Gated Convolutional Layers and Joint K-Means Weight Initialization

Brady Zhou
University of Texas at Austin
brady.zhou@utexas.edu

Abstract

Deep neural networks are highly convex, nonlinear functions and optimization theory for them has many more questions than answers so far. Despite this, many techniques have been develop to ease the training process and enable training of deeper networks, where initialization is more crucial. These techniques consist of batch normalization and smarter weight initialization, as well as different nonlinearities. In this work, we focus on convolutional layers with ReLU nonlinearities and present an extension on data dependent weight initialization that initializes the kernels to align with clusters of input patches in a joint layer fashion. This leads to the weights capturing the modes of the data such that the optimization process is expedited, which we demonstrate with experiments on CIFAR-10 and comparisons with other popular initialization choices. Code is available at http://www.github.com/bradyz/nn_weights.

1. Introduction

In designing a deep network, there are several considerations to take into account. Many factors such as depth, width, optimizers have significant effect on the training and testing performance, leading to a large amount of time "tuning", or running hyperparameter search. In this work, our contributions are the following

- Convolutional layers with increased flexibility and expressive power
- Joint-layer, data dependent weight initialization
- Weight regularization that encourages filter diversity

We focus on running experiments comparing our initialization method versus more standard initialization methods, and leave comparisons against other unsupervised methods as future work.

2. Related Work

Weight Initalization For all nearly all deep learning models, the parameters are learned through some first order descent method such as stochastic gradient descent (SGD) or more recently, ADAM [6]. With these optimization techniques all converging to local minimum, it is crucial that the network is initialized well. The most basic way to initialize the weights is to sample from $\mathcal{N}(0, \sigma)$, where σ is some small constant like 10^{-3}. In deeper networks, the initialization is even more crucial. X. Glorot and Y. Bengio [2] present a commonly used "Xavier initialization", which is simply scales σ by the number of inputs and outputs, that is, $\sigma = \frac{2}{n_{in}+n_{out}}$. Under the assumption that the inputs are unit gaussian, Xavier initialization makes it such that the outputs are close to unit gaussian. Similarly, [4] introduces "He initialization", which provides analysis for networks using the ReLU nonlinearity, and set $\sigma = \frac{2}{n_{in}}$, to increase the variance since an expected half of all units will be set to zero from the ReLU layer.

Supervised Pre-training A common tactic employed by deep learning practitioners is to first pre-train on another larger dataset with class labels, such as Imagenet [1], and then "fine-tune" on the original dataset by reducing the learning rate and retraining upper layers. The idea behind this is that the network learns a good representation for images, thus making training easier given less data.

Unsupervised Pre-training In many tasks, labels are very typically expensive to acquire, but there is an abundance of images available. To exploit this, M. Noroozi *et al.* [8] construct an unsupervised learning task which consists of reconstructing a scrambled version of the image. Using autoencoders to minimize image reconstruction loss can also be used to learn a good feature representation. In [9], D. Pathik *et al.* use image inpainting to pretrain their networks. The methods discussed so far consist mostly of "semi-supervised" methods, contrary to fully unsupervised methods [7]. In their method, they apply spherical K-means to the inputs patches of every layer and extract c clusters, where c corresponds to the number of filters. This way, the convolutional filters are initialized such that the modes of the activations are captured. Our work is heavily motivated by this, with the additional observation that networks are trained jointly, and so the clustering should be learned jointly, as opposed to layer-by-layer as in their work.

Weight Regularization Weight decay is a standard technique used almost universally in training deep nets. Along with the normal loss function, there is an additional loss added that penalizes the norm of the weights. Concretely, the loss function is $l_{reg}(\theta) = \frac{\lambda}{2}||\theta||^2$, where θ denotes the model parameters. We can see that this all this loss function does is induces weights with smaller norms. This technique is simple, fast to compute, and extremely effective. This method has adopted by nearly all present day training.

