DSD: Regularizing Deep Neural Networks with Dense-Sparse-Dense Training Flow

Song Han
Stanford University
songhan@stanford.edu

Jeff Pool
NVIDIA
jpool@nvidia.com

Sharan Narang
Baidu Research
sharan@baidu.com

Huizi Mao
Tsinghua University
maohz12@mails.tsinghua.edu.cn

Shijian Tang
Stanford University
sjtang@stanford.edu

Erich Elsen
Baidu Research
erichelsen@baidu.com

Bryan Catanzaro
Baidu Research
bcatanzaro@baidu.com

John Tran
NVIDIA
johntran@nvidia.com

William J. Dally
Stanford University
NVIDIA
dally@stanford.edu

Abstract

Modern deep neural networks have a large number of parameters, making them very powerful machine learning systems. A critical issue for training such large networks on large-scale data-sets is to prevent overfitting while at the same time providing enough model capacity. We propose DSD, a dense-sparse-dense training flow, for regularizing deep neural networks. In the first D step, we train a dense network to learn which connections are important. In the S step, we regularize the network by pruning the unimportant connections and retrain the network given the sparsity constraint. In the final D step, we increase the model capacity by freeing the sparsity constraint, re-initializing the pruned parameters, and retraining the whole dense network. Experiments show that DSD training can improve the performance of a wide range of CNN, RNN and LSTMs on the tasks of image classification, caption generation and speech recognition. On the Imagenet dataset, DSD improved the absolute accuracy of AlexNet, GoogleNet, VGG-16, ResNet-50, ResNet-152 and SqueezeNet by a geo-mean of 2.1 points (Top-1) and 1.4 points (Top-5). On the WSJ'92 and WSJ'93 dataset, DSD improved DeepSpeech-2 WER by 0.53 and 1.08 points. On the Flickr-8K dataset, DSD improved the NeuralTalk BLEU score by 2.0 points. DSD training flow produces the same model architecture and doesn't incur any inference overhead.

1 Introduction

Deep neural networks (DNNs) have shown significant improvements in many application domains, ranging from computer vision [1] to natural language processing [2] and speech recognition [3]. The abundance of more powerful hardware [4] makes it easier to train complicated DNN models with large capacities. The upside of complicated models is that they are very expressive and can capture the highly non-linear relationship between features and output. The downside of such large models is that they are prone to capturing the noise, rather than the intended pattern, in the training dataset. This noise does not generalize to new datasets, leading to over-fitting and a high variance.

In contrast, simply reducing the model capacity would lead to the other extreme, causing a machine learning system to miss the relevant relations between features and target outputs, leading to under-fitting and a high bias. Bias and variance are hard to optimize at the same time.

```
Algorithm 1: Workflow of DSD training
Initialization: $W^{(0)}$ with $W^{(0)} \sim N(0, \Sigma)$
Output: $W^{(t)}$.
------------------------------ Initial Dense Phase ------------------------------
while not converged do
    $\tilde{W}^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)})$;
    $t = t + 1$;
end
------------------------------ Sparse Phase ------------------------------
$S = sort(|W^{(t-1)}|);\ \lambda = S_{k_i};\ Mask = \mathbb{1}(|W^{(t-1)}| > \lambda)$;
while not converged do
    $\tilde{W}^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)})$;
    $\tilde{W}^{(t)} = W^{(t)} \cdot Mask$;
    $t = t + 1$;
end
------------------------------ Final Dense Phase ------------------------------
while not converged do
    $\tilde{W}^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)})$;
    $t = t + 1$;
end
goto Sparse Phase for iterative DSD;
```

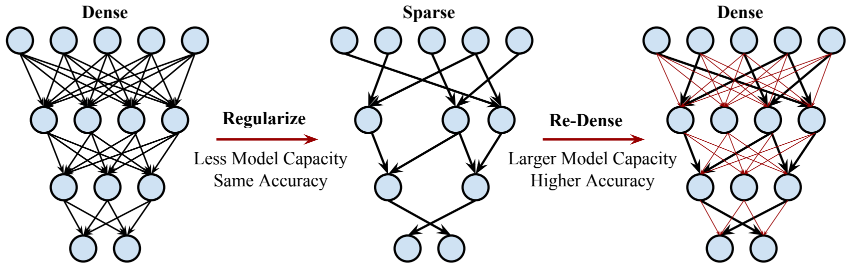


Figure 1: Dense-Sparse-Dense Training Flow. The sparse training regularizes the network and prevents overfitting, while the second dense training increases the model capacity and decreases bias, leading to higher prediction accuracy.

To solve this problem, we propose dense-sparse-dense training flow (DSD), a novel training method that first regularizes the model through sparsity-constrained optimization, and then increases the model capacity by recovering and retraining on pruned weights. At test time, the final model produced by DSD training still has the same architecture and dimension as the original dense model, and DSD training doesn't incur any inference overhead. We experimented with DSD training on 9 mainstream CNN / RNN / LSTMs for image classification, image caption and speech recognition, and found substantial performance improvements.

2 DSD Training Flow

Our DSD training employs a three-step process: dense, sparse, dense. Each step is illustrated in Figure 1 and Algorithm 1. The progression of weight distribution is plotted in Figure 2.

Initial Dense Training: The first D step learns the connectivity via normal network training on the dense network. Unlike conventional training, however, the goal of this D step is not to learn the final values of the weights; rather, we are learning which connections are important.

Sparse Training: The S step prunes the low-weight connections and retrains the sparse network. All connections with weights below a threshold are removed from the network, converting a dense network into a sparse network. This truncation-based procedure has provable advantage in statistical accuracy in comparison with their non-truncated counterparts [5, 6, 7]. We use sensitivity-based analysis [8, 9] to find a separate threshold for each layer. This threshold is denoted as λ in Algorithm 1. Then we retrain the sparse network, which can fully recover the model accuracy under the sparsity constraint [8]. The S step adds the sparsity constraint as a strong regularization to prevent over-fitting.

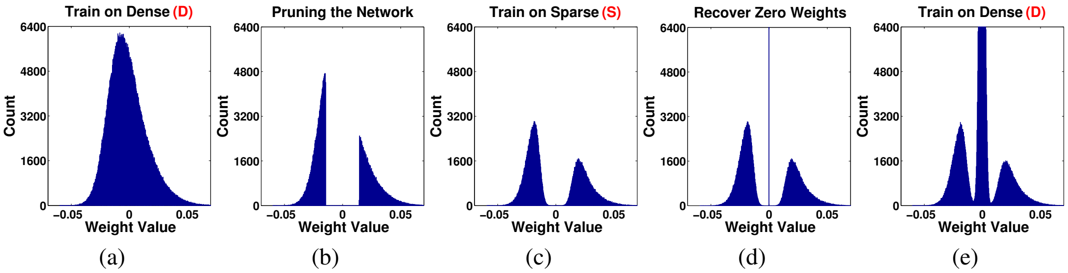


Figure 2: Weight distribution of the original GoogleNet (a), pruned GoogleNet (b), retrain the sparsity-constrained GoogleNet (c), get rid of the sparisty constraint and recover the zero weights (d), retrain the dense network (e).

Final Dense Training The final D step recovers the pruned connections, making the network dense again. These previously-pruned connections are initialized to zero and retrained with 1/10 the original learning rate (since the sparse network is already at a good local minima). Dropout ratios and weight decay remained unchanged. Restoring the pruned connections increases the dimensionality of the network, and more parameters make it easier for the network to slide down the saddle point to arrive at a better local minima. This step adds model capacity and lets the model have less bias.

To visualize the DSD training flow, we plotted the progression of weight distribution in Figure 2. The figure is plotted using GoogleNet inception_5b3x3 layer, and we found that this progression of weight distribution is very representative for AlexNet, VGGNet, ResNet and SqueezeNet as well.

The original distribution of weights is centered on zero with tails dropping off quickly. Pruning is based on absolute value so after pruning the large center region is truncated away. The network parameters adjust themselves during the retraining phase, so in (c) the boundary becomes soft and forms a bimodal distribution. In (d), all the pruned weights comes back again and reinitialized to zero. Finally, in (e), the previously-pruned weights are retrained in the final dense training step. In this step, comparing Figure (d) and (e), the old weights' distribution almost remained the same, while the new weights become more spread around zero,

3 Experiments

We applied DSD training to different kinds of neural networks on data-sets from different domains. We found that DSD training improved the accuracy for *all* these networks compared to neural networks that were not trained with DSD. The neural networks are chosen from CNN, RNN and LSTMs; The data sets are chosen from image classification, speech recognition, and caption generation. An overview of the networks and dataset we used are shown in Table 1.

Table 1: Overview of the deep neural networks used to experiment DSD training

Neural Network	Domain	Dataset	#Parameters	#Layers	Type
AlexNet	Vision	ImageNet	60 Million	8	CNN
VGG-16	Vision	ImageNet	138 Million	16	CNN
GoogleNet	Vision	ImageNet	13 Million	64	CNN
ResNet-50	Vision	ImageNet	25 Million	54	CNN
ResNet-152	Vision	ImageNet	60 Million	156	CNN
SqueezeNet	Vision	ImageNet	1.2 Million	26	CNN
DeepSpeech	Speech	WSJ	8 Million	5	RNN
DeepSpeech-2	Speech	Baidu internal	67 Million	10	RNN
NeuralTalk	Language	Flickr-8K	6.8 Million	4	RNN+LSTM

3.1 AlexNet

We experimented with BVLC AlexNet obtained from Caffe Model Zoo [10]. It has 61 million parameters across 5 convolutional layers and 3 fully connected layers. We pruned the network to be 89% sparse (11% non-zeros), the same as in previous work [8]. Retraining the sparse network fully recovers the original accuracy, as shown in Table 2. After re-dense training, AlexNet obtained absolute improvements of 1.4% (Top-1) and 1.0% (Top-5) over the baseline.

Table 2: DSD results on AlexNet

AlexNet	Top-1 Err	Top-5 Err	Sparsity
Baseline	42.8%	19.7%	0%
Sparse	42.7%	19.7%	89%
DSD	**41.4%**	**18.7%**	0%
Improvement (abs)	1.4%	1.0%	-
Improvement (rel)	**3.3%**	**5.1%**	-

3.2 GoogleNet

We experimented with the BVLC GoogleNet [11] model obtained from the Caffe Model Zoo [10]. It has 13 million parameters and 57 Conv layers. Following the sensitivity analysis methodology, we pruned the network to be 64% sparse. Retraining the sparse network gave a small improvement in accuracy due to regularization, as shown in Table 3. After the final dense training step, Googlenet obtained absolute improvements of 1.4% (Top-1) and 0.8% (Top-1) over the baseline.

Table 3: DSD results on GoogleNet

GoogleNet	Top-1 Err	Top-5 Err	Sparsity
Baseline	31.3%	11.1%	0%
Sparse	31.2%	10.9%	64%
DSD	**29.9%**	**10.2%**	0%
Improvement (abs)	1.4%	0.9%	-
Improvement (rel)	**4.5%**	**8.1%**	-

3.3 VGGNet

We also explored DSD training on VGG-16[12] model obtained from Caffe Model Zoo [10]. It has 138 million parameters with 13 convolutional layers and 3 fully-connected layers. We tried two pruning approaches: aggressive pruning for compression, using the methodology of past work [8] to prune to 92% sparsity, and pruning more gently using sensitivity analysis to only 35% sparsity. With aggressive pruning, two DSD iterations reduced the error by 2.8% (Top-1) and 1.7% (Top-5). However, on a gently-pruned network, after only one DSD iteration we reduced the error of the network by 4.2% (Top-1) and 2.6% (Top-5), detailed in Table 4.

Table 4: DSD results on VGG-16

VGG-16	Top-1 Err	Top-5 Err	Sparsity
Baseline	31.5%	11.3%	0%
Sparse	29.1%	9.7%	35%
DSD	**27.3%**	**8.7%**	0%
Improvement (abs)	4.2%	2.6%	-
Improvement (rel)	**13.4%**	**23.3%**	-

3.4 ResNet

Deep Residual Networks [1] (ResNets) were the top performer in the 2015 ImageNet ILSVRC [13] classification challenge, including novel features including shortcut connections, a deeper network, and batch normalization [14]. We applied DSD training to this new network architecture.

We examined pre-trained ResNet-50 and ResNet-152 provided by the original authors. Despite their large number of layers (54 and 156), these networks only have 25 and 60 million parameters, respectively. Without data augmentation, the error of the pre-trained caffemodels on our standard ImageNet data set is 26.75%/8.76% (ResNet-50) and 25.0%/7.7% (ResNet-152). We used sensitivity-based analysis to prune all layers (except BN layers) to an overall sparsity of 52%/55%. A single DSD pass for these networks reduced error by 1.60%/1.01% (ResNet-50) and 1.5%/1.0% (ResNet-152), shown in Table 5 and Table 6.

We performed a second DSD iteration on ResNet-50, pruning the network to be 52% sparse, decreasing the learning rate by another factor of 10x, and keeping other factors unchanged (dropout, batch normalization, L2 regularization). Retraining this pruned network, re-densifying again, and a final re-training decreased the Top-1 and Top-5 errors to 24.97% and 7.60%. Each stage of this process can be seen in Table 5.

Finally, we compared the effect of gentle vs. aggressive pruning of DSD training on ResNet-50, with 52% and 67% sparsity. Though we could more than regain the accuracy lost from aggressive pruning, it achieved 1.0% worse accuracy compared with less-aggressive pruning. Gentle pruning gave better final accuracy results on ResNet-50, as it did for VGG16.

Table 5: DSD results on ResNet-50

ResNet-50	Top-1 Err	Top-5 Err	Sparsity
Baseline	26.75%	8.76%	0%
Sparse-Iter1	26.34%	8.28%	52%
DSD-Iter1	25.15%	7.75%	0%
Sparse-Iter2	25.00%	7.64%	52%
DSD-Iter2	**24.97%**	**7.60%**	0%
Improvement (abs)	1.78%	1.16%	-
Improvement (rel)	**6.65%**	**13.24%**	-

Table 6: DSD results on ResNet-152

ResNet-152	Top-1 Err	Top-5 Err	Sparsity
Baseline	25.0%	7.7%	0%
Sparse-Iter1	24.4%	7.2%	55%
DSD-Iter1	**23.5%**	**6.7%**	0%
Improvement (abs)	1.5%	1.0%	-
Improvement (rel)	**5.9%**	**13.3%**	-

3.5 SqueezeNet

SqueezeNet [9] is a very small network targeting mobile applications. It is a fully-convolutional network without any FC layer. The model takes only 4.8MB, having 50x fewer parameters but the same accuracy as AlexNet on ImageNet. We applied our approach to this network to see how DSD training behaves on such super compact network.

We obtained the baseline model from the Caffe Model Zoo [10]. Even though SqueezeNet is very compact, we were able to prune away 67% of the parameters while improving accuracy. After the final dense training, we observed that the ImageNet accuracy improved by 4.3% (Top-1) and 3.2% (Top-5), which is a greater improvement than for the other networks. We suspect that this compact model is very sensitive to the difference in model capacity between sparse and dense and thus benefits more by eliminating under-fitting.

Table 7: DSD results on SqueezeNet

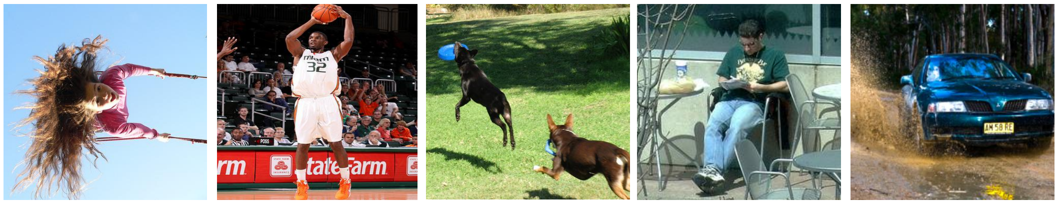
SqueezeNet	Top-1 Err	Top-5 Err	Sparsity
Baseline	42.5%	19.7%	0%
Sparse	41.3%	18.7%	67%
DSD	**38.2%**	**16.5%**	0%
Improvement (abs)	4.3%	3.2%	-
Improvement (rel)	**10.1%**	**16.2%**	-

3.6 NeuralTalk

Having examined the effectiveness of DSD training on CNN, we evaluated DSD training on RNN and LSTM. We applied DSD to NeuralTalk [15], an LSTM for generating image descriptions. It uses a CNN as an image feature extractor, and an RNN with LSTM cell to generate captions. To verify DSD training on the RNN and LSTM, we fixed the CNN weights and only train the RNN and LSTM weights. The baseline NeuralTalk model we used is the flickr8k_cnn_lstm_v1.p downloaded from NeuralTalk Model Zoo. It has 6.8 million parameters with the following dimensions: We:4096x512, WLSTM:1025x2048, Wd:512x2538 and Ws:2538x512. It achieved BLEU1-4 score of [57.2, 38.6, 25.4, 16.8], which matches with the results published in the original paper [15].

In the pruning step, we pruned all layers except Ws, the word embedding lookup table, to 80% of its original size. We retrained the remaining sparse network using similar hyper parameters as the original paper: learning rate at 1e-3, decay_rate at 0.997, batch size 128. Retraining the sparse network improved the BLUE score by [1.2, 1.1, 0.9, 0.7]. Getting rid of sparsity constraint and retraining the dense network further improved BLEU score by [2.0, 2.1, 0.9, 1.7] compared with the baseline model.

Since BLEU score is not the sole criteria measuring auto-caption system, we visualized the captions generated by DSD showing that DSD training improves the caption performance. In Figure 3, the baseline model fails to describe image 1,4,5. For example, in the first image, the baseline model mistakes the girl with a boy, and mistakes the girl's hair with rock; the sparse model can tell that's a girl, and the DSD model can further identify the swing. In the the second image, DSD training



✗ **Baseline**: a boy in a red shirt is climbing a rock wall.

