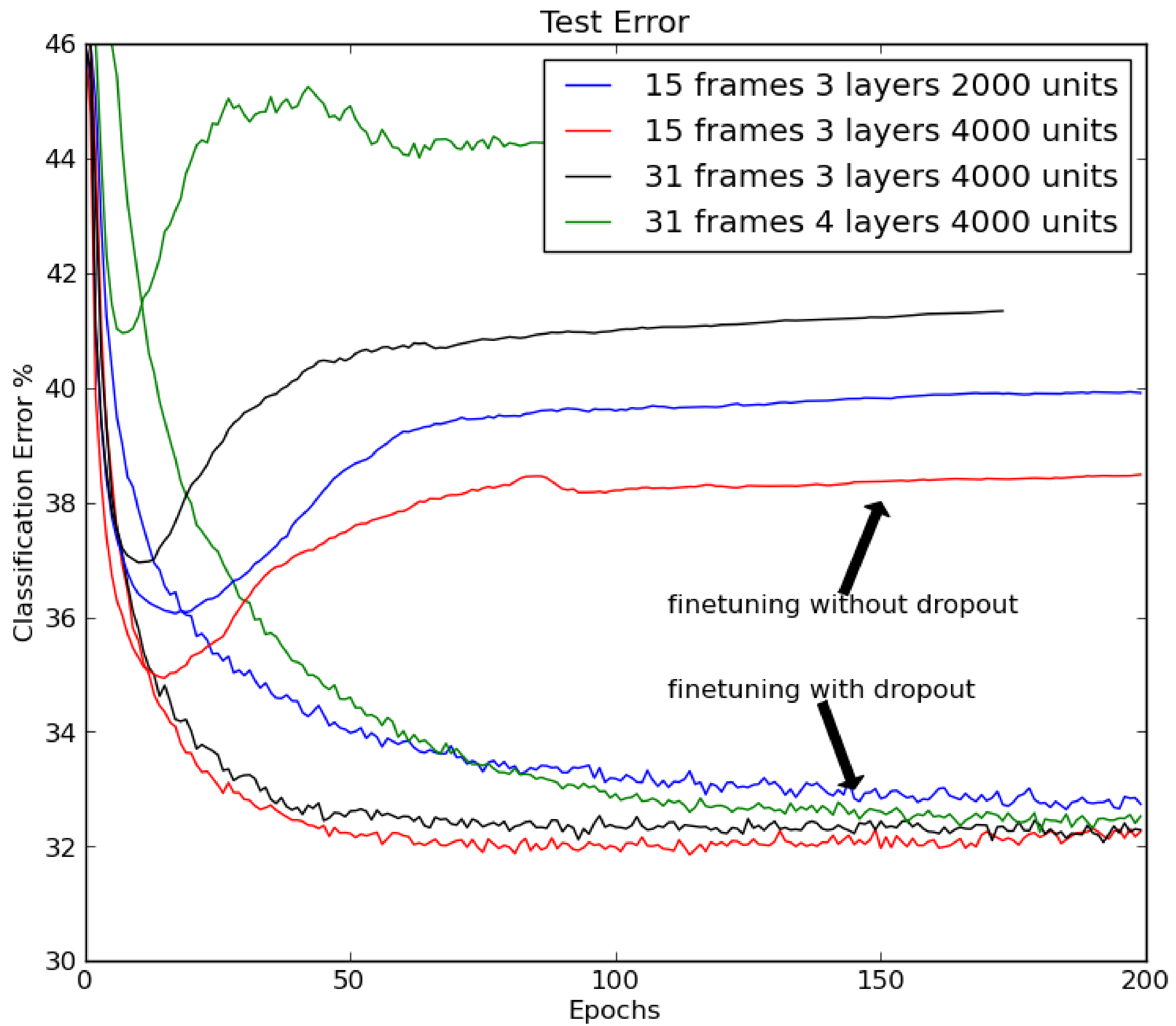
- If your deep neural net is significantly overfitting, it will reduce the number of errors by a lot.

- If your deep neural net is not overfitting you should be using a bigger one.
  - The brain is clearly in the regime where

    # parameters >> # training cases
- Synapses are cheaper than experiences!

Experiments on TIMIT (Nitish Srivastava)

- First pre-train a deep neural network one layer at a time on unlabeled windows of acoustic coefficients.
- Then fine-tune it to discriminate between the classes using a small learning rate.
- Standard fine-tuning: 22.7% error on test set
- Dropout fine-tuning: 19.7% error on test set
  - This was a record for speaker-independent methods.


Test Error
15 frames 3 layers 2000 units
15 frames 3 layers 4000 units
31 frames 3 layers 4000 units
31 frames 4 layers 4000 units
finetuning without dropout
finetuning with dropout
Classification Error %
Epochs
46
44
42
40
38
36
34
32
30
0
50
100
150
200

The ILSVRC-2012 competition on ImageNet

- The dataset has 1.2 million high-resolution training images.
- The classification task:
  - Get the “correct” class in your top 5 bets. There are 1000 classes.
- Some of the best existing computer vision methods were tried on this dataset by leading computer vision groups from Oxford, INRIA, XRCE, …

- Krizhevsky et. al. — 16.4%

Error rates on the ILSVRC-2012 competition

University of Tokyo	26.1%
Oxford University Vision Group	26.9%
INRIA + XRCE	27.0%
University of Amsterdam	29.5%

A better way to think about dropout

- If a hidden unit knows which other hidden units are present, it can co-adapt to them on the training data.
  - But complex co-adaptations are likely to go wrong on new test data.
  - Big, complex conspiracies are not robust.
- If a hidden unit has to work well with combinatorially many sets of co-workers, it is more likely to do something that is individually useful, but also marginally useful given what its co-workers typically achieve.

Comparison with Bayesian approach

- Bayes: Sample lots of separate models from the posterior distribution over parameters.
  - At test time, average the predictions of all these models.

- Dropout: Learn exponentially many models with shared weights.
  - At test time weight all exponentially many models equally.
  - This can be approximated very efficiently.

An alternative to dropout

- In dropout, each neuron computes an activity, p, using the logistic function. Then it sends p to the next layer with a probability of 0.5.
- This has exactly the same expected value as sending 0.5 with probability p.
  - That is exactly what a stochastic binary neuron does (if we call 0.5 one spike)
  - So what happens if we use stochastic binary neurons in the forward pass but do the backward pass as if we had done a "normal" forward pass?

The effect of only sending one bit

- The deep neural network learns slower and gets more errors on the training data.
  - But it generalizes much better.
  - Its almost as big a win as using dropout.

- Dropout variance = $p^2/4$
- Stochastic bit variance = $p(1-p)/4$
  - Stochastic bits have more variance for small p.
  - This is the Poisson limit and resembles neurons

An amusing piece of history

- In 2005 we discovered that deep nets can be pre-trained effectively on unlabeled data by learning a stack of “Restricted Boltzmann Machines”.
- The pre-training uses stochastic binary units. After pre-training we cheat and use backpropagation by pretending that they are deterministic units that send the real-valued outputs of logistics.
  – We would get less overfitting if we stayed with stochastic binary neurons in the forward pass.

Some explanations for why cortical neurons don’t send analog values

- There is no efficient way for them to do it.
  - But some neurons use the precise times of spikes very effectively.
- Evolution just didn’t figure it out.
  - Evolution had hundreds of millions of years. If neurons wanted to send analog values evolution would have found a way.
- Its better to send stochastic spikes because they act as a great regularizer.
  - This helps the brain to use a lot of neurons without overfitting (10^14 parameters,10^9 seconds)

THE END OF THIS PART

See my webpage for our paper:

Improving neural networks by preventing co-adaptation of feature detectors.
Hinton, Srivastava, Krizhevsky, Sutskever & Salakhutdinov