3. Approach

3.1. Gated Convolutional Layers

The intuition behind this was the observation that even though ConvNets are highly nonlinear, there is still only one "pathway" information can flow in a feedforward network, that is, all the kernels are being used on every pass. The original convolutional layer is of the form

$$y = x \circledast w \tag{1}$$

In this work, we experiment with a gated convolution that allows dynamic use of convolutional kernels, allowing for different kernels to be used, based on the input activation. Note that biases and striding is omitted for clarity.

$$\begin{aligned} \psi &= \sigma(x \circledast w_1) \\ \phi_1 &= x \circledast w_2 \\ \phi_2 &= x \circledast w_3 \\ y &= \psi \odot \phi_1 + (1 - \psi) \odot \phi_2 \end{aligned} \tag{2}$$

where $\odot$ denotes the Hadamard product, or element-wise multiplication. Here, $x \in \mathbb{R}^{h \times w \times c}$, $y \in \mathbb{R}^{h \times w \times c'}$, and $w_i \in \mathbb{R}^{k \times k \times c \times c'}$. The intuition behind this layer is that ψ acts as a simple nonlinear classifier on the input activation, and determines which kernel to use.

For a vanilla convolutional layer with kernel size k, and number of input channels c and number of output channels c', the number of parameters required is $k \times k \times c \times c'$. For our gated layer, we can see that the number of parameters for the same settings is $k \times k \times 3c \times c'$. Even though the number of parameters grows linearly, the number of "pathways" the information can go through growths exponentially.

3.2. Data Dependent K-Means Weight Initialization

Given a set of input images without training samples $\{x_1, x_2, \ldots, x_n\}$, and layer weights $\{\theta_1, \theta_2, \ldots, \theta_m\}$, where m is the number of layers, we seek to initialize the weights to maximally align with clusters of input patches for each layer.

For a single layer, parameterized by k weights $\{\theta_1, \theta_2, \ldots, \theta_k\}$, we can express the spherical K-means objective function as follows

$$\max_{\theta} \sum_{i=1}^{n} \max_{k} x_i^\intercal \frac{\theta_k}{||\theta_k||} \tag{3}$$

Note that we put a unit norm constraint on each weight to prevent the weights from growing to infinity. This objective function is piecewise differentiable, with discontinuities when the maximum dot product between two clusters of input patches is the same. Despite this, it is easily trained with gradient descent and has convergence guarantees. This is trivially extended to support to multiple layers, by simply summing the same objective function over all the layers. One subtlety between this and the original algorithm in [7] is that our approach could potentially, and does in fact align multiple weights with a single input cluster. This is unfavorable, as this leads to duplicate weights and is addressed by an extra regularization term that penalizes weights that are aligned with each other.

In our experiments, we use images from the test set to initialize our weights, and show the results from using 256 images. We perform gradient steps minimizing the objective function and terminate after a fixed number of iterations (10000), which takes approximately 5 minutes. This initialization time is for the most part negligible to the full training time, approximately 1 hour for CIFAR-10 on the network we use. Future extensions include initialization using the entire training/test set, and better stopping criterion.

3.3. Cosine Weight Regularization

To address the problem of weights collapsing onto the same mode in the initialization before, we apply a penalty on just this. We measure the cosine similarity between each filter within a single layer and try to minimize this. This is equivalent to maximizing pairwise orthogonality. We formulate this as a layer-wise optimization, as follows

$$\min_{\theta_1, \theta_2, \ldots, \theta_n} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Big(\frac{\theta_i}{||\theta_i||}^\intercal \frac{\theta_j}{||\theta_j||} \Big)^2 \tag{4}$$

Now, the full objective function for data dependent K-means is

$$l(\theta) = l_{kmeans}(\theta) + \lambda l_{cos}(\theta) \tag{5}$$

where λ is determined empirically. In section 4 we show results from different levels of λ.

To incorporate this regularization into our loss function, we have to take the gradients with respect to θ_k. This is trivial to compute and is of the form

$$\frac{\partial}{\partial \theta_k} l_{cos}(\theta) = \sum_{i=1}^{n} \Big(\frac{\theta_i}{||\theta_i||}^\intercal \frac{\theta_k}{||\theta_k||} \Big) \Big(I - \frac{\theta_k \theta_k^\intercal}{||\theta_k||^2} \Big) \frac{\theta_i}{||\theta_i|| ||\theta_k||} \tag{6}$$

This cosine regularization loss would be near to useless on its own - as two random normal vectors are already orthogonal with higher probability as the number of dimensions increases, but is necessary for our approach to "push" weights to different input patch clusters.

4. Results

To demonstrate the effectiveness of the methods presented, we run several experiments on the CIFAR-10 [7] dataset for classification accuracy. For all experiments, we use the following network architecture shown in Table 1.

Layer	Filter Size	Stride	Output Size
Conv1_1	11	1	$32 \times 32 \times 16$
Conv1_2	11	2	$16 \times 16 \times 16$
Conv2_1	5	1	$16 \times 16 \times 32$
Conv2_2	5	2	$8 \times 8 \times 32$
Conv3_1	3	1	$8 \times 8 \times 64$
Conv3_2	3	1	$8 \times 8 \times 64$
Conv4_1	3	1	$8 \times 8 \times 128$
Conv4_2	3	1	$8 \times 8 \times 128$
Global Avg Pool	-	-	128
Fc_5	-	-	128
Fc_6	-	-	128
Logits	-	-	10

Table 1: Network Architecture

Note that each layer besides the pooling and final layer is followed by batch normalization [5] and a ReLU nonlinearity . The filter size for the first layer has a large receptive field so that we can visualize the weights to see what the initialization has learned. Additionally, this network is rather shallow compared to the current state of the art architectures, but still provides some insight.

We compare our method using some popular choices for weight initialization - $\mathcal{N}(0, 10^{-3})$, Xavier [2], and He initialization [4], and optimize the network using SGD with momentum and ADAM [6]. For all experiments, we train with batch size 128, with a learning rate starting at 10^{-3} for 50000 iterations, which is then halved and run for 25000 more iterations. Data augmentation consists only of only horizontal flipping, and preprocessing consists of zero centering and scaling the image to the range $[-1, 1]$. Additionally, we use standard weight decay with a factor of 10^{-5}.

4.1. Training Speed

One reason why fine tuning ImageNet [1] pretrained models has seen so much success is its ability to capture general low and mid level features that generalize over all natural images. Our method exhibits a similar behavior, as the convolutional kernels are initialized to recurring image patches from the dataset. We claim the convolutional layer's parameters are near local minimum and the fully connected layers require the most learning, which leads to the expedited training as shown in Figure 1

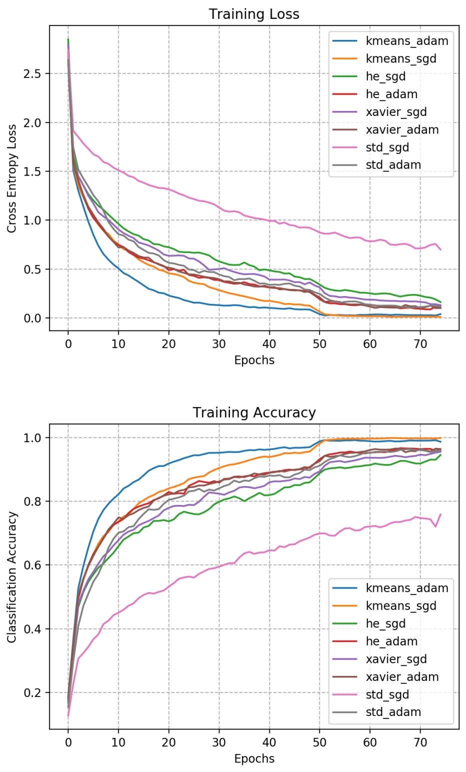

Figure 1: Cross entropy loss and classification accuracy during training. In these trials, cosine regularization was used with $\lambda = 100$.

One intriguing property of training when initialized with our method is that even Ours-SGD outperforms all other initializations, even with ADAM, which has shown to have increased convergence times.

4.2. Classification

We run experiments on CIFAR-10, which consists of 50000 training images and 10000 test images. The images are colored and of size 32×32. We present results using classification error and loss using varying parameters of our method.

Note that our method outperforms all others using the SGD optimizer, but only marginally. Using ADAM, all the methods perform rather similarly, but our method suffers from heavy overfitting, as shown in the appendix. This behavior is probably caused from lack of extensive data augmentation and other regularization techniques such as dropout.