✗ **Sparse**: a young girl is jumping off a tree.

✓ **DSD**: a young girl in a pink shirt is swinging on a swing.

○ **Baseline**: a basketball player in a red uniform is playing with a ball.

○ **Sparse**: a basketball player in a blue uniform is jumping over the goal.

✓ **DSD**: a basketball player in a white uniform is trying to make a shot.

✓ **Baseline**: two dogs are playing together in a field.

✓ **Sparse**: two dogs are playing in a field.

✓ **DSD**: two dogs are playing in the grass.

✗ **Baseline**: a man and a woman are sitting on a bench.

○ **Sparse**: a man is sitting on a bench with his hands in the air.

○ **DSD**: a man is sitting on a bench with his arms folded.

✗ **Baseline**: a person in a red jacket is riding a bike through the woods.

✓ **Sparse**: a car drives through a mud puddle.

✓ **DSD**: a car drives through a forest.

Figure 3: Visualization of DSD training improves the performance of image captioning.

can tell the player is trying to make a shot, rather than the baseline just saying he's playing with a ball. It's interesting to notice that sparse model sometimes works better than DSD model: in the last image, the sparse model correctly captured the mud puddle, while the DSD model only captured the forest from the background. The good performance of DSD training generalizes beyond these examples, more image caption results generated by DSD training is provided in the appendix.

Table 8: DSD results on NeuralTalk

NeuralTalk	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Sparsity
Baseline	57.2	38.6	25.4	16.8	0%
Sparse	58.4	39.7	26.3	17.5	80%
DSD	**59.2**	**40.7**	**27.4**	**18.5**	0%
Improvement (abs)	2.0	2.1	2.0	1.7	-
Improvement (rel)	**3.5%**	**5.4%**	**7.9%**	**10.1%**	-

3.7 DeepSpeech

We explore DSD training on speech recognition tasks using both Deep Speech 1 (DS1) and Deep Speech 2 (DS2) network [16, 3]. DSD training improves relative accuracy of both DS1 and DS2 models on the Wall Street Journal (WSJ) test sets by $2.1\% \sim 7.4\%$.

The DS1 model is a 5 layer network with 1 Bidirectional Recurrent layer. The DS1 model is described in Table 9. The training data set used for this model is Wall Street Journal (WSJ), which contains approximately 37,000 training utterances (81 hours of speech). We benchmark DSD training on two test sets from the WSJ corpus of read articles (WSJ'92 and WSJ'93). The Word Error Rate (WER) reported on the test sets for the baseline models is different from the those in DeepSpeech2 [3] due to two factors. First, in DeepSpeech2 the models were trained using much larger data sets containing approximately 12,000 hours of multi-speaker speech data. Secondly, WER was evaluated with beam search and a language model in DeepSpeech2; here the network output is obtained using only max decoding to show improvement in the neural network accuracy, filtering out other parts.

Table 9: Deep Speech 1 Architecture

Layer ID	Type	#Params
layer 0	Convolution	1814528
layer 1	FullyConnected	1049600
layer 2	FullyConnected	1049600
layer 3	Bidirectional Recurrent	3146752
layer 4	FullyConnected	1049600
layer 5	CTCCost	29725

Table 10: DSD results on Deep Speech 1: Word Error Rate (WER)

DeepSpeech 1	WSJ '92	WSJ '93	Sparsity	Epochs
Dense Iter 0	29.82	34.57	0%	**50**
Sparse Iter 1	27.90	32.99	32.2%	**50**
Dense Iter 1	27.90	32.20	0%	**50**
Sparse Iter 2	27.45	32.99	31.70%	**50**
Dense Iter 2	**27.45**	**31.59**	0%	**50**
Fully Trained Baseline	28.03	33.55	0%	**150**
Improvement (abs)	0.58	1.96	-	-
Improvement (rel)	**2.07%**	**5.84%**	-	-

The baseline DS1 model is trained for 50 epochs on WSJ training data. The weights from this model are pruned for the sparse iteration of DSD training. Weights are pruned in the Fully Connected layers and the Bidirectional Recurrent layer only. Each layer is pruned to achieve 50% sparsity. This results in overall sparsity of 32.2% across the entire network. This sparse model is re-trained on 50 epochs of WSJ data. For the final dense training, the pruned weights are initialized to zero and trained again on 50 epochs of WSJ training data. This step completes one iteration of DSD training. We use Nesterov SGD to train the model, reduce the learning rate with each re-training, and keep all other hyper parameters unchanged.

We first wanted to compare the DSD results with a baseline model trained for the same number of epochs. The first 3 rows of Table 10 shows the WER when the DSD model is trained for 50+50+50=150 epochs, and the 6th line shows the baseline model trained by 150 epochs. DSD training improves WER by 0.13 (WSJ '92) and 0.56 (WSJ '92) given same number of epochs.

Given more epochs, DSD can further improves the accuracy. For the second DSD iteration, the layer weights from the dense retraining are pruned to preserve 75% of the weights. The overall sparsity for this network is approximately 16%. Similar to the first iteration, the sparse model and subsequent dense model are each retrained for 50 epochs. The learning rate is scaled down for each re-training steps. The results are shown in Table 10. The baseline results in this table correspond to the Same #Epochs Baseline in Table 10.

The second iteration of DSD further improves the accuracy of the model. DSD training provides an overall relative improvement of 2% (WSJ '92) and 5.84% (WSJ '93) compared to the fully trained baseline model. Therefore, with more epochs of training, DSD training achieves better performance than the baseline model that has been fully trained.

3.8 DeepSpeech 2

We also evaluated DSD regularization on the Deep Speech 2 (DS2) network. The DS2 model is described in Table 11. This network has 7 Bidirectional Recurrent layers with approximately 67 million parameters which is around 8 times larger than the DS1 model. The Bidirectional layers in DS2 have three times the number of parameters compared to the Bidirectional layer in DS1. To train this network for DSD experiments, a subset of the internal English training set is used. Specifically, the training set comprises of nearly 1.4 million utterances totaling to 2100 hours of spoken data. The DS2 model is trained using Nesterov SGD for 20 epochs for each training step. Similar to DS1 experiments, learning rate is reduced with each re-training. The other hyper parameters remain unchanged.

