Design Philosophy of Optimization for Deep Learning

Ian Goodfellow
Senior Research Scientist

Stanford, California
2016-03-07
High-Level Lessons

• Strive for *success*, not *perfection*

• Simple optimization methods are successful

• A little model redesign goes farther than a lot of optimization algorithm redesign
Terminology

- Cost function
- Gradient
- Hessian
- Curvature
- Critical points: minima, maxima, saddle points
Derivatives and Second Derivatives

Cost function
\[ J(\theta) \]

Gradient
\[ g = \nabla_{\theta} J(\theta) \]
\[ g_i = \frac{\partial}{\partial \theta_i} J(\theta) \]

Hessian
\[ H \]
\[ H_{i,j} = \frac{\partial}{\partial \theta_j} g_i \]
Critical Points

- All positive eigenvalues
- All negative eigenvalues
- Some positive and some negative
High-Level Lessons

• Strive for *success*, not *perfection*

• Simple optimization methods are successful

• A little model redesign goes farther than a lot of optimization algorithm redesign
Ideally, we would like to arrive at the global minimum, but this might not be possible.

This local minimum performs nearly as well as the global one, so it is an acceptable halting point.

This local minimum performs poorly, and should be avoided.
No Critical Point

---

Classification error rate

---

Deep Learning, Goodfellow, Bengio, and Courville 2016

www.deeplearningbook.org
High-Level Lessons

• Strive for success, not perfection

• Simple optimization methods are successful

• A little model redesign goes farther than a lot of optimization algorithm redesign
The old myth of SGD failure

- SGD usually moves downhill
- SGD eventually encounters a critical point
- Usually this is a minimum
- However, it is a *local minimum*
- The cost function is high at this point
- Some *global minimum* is the real target, and has much lower cost
The new myth of SGD failure

• SGD usually moves downhill

• SGD eventually encounters a critical point

• Usually this is a saddle point

• SGD is stuck, and the main reason it is stuck is that it fails to exploit negative curvature
Gradient descent flees saddle points

\[ \frac{d}{dt} \theta(t) = -g - H (\theta(t) - \theta(0)) \]

\[ \Rightarrow \]

\[ \theta(t) = \theta(0) - Q \Lambda'(t) Q^T g \]

where

\[ \Lambda'(t) = \frac{1 - \exp(-\lambda t)}{\lambda} . \]

Saddle points are a problem…. for Newton’s method, not SGD.

“Qualitatively Characterizing Neural Network Optimization Problems,”
Goodfellow, Vinyals and Saxe, ICLR 2015
(Cartoon of Saxe et al 2013's worldview)
"Qualitatively Characterizing Neural Network Optimization Problems."
Goodfellow, Vinyals and Saxe, ICLR 2015
Qualitatively Characterizing Neural Network Optimization Problems,
Goodfellow, Vinyals and Saxe, ICLR 2015
“Qualitatively Characterizing Neural Network Optimization Problems,”
Goodfellow, Vinyals and Saxe, ICLR 2015
High-Level Lessons

- Strive for *success*, not *perfection*

- Simple optimization methods are successful

- A little model redesign goes farther than a lot of optimization algorithm redesign
Two Extreme Positions

• Convex optimization: Design model within a set of formal constraints, such that efficient and perfect optimization is guaranteed

• Fully general optimization: Write down whatever model seems most intuitive, hope you can optimize it
Modern Deep Nets are very (Piece-Wise) Linear

- Rectified Linear Unit
- Carefully tuned sigmoid
- Maxout
- LSTM (addition is linear)
Batch Normalization

- Consider a very deep net with
  - No nonlinearities
  - Only one unit per layer
- $y = \text{abcdef...x}$
Batch Normalization

\[ Z = XW \]

\[ \tilde{Z} = Z - \frac{1}{m} \sum_{i=1}^{m} Z_i,:) \]

\[ \hat{Z} = \frac{\tilde{Z}}{\sqrt{\epsilon + \frac{1}{m} \sum_{i=1}^{m} \tilde{Z}^2_i,:)}} \]

\[ H = \max\{0, \gamma \hat{Z} + \beta\} \]

Before SGD step

After SGD step

We have presented a novel mechanism for dramatically accelerating the training of deep networks. It is based on the premise that covariate shift, which is known to complicate the training of machine learning systems, also applies to deep networks.

We demonstrate in Fig. 4 that batch normalization allows deep networks with Batch Normalization to reach 72.2% accuracy after 5 times fewer steps than required by Inception. Interestingly, in Sec. 4.2.1, we significantly increase the training speed of the original Inception with sigmoid, but tempted to train the original Inception with sigmoid, but increased by a factor of 5, to 0.0075. The same learning rate increase with original Inception caused the model to train somewhat slower and its batch-normalized variants, vs. the number of training steps. The Figure 3 shows, for each network, the maximum validation accuracy reached by the network and the number of steps to reach it.

Table 2 shows the validation accuracy achieved by the Inception model trained with the initial learning rate of 0.001 and its batch-normalized variants, vs. the number of training steps. The Figure 2 shows the validation accuracy of the Single crop validation accuracy of Inception with Batch Normalization and the maximum validation accuracy achieved by the network.

Net2Net

• Transferring knowledge between neural nets is hard

• Restrict the model architecture to make it easy

Figure 1: Comparison between a traditional workflow and the Net2Net Workflow; Net2Net reuses information from an already trained model to speed up the training of a new model.

More ambitiously, real machine learning systems will eventually become lifelong learning systems (Thrun, 1995; Silver et al., 2013; Mitchell et al., 2015). These machine learning systems need to continue to function for long periods of time and continually experience new training examples as these examples become available. We can think of a lifelong learning system as experiencing a continually growing training set. The optimal model complexity changes as training set size changes over time. Initially, a small model may be preferred, in order to prevent overfitting and to reduce the computational cost of using the model. Later, a large model may be necessary to fully utilize the large dataset.

Net2Net operations allow us to smoothly instantiate a significantly larger model and immediately begin using it in our lifelong learning system, rather than needing to spend weeks or months re-train a larger model from scratch on the latest, largest version of the training set.

2 METHODOLOGY

In this section, we describe our new Net2Net operations and how we applied them on real deep neural nets.

2.1 FEATURE PREDICTION

We briefly experimented with a method that proved not to offer a significant advantage: training a large student network beginning from a random initialization, and introducing a set of extra “teacher prediction” layers into the student network. Specifically, several convolutional hidden layers of the student network were provided as input to new, learned, convolutional layers. The cost function was modified to include terms encouraging the output of these auxiliary layers to be close to a corresponding layer in the teacher network. In other words, the student is trained to use each of its hidden layers to predict the values of the hidden layers in the teacher.

The goal was that the teacher would provide a good internal representation for the task that the student could quickly copy and then begin to refine. The approach resembles the FitNets (Romero et al., 2014) strategy for training very thin networks of moderate depth. Unfortunately, we did not find that this method offered any compelling speedup or other advantage relative to the baseline approach. This may be because our baseline was very strong, based on training with batch normalization (Ioffe & Szegedy, 2015). Mahayri et al. (2015) independently observed that the benefits of the FitNets training strategy were eliminated after changing the model to use batch normalization.

The FitNets-style approach to Net2Net learning is very general, in the sense that, if successful, it would allow any architecture of student network to learn from any architecture of teacher network. Though we were not able to make this general approach work, we encourage other researchers to attempt to design fully general Net2Net strategies in the future. We instead turned to different Net2Net strategies that were limited in scope but more effective.

2.2 FUNCTION-PRESERVING INITIALIZATIONS

We introduce two effective Net2Net strategies. Both are based on initializing the student network to represent the same function as the teacher, then continuing to train the student network by normal means. Specifically, suppose that a teacher network is represented by a function $y = f(x; \theta)$ where...
would not implement the same function as the old unit vector for pre-existing unit operation. The random remapping for the multiplication parameters must match the random by learned parameters that allow the layer to represent any range of outputs despite the normalization layer involves both a standard linear transformation, but also involves elementwise multiplication. One example is the layer structure used by batch normalization (Ioffe & Szegedy, 2015). To explain, we provide examples of two computational graph structures that impose specific constraints as defined by the computation graph. Care needs to be taken to ensure that the remapping functions do in fact result in function preservation. Importantly, these by Fig. 2. So far we have only discussed the use of a single random mapping function to expand described by an arbitrary directed acyclic computation graph. This general procedure is illustrated. This description can be generalized to making multiple layers wider, with the layers composed as by Fig. 2. Under review as a conference paper at ICLR 2016

![Diagram](image-url)

is hard to do general factorization of layers which non-linear transformation units, such as rectified
network that factorizes the original layer. Making a deeper but equivalent representation. However it
The approach we take is a specific case of a more general approach, that is to build a multiple layer
network on training data in order to estimate the activation statistics.
the normalization of the layer's statistics. This requires running forward propagation through the
normalization, we must set the output scale and output bias of the normalization layer to undo
In some cases, to build an identity layer requires additional work. For example, when using batch

It is essential that each unit be replicated at least once, hence the constraint that the resulting layer
initialized to an identity matrix, but remains free to learn to take on any value later. This operation

We also introduce a second function-preserving transformation, 

Another example is concatenation. If we concatenate the output of layer 1 and layer 2, then pass this

**Figure 3: The Net2DeeperNet Transformation**

Under review as a conference paper at ICLR 2016

Under review as a conference paper at ICLR 2016

(a) Training Accuracy of Different Methods
(b) Validation Accuracy of Different Methods

Figure 5: Comparison of methods of training a deeper model

inception model for reference, which should be easier to train than these larger models. We can find that the models initialized with \textit{Net2Net} operations converge even faster than the standard model. This example really demonstrate the advantage of \textit{Net2Net} approach which helps us to explore the design space faster and advance the results in deep learning.

\section{Discussion}

Our \textit{Net2Net} operators have demonstrated that it is possible to rapidly transfer knowledge from a small neural network to a significantly larger neural network under some architectural constraints. We have demonstrated that we can train larger neural networks to improve performance on ImageNet recognition using this approach. \textit{Net2Net} may also now be used as a technique for exploring model families more rapidly, reducing the amount of time needed for typical machine learning workflows. We hope that future research will uncover new ways of transferring knowledge between neural networks. In particular, we hope future research will reveal more general knowledge transfer methods.

Residual Nets

- Similar to much older skip connections strategies
- Add much shorter paths from input to output while retaining depth
- Multi-step program initialized to sequence of no-ops

He et al, 2015
High-Level Lessons

• Strive for *success*, not *perfection*

• Simple optimization methods are successful

• A little model redesign goes farther than a lot of optimization algorithm redesign
Deep Learning
Goodfellow, Bengio, and Courville

In preparation for MIT Press