## Motivation

- Computational Efficiency (inkl. Horowitz) !! Why do we do this?
- Improved Generalization (train with low precision (lp) but keep hp weights and use them during test -¿ better results) Some power gains during training, no power gains during test.
- Energy efficiency during test time if you use lp weights, sacrifice accuracy for power saving
- Examples and related concepts: dropout, ReLu (sparse gradient), influence of feature discretization,

## Binarization Schemes

- Deterministic vs. Stochastic

## Gradient Estimation

- Path Derivative Gradient Estimators (reparametrization, ST, SlopeAnnealing)
- Score Function based Gradient Estimators (reinforce+many others)
- Expectation Backpropagation

See Gumble Softmax Paper for overview

## Algorithms Adam + BN

- General Review
- Illustrate their importance for QNNs
- Implement algorithm to see how it works without Adam and BN?

## Full Algorithm

- Try to visualize, illustrate procedure (highlight similarities and differences to conventional NN training and inference)

## Experiments

- Describe different NN architectures (DNN, CNN, RNN)
- Describe data sets (MNIST, CIFAR, ImageNet, PennTree)
- binary vs low precision

## Results

- Training results
- Performance results (look up older benchmarks to compare loss in in accuracy to)
- find comparisons from other references

# Evolution of the QNN paper

### Training NN with lp multiplications

- The inital motivation was to reduce the power consumption of multipliers, which are claimed to be the most power hungry.
- No Binarization of anything yet, only lp weights during forward-prop
- A common pattern already shows: lp multipliers + hp accumulators. Parameter updates are also hp.
- Cost of multiplier $O(precision^2)$ while cost of accumulator $O(precision)$
- Problem with fixed point: low dynamic range $\rightarrow$ dynamic fixed point.
- Evaluation of maxout on MNIST, CIFAR, SVHN with three formats (floating point, fixed point, dynamic fixed point)
- Techniques: Maxout, dropout, momentum, weight decay, dyn. fixed point, update prc vs prop prec.
- Comparison floating point vs. fixed point ?
- Explain demand for sufficient precison of update (compared to Propagations) due to SGD updates
- Half-precision has little to no impact (hp fine-tuning?)
- Couple of plots showing final test error as function of everything
- References to lp network training in the 90ies!
- No evaluation of computational gains, only show how robust NNs are to lp muls.

### Binary Connect: Training DNNs with binary weights during propagations

- Emphasis shifts from efficiency to generalization properties (ala dropout)
- Only binarization of weights, activations not yet binary! Backprop still contains muls!
- mention discretization as form of noise which preserves expected value of weights.
- Again: need for hp accumulation (cite several studies incl haml limtied precision one), Interesting: reference which says brain synapse precision is around 6-12 bits
- Relations to dropout, dropconnect (!), variational weight noise?
- DropConnect: Only expected value of weights needs hp
- Another reference about hardware cost of add+mul [22]
- Binarization: Deterministic vs. stochastic, Hard sigmoid more efficient!
- Algorithm: Biases not binarized!
- Tricks: Weight clipping, batch norm, adam, normalized initialization, L2-SVM output layer
- Inference: Three ways: keep binary weights, use hp weights or ensemble of binary networks (sampled)
- Compute dropout+DNN Performance numbers for CIFAR + SVHN?
- Fig 1: feature coadaption comparison
- Ref. [39] about binary DNN: Fixed-Point feedforward deep Neural...retrains network!
- 2/3 of all muls are due to forward/backward prop
- No demonstration of power saving/efficieny gains