Initialization	Optimizer	Test Acc.	Test Loss
$\mathcal{N}(0, 10^{-3})$	SGD	0.697	0.876
Xavier	SGD	0.751	0.772
He	SGD	0.740	0.824
K-Means	SGD	**0.767**	**0.763**
$\mathcal{N}(0, 10^{-3})$	ADAM	0.801	**0.638**
Xavier	ADAM	0.806	0.663
He	ADAM	**0.811**	0.652
K-Means	ADAM	0.803	0.778

To test out the gated convolution, we run experiments on CIFAR-10 and CIFAR-100, with the two of the same network architecture from 1. The vanilla network has all convolutional output sizes multiplied by 3, to make the number of parameters the same.

Method	Dataset	Test Acc.	Test Loss
Vanilla	CIFAR-10	0.885	0.403
Vanilla	CIFAR-100	0.605	1.685
Ours	CIFAR-10	0.882	0.397
Ours	CIFAR-100	0.608	1.640

This experiment is inconclusive and does not show any significant increase or decrease of expressive power as hypothesized. There are two ways to interpret these results. In favor of normal convolutional layers - now a single output activation map has outputs from different weights could provide a noisy signal for the later layers. Also, the branch that decides which kernel to use is a single linear classifier, which is not powerful enough to decide which branch to output. In favor of our gated layers - we can achieve the same accuracy using only a third of the number of output channels, signaling that normal convolutional layers are lower rank, and our layers can encode the same amount of information with less space.

4.3. Filter Visualization

The goal of our K-means initialization is to be able to learn good features before the supervised training starts. To show this, we can first visualize the first layer's weights from a fully converged network Xavier-Adam after 150000 gradient steps of training.

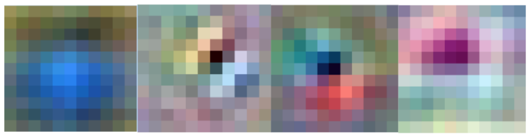

Figure 2: Visualization of Conv1_1 kernels from a fully converged model.

We then run our weight initialization without cosine regularization and after convergence (10000 gradient steps of batch size 64), we achieve the following filters.

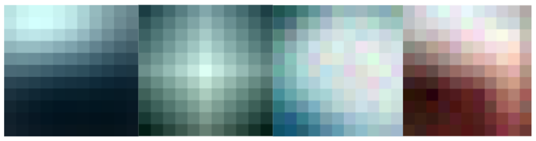

Figure 3: Visualization of Conv1_1 kernels from a network initialized with our method. Note that no labels have been used.

This result confirms our hypothesis that meaningful features can in fact be learned from the input data alone and can explain the expedited training process.

5. Conclusion

In this section we describe our work at a high level and in honest terms (TLDR). This final project was used as a medium to explore a few of the ideas in a large notepad of ideas untested. The first idea presented, gated convolutional layers, was a small idea thought of a while ago when brainstorming about dynamic convolutional layers, inspired by video interpolation [3], where the output of the network is a kernel itself, which is then convolved with the inputs. The results from using our layer do not show any improvements over a standard convolutional layer, but this could be due to the size of the dataset, as well as the network size. Since we divide the number of output layers by 3 to offset the parameter usage, we end up having less features for the fully connected layer. This bottleneck is perhaps the main reason why there are no significant gains.

Next, the data dependent initialization is a really interesting topic as initialization in general is for the most part very unsolved, and can lead to extremely different results depending on input distributions and initialization method. It also makes sense that there is a set of "good" features to learn that can be learned from the data alone that supervision and training can benefit from. Unfortunately, evaluation for this method was not as thorough as possible, only being tested on CIFAR-10. The evaluation section was kept as basic as possible to fairly compare all methods on a standard dataset, but from the expedited learning and the unsupervised nature, this method would probably shine on more difficult datasets, which remained untested. Additionally, the cosine regularization loss is essentially a hack to prevent the kernels from collapsing onto the same input modes. We can see this loss is linear in the angle, and should be compared to a loss that is inversely exponential in the angle. That is, a loss that only has a large penalty if two vectors are near aligned, and a very small penalty otherwise.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[2] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[3] Z. Gong and Z. Yang. Video frame interpolation and extrapolation.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv preprint arXiv:1511.06856*, 2015.

[8] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016.

[9] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.