Table 11: Deep Speech 2 Architecture

Layer ID	Type	#Params
layer 0	2D Convolution	19616
layer 1	2D Convolution	239168
layer 2	Bidirectional Recurrent	8507840
layer 3	Bidirectional Recurrent	9296320
layer 4	Bidirectional Recurrent	9296320
layer 5	Bidirectional Recurrent	9296320
layer 6	Bidirectional Recurrent	9296320
layer 7	Bidirectional Recurrent	9296320
layer 8	Bidirectional Recurrent	9296320
layer 9	FullyConnected	3101120
layer 10	CTCCost	95054

Table 12: DSD results on Deep Speech 2 (WER)

DeepSpeech 2	WSJ '92	WSJ '93	Sparsity	Epochs
Dense Iter 0	11.83	17.42	0%	**20**
Sparse Iter 1	10.65	14.84	34.3%	**20**
Dense Iter 1	9.11	13.96	0%	**20**
Sparse Iter 2	8.94	14.02	17.2%	**20**
Dense Iter 2	**9.02**	**13.44**	0%	**20**
Fully Trained Baseline	9.55	14.52	0%	**60**
Improvement (abs)	0.53	1.08	-	-
Improvement (rel)	**5.55%**	**7.44%**	-	-

Table 12 shows the results of the two iterations of DSD training. For the first sparse re-training, 50% of the parameters from the Bidirectional Recurrent Layers and Fully Connected layer are pruned resulting in an overall sparsity of 34.3% across the entire network. The Baseline model is trained for 60 epochs to provide a fair comparison with DSD training. The baseline model shows no improvement after 40 epochs. With one iteration of re-training, WER improves by 0.44 (WSJ '92) and 0.56 (WSJ '93) compared to the fully trained baseline model.

Another iteration of DSD training achieves further improvement in accuracy as shown in Table 12. For the second sparse iteration, 25% of parameters in the Fully Connected layer and Bidirectional Recurrent layers are pruned. DSD training achieves an overall improvement of 5.45% (WSJ '92) and 7.44% (WSJ '93) on the DS2 architecture. These results are inline with DSD experiments on the smaller DS1 network. We can conclude that DSD re-training continues to show improvement in accuracy with larger layers and deeper networks.

4 Related Work

Dropout and DropConnect: All of DSD, Dropout [17] and DropConnnect [18] regularize neural networks and prevent over-fitting. The difference is that, Dropout and DropConnect use a *random* sparsity pattern at each SGD iteration, while DSD training learns on a *fixed* sparsity pattern throughout sparse training. Additionally, the last re-densification phase of DSD training has the ability to increase model capacity, while Dropout and DropConnect don't. Our ImageNet experiments show that DSD training works well together with Dropout; they are not mutually exclusive.

Model Compression: Both model compression [19, 8] and DSD training requires network pruning [20, 21]. DSD training can work on top of the compressed model. The difference is that DSD training doesn't require an aggressively pruned network to improve its accuracy. A modestly pruned network (50%-60% sparse) can work well. However, model compression requires aggressively pruning the network to achieve high compression rate.

Sparsity-Constrained Optimization: Truncation-based sparse network has been theoretically analyzed for learning a broad range of statistical models in high dimensions, such as sparse linear regression [5, 7], sparse principal component analysis [6, 22, 23], and sparse latent variable model [24]. This analysis shows that under these models, the truncation-based procedure has provable advantage in statistical accuracy in comparison with their non-truncated counterparts, especially for high dimensions. This line of work lays the theoretical foundation for our pruning procedure, aligns well with our observations, and justifies the desirable performance in terms of prediction accuracy.

5 Discussions

DSD training involves pruning on a converged model. It is more timing-consuming than common training methods. Therefore we are interested if such a procedure can be eliminated, i.e., pruning the model before it converges. The answer is yes. Experiments on AlexNet and DeepSpeech showed that we can reduce the training epochs by fusing the D and S training stages.

The most aggressive approach is to train a sparse network from scratch. We pruned half of the parameters of the BVLC AlexNet and initialized a network with the same sparsity pattern. In our two repeated experiments, one model diverged after 80k iterations of training and the other converged at a top-1 accuracy of 33.0% and a top-5 accuracy of 41.6%. Sparse from scratch seems to be difficult.

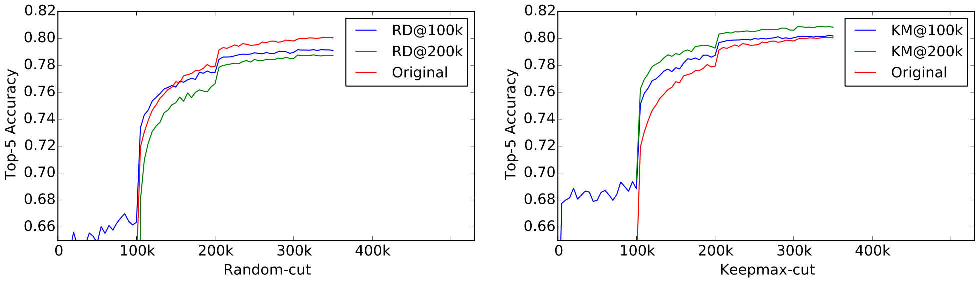


Figure 4: Learning curve of early pruning: Random-Cut (Left) and Keepmax-Cut (Right), pruning after 100k and 200k iterations. All curves have the same learning rate at the same x coordinate.

Another approach is to prune the network during training. We pruned AlexNet after training for 100k and 200k iterations before it converged at 400k. Two pruning schemes are adopted: magnitude-based pruning (Keepmax-Cut) which we used throughout this paper, and random pruning (Random-Cut). After pruning, the 100k-iteration model is trained with a learning rate of 10^{-2} and the 200k-iteration one of 10^{-3}, both are optimal.

The results are plotted in Figure 4. Keepmax-Cut@200K obtained 0.36% better top-5 accuracy, which indicates that we can reduce the training epochs by fusing the DSD training stages. Random-Cut, in contrast, greatly harmed the accuracy and caused the model to converge worse. Random-Cut@100k deteriorates the final accuracy by 1.06% and that on Random-Cut@200k iterations by 1.51%. We conjecture that sparsity helps regularize the network but adds to the difficulty of convergence; therefore, the initial accuracy after pruning is of great importance.

The DeepSpeech experiments also provide some support for reducing DSD training time. In the experiments, weights are pruned early after training the initial model prematurely for only 50 epochs. In contrast, the fully trained model requires 75 epochs. The DSD training achieves better performance than the fully trained model. This shows that we can reduce the number of epochs for the initial training by fusing it with the sparse re-training stage. Similar finding is observed in DeepSpeech-2.

In short, we have three observations: early pruning by fusing the first D and S step together can make DSD training converge faster and better; magnitude-based pruning learns the correct sparsity pattern better than random pruning; and sparsity from scratch leads to poor convergence.

6 Future Work

We consider applying separate learning rates and separate weight decay for old survived weights and new recovered weights for the re-dense training step. After the sparse training, the survived weights have already arrived at a good local minima, so it shouldn't be drastically perturbed, they should have smaller learning rate. On the contrary, those new recovered weights are trained from scratch and should have enough energy to explore an even better local minima, thus they need larger learning rate. To the extreme, we may fix the survived weights (lr=0) and only train the newly recovered weights in the last D step.

7 Conclusion

We introduce DSD, a dense-sparse-dense training method that regularizes neural networks by pruning and then restoring connections. Our method learns which connections are important and regularizes the network by pruning the unimportant connections and retraining. Finally, the pruned connections are restored and the network retrained again. This increases the dimensionality of parameters and model capacity. DSD training prevents overfitting and, at the same time, provides enough model capacity. We highlight our experiments on AlexNet, GoogleNet, VGGNet, ResNet and SqueezeNet on ImageNet dataset, NeuralTalk on Flickr-8K dataset, DeepSpeech and DeepSpeech-2 on WSJ dataset, showing that both CNN, RNN and LSTMs can benefit from DSD training, improving the absolute accuracy by $1\%\sim4\%$. For comparison, the absolute difference between the top contenders in the classification task of ILSVRC 2015 [13] was only 0.014%.

Acknowledgement

We thank Forrest Iandola for useful dicussions.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[2] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[3] Dario Amodei et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv, preprint arXiv:1512.02595*, 2015.

[4] NVIDIA DGX-1 Deep Learning System. http://www.nvidia.com/object/deep-learning-system.html.

[5] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. In *Advances in neural information processing systems*, pages 905–912, 2009.

[6] Xiao-Tong Yuan and Tong Zhang. Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1):899–925, 2013.

[7] Xiao-Tong Yuan, Ping Li, and Tong Zhang. Gradient hard thresholding pursuit for sparsity-constrained optimization. *arXiv preprint arXiv:1311.5750*, 2013.

[8] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[9] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 1MB model size. *arXiv:1602.07360*.

[10] Yangqing Jia. BVLC caffe model zoo. http://caffe.berkeleyvision.org/model_zoo.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[15] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[16] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Ng. Deep speech: Scaling up end-to-end speech recognition. *arXiv, preprint arXiv:1412.5567*, 2014.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

[18] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, pages 1058–1066, 2013.

[19] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*, 2016.

[20] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1990.

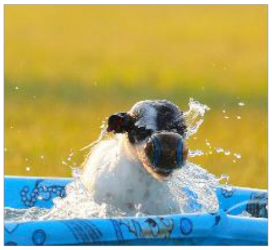
[21] Babak Hassibi, David G Stork, et al. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, pages 164–164, 1993.

[22] Zhaoran Wang, Huanran Lu, and Han Liu. Nonconvex statistical optimization: Minimax-optimal sparse pca in polynomial time. *arXiv preprint arXiv:1408.5352*, 2014.

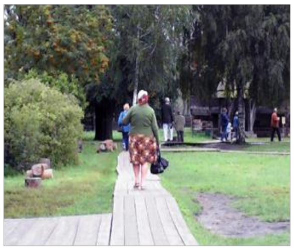
[23] Kean Ming Tan, Zhaoran Wang, Han Liu, and Tong Zhang. Sparse generalized eigenvalue problem: Optimal statistical rates via truncated rayleigh flow. *arXiv preprint arXiv:1604.08697*, 2016.

[24] Zhaoran Wang, Quanquan Gu, Yang Ning, and Han Liu. High dimensional expectation-maximization algorithm: Statistical optimization and asymptotic normality. *arXiv preprint arXiv:1412.8729*, 2014.

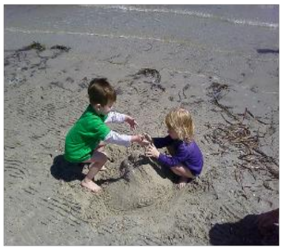
A Appendix: More Examples of DSD Training Improves the Captions Generated by NeuralTalk (Images from Flickr-8K Test Set)



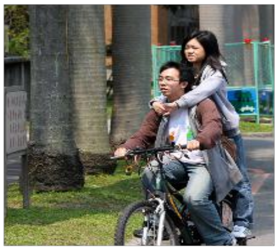
✗ **Baseline**: a boy is swimming in a pool.
○ **Sparse**: a small black dog is jumping into a pool.
✓ **DSD**: a black and white dog is swimming in a pool.



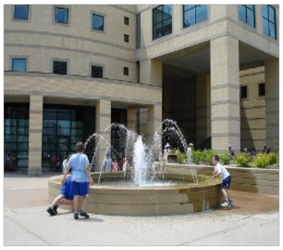
✗ **Baseline**: a group of people are standing in front of a building.
✗ **Sparse**: a group of people are standing in front of a building.
✓ **DSD**: a group of people are walking in a park.



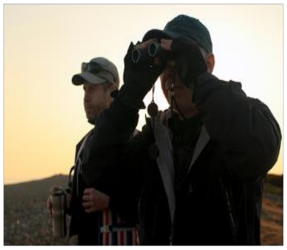
✗ **Baseline**: two girls in bathing suits are playing in the water.
✓ **Sparse**: two children are playing in the sand.
✓ **DSD**: two children are playing in the sand.



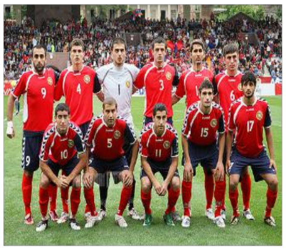
○ **Baseline**: a man in a red shirt and jeans is riding a bicycle down a street.
○ **Sparse**: a man in a red shirt and a woman in a wheelchair.
✓ **DSD**: a man and a woman are riding on a street.



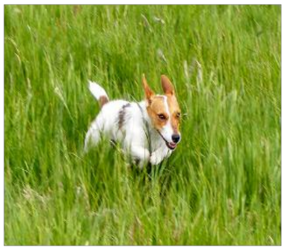
✗ **Baseline**: a group of people sit on a bench in front of a building.
○ **Sparse**: a group of people are standing in front of a building.
✓ **DSD**: a group of people are standing in a fountain.



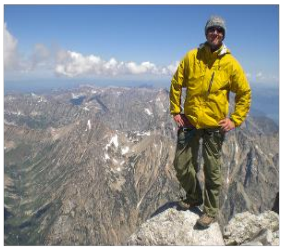
✗ **Baseline**: a man in a black jacket and a black jacket is smiling.
✗ **Sparse**: a man and a woman are standing in front of a mountain.
✓ **DSD**: a man in a black jacket is standing next to a man in a black shirt.



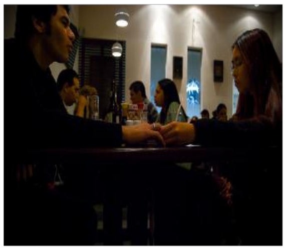
○ **Baseline**: a group of football players in red uniforms.
○ **Sparse**: a group of football players in a field.
✓ **DSD**: a group of football players in red and white uniforms.



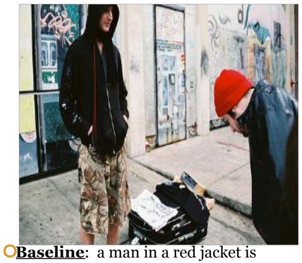
○ **Baseline**: a dog runs through the grass.
○ **Sparse**: a dog runs through the grass.
✓ **DSD**: a white and brown dog is running through the grass.



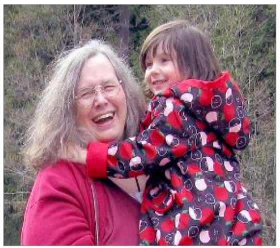
○ **Baseline**: a man in a red shirt is standing on a rock.
○ **Sparse**: a man in a red jacket is standing on a mountaintop.
✓ **DSD**: a man is standing on a rock overlooking the mountains.



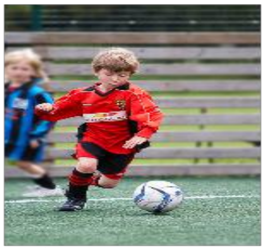
✗ **Baseline**: a group of people are sitting in a subway station.
✗ **Sparse**: a man and a woman are sitting on a couch.
✓ **DSD**: a group of people are sitting at a table in a room.



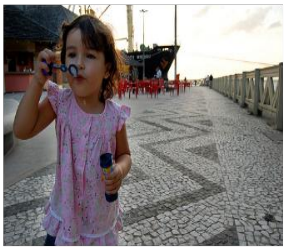
○ **Baseline**: a man in a red jacket is standing in front of a white building.
○ **Sparse**: a man in a black jacket is standing in front of a brick wall.
✓ **DSD**: a man in a black jacket is standing in front of a white building.



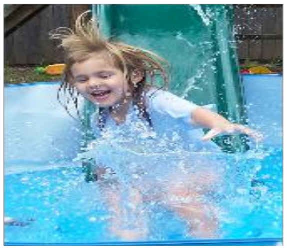
✗ **Baseline**:a young girl in a red dress is holding a camera.
✗ **Sparse**: a little girl in a pink dress is standing in front of a tree.
○ **DSD**: a little girl in a red dress is holding a red and white flowers.



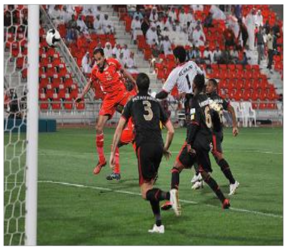
○ **Baseline**: a soccer player in a red and white uniform is playing with a soccer ball.
✓ **Sparse**: two boys playing soccer.
✓ **DSD**: two boys playing soccer.



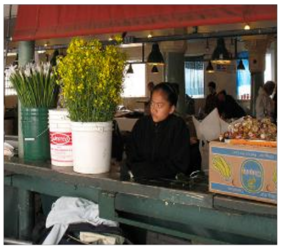
○ **Baseline**: a girl in a white dress is standing on a sidewalk.
○ **Sparse**: a girl in a pink shirt is standing in front of a white building.
✓ **DSD**: a girl in a pink dress is walking on a sidewalk.



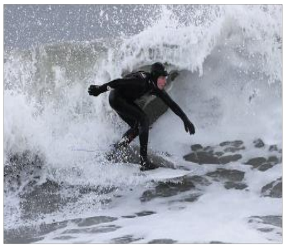
○ **Baseline**: a young girl in a swimming pool.
✗ **Sparse**: a young boy in a swimming pool.
○ **DSD**: a girl in a pink bathing suit jumps into a pool.



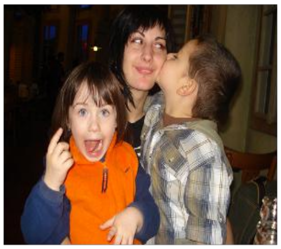
✗ **Baseline**: a soccer player in a red and white uniform is running on the field.
○ **Sparse**: a soccer player in a red uniform is tackling another player in a white uniform.
✓ **DSD**: a soccer player in a red uniform kicks a soccer ball.



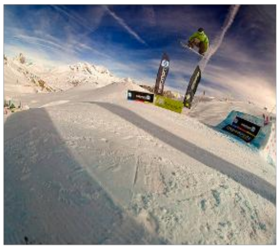
✗**Baseline**: a man in a red shirt is sitting in a subway station.
○**Sparse**: a woman in a blue shirt is standing in front of a store.
○**DSD**: a man in a black shirt is standing in front of a restaurant.



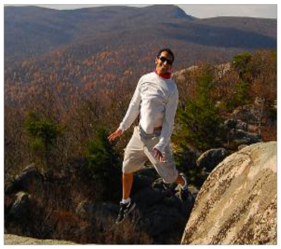
✓**Baseline**: a surfer is riding a wave.
✓**Sparse**: a man in a black wetsuit is surfing on a wave.
✓**DSD**: a man in a black wetsuit is surfing a wave.



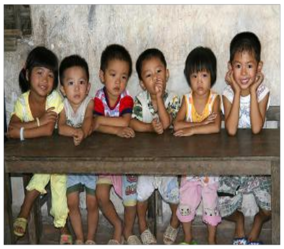
○**Baseline**: two young girls are posing for a picture.
✗**Sparse**: a young girl with a blue shirt is blowing bubbles.
✓**DSD**: a young boy and a woman smile for the camera.



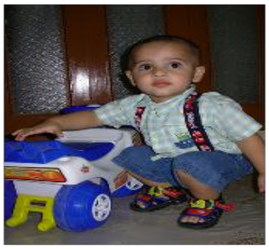
✗**Baseline**: a snowboarder flies through the air.
✓**Sparse**: a person is snowboarding down a snowy hill.
✓**DSD**: a person on a snowboard is jumping over a snowy hill.



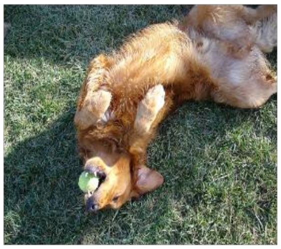
○**Baseline**: a man in a red shirt is standing on top of a rock.
○**Sparse**: a man in a red shirt is standing on a cliff overlooking the mountains.
✓**DSD**: a man is standing on a rock overlooking the mountains.



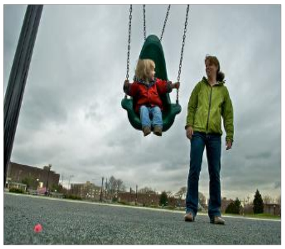
○**Baseline**: a group of people sit on a bench.
○**Sparse**: a group of people are sitting on a bench.
✓**DSD**: a group of children are sitting on a bench.



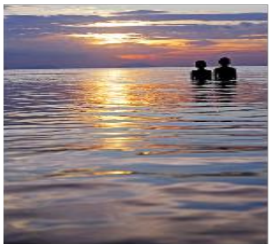
○**Baseline**: a little boy is playing with a toy.
✗**Sparse**: a little boy in a blue shirt is playing with bubbles.
✓**DSD**: a baby in a blue shirt is playing with a toy.



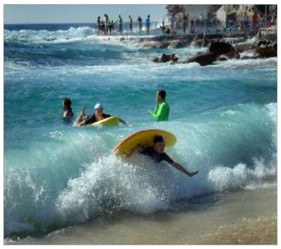
✗**Baseline**: a brown dog is running through the grassy.
✓**Sparse**: a brown dog is playing with a ball.
✓**DSD**: a brown dog is playing with a ball.



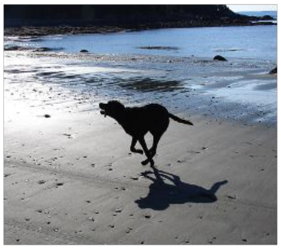
○**Baseline**: a boy in a red shirt is jumping on a trampoline.
○**Sparse**: a boy in a red shirt is jumping in the air.
✓**DSD**: a boy in a red shirt is jumping off a swing.



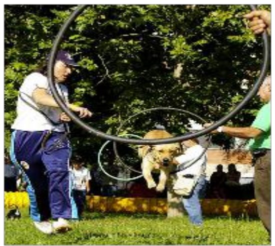
✗**Baseline**: a man is standing on the edge of a cliff.
○**Sparse**: a man is standing on the shore of a lake.
✓**DSD**: a man is standing on the shore of the ocean.



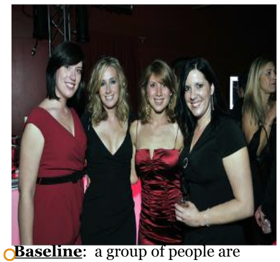
✗**Baseline**: two people are riding a boat on the beach.
○**Sparse**: two people are riding a wave on a beach.
✓**DSD**: a man in a yellow kayak is riding a wave.



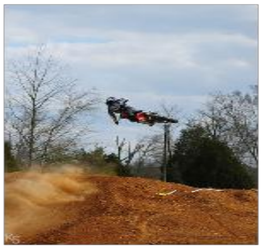
○**Baseline**: a black and white dog is running on the beach.
○**Sparse**: a black and white dog running on the beach.
✓**DSD**: a black dog is running on the beach.



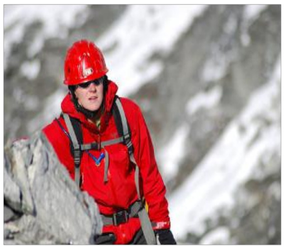
○**Baseline**: a man and a dog are playing with a ball.
✗**Sparse**: a man and a woman are playing tug of war.
✓**DSD**: a man and a woman are playing with a dog.



○**Baseline**: a group of people are standing in a room.
○**Sparse**: a group of people gather together.
✓**DSD**: a group of people are posing for a picture.



○**Baseline**: a man in a red jacket is riding a bike through the woods.
✗**Sparse**: a man in a red jacket is doing a jump on a snowboard.
✓**DSD**: a person on a dirt bike jumps over a hill.



✓**Baseline**: a man in a red jacket and a helmet is standing in the snow.
✓**Sparse**: a man in a red jacket and a helmet is standing in the snow.
✓**DSD**: a man in a red jacket is standing in front of a snowy mountain.