**Neural Networks with few Multiplications**

- Muls again in foucs ("Multiplier light networks")
- Binarization of weights + quantization (!) of activations (but only in backprop), activations in forward phase are not quantized!
- No quantization of BN or ADAM
- Convert muls in backprop to bitshifts by pow2 quantization of activations
- Ref backprop without muls 1999
- First mention of random number generation cost
- Ternary connect -1,0,+1, sampling scheme similar to lp reference!
- how does hp error signal back prop? how about hp input?
- Most muls in weight updates (2MN+3M muls at least)
- Mention muls incurred by batch norm! Cost of divison!
- Table 2 analysis of total num of muls
- Compares only to ordinary SGD (but ok they use the same for the QNN)
- QNN converges slower but better in the end fig 1
- They dont explain the bit shift operations properly
- non-uniform distribution of activations
- Explanation of regularization by lp weights - VERY similar to entropy SDG ideas! large-basin solution, small description length, Additional reference (Neelakatan: Adding gradient noise)

**Quantized Neural Networks**

- Very lp activations + 6bit gradients (enabling bit-wise ops)
- first to actually consider gradient of det./stoch. binary units.
- first experiments on RNNs + ImageNet
- Extensive references (e.g. network compression for inference)
- MAC ops replaced by XNOR and popcount ops
- CNNs benefit mainly from activation quantization (large neuron to weight ratio)
- Binary kernel repetition (worth mentioning?)
- Abandon stochastic binarization (only in activations at train time)
- NEW: Gradient estimators (clipped straight through) due to binary activations. But: weights are binarized as well, but still they take gradients w.r.t. weights as if they were continuous...
- Shaky argument about independence of binarizations for the "derivation" of the clipped ST estimator. But intuitively ok (eq. 6)
- Address cost of BN, which is particularly large for CNNs + inverse sqrt
- How does the approximate shift work in case of negative argument?
- Make screenshpt of "very unoptimized" baseline gpu kernel (theano repo), shift based BN+ADAM are in Torch repo

- Algorithm 1: input binarization? activation functions missing - no, they use det./stoch. binary units!
- Need to disentangle back-prop equations...
- What about multiplications in BackBatchNorm? No mention.
- Input handling: 8bit fixed point , check algorithm 4...
- More than one bit: Quantization+XNORpopcount, Ref to DoReFa net and logarithmic data representation papers.
- Extensions:
  - Approximate BackBatchNorm as well (if not already done)
  - Use non-uniform quantization scheme for weights/activations
  - consider a gradient estimate for quantized weights and maybe a better estimate than the straight through for activations.
  - investigate degradation depending on which layers are quantized. Are early layers more sensitive? Where can one quantize more aggressively?

**ZipML Framework**

- They apply (non-uniform) quantization scheme only to input data (?)
- How would you apply such an adaptive scheme to weights, activations, gradients?

**XNOR-Net**

- Their comparison to BNN on ImageNet is really strange, did they implement it badly? Because back then BNN did not yet have numbers for imagenet, but they claim it performs worse than BNN. Only later Bengio published (better!) numbers than XNOR.
- Memory/op count for AlexNet (compare DNN and CNN weight/activation counts)
- They consider binary weights+activations and only binary weights.
- Mention several network compression approaches (on pretrained networks)
- Quite extensive review of related techiques.
- Interesting equivalence of sign(W) to minimum square distance to W
- They claim "there is no need to keep hp weights"...in contrast to BNN!
- Procedure for activation binarization not quite as elegant (quite heuristic).
- Influence of pooling on binarized input, use of relu after binary conv.
- They actually quantize the gradient as well, but with max scaling - drop in accuracy only 1.4%...suspicious.
- Very similar to QNN, only difference is the scaling factor
- Speed-up calculations (62x in theory)
-

**DoReFa-Net**

- Gradients in BNN have hp, backprop convolutions consume much time, but actually they only sum hp numbers, no muls. DoReFa-Net tries to replace those with bit conv kernels. So BNN comes from the mul side, but DoReFa-Net also complains about hp adds...
- Introduces low bit weights, activations and gradients.
- Why does XNOR gradient binarization not count as binary?
- Configuration-Space Exploration, precision matters most in gradients, activations and weights (in this order). (W,A,G) ca (1,2,4)bits
- Gradients need to be stochastically quantized + larger value range.
- They address the need to use the straight through estimator for quantization, however, they simply use the identity constrained to [0,1]. Is there something better for quantize_k?
- Discussion of straight-through estimators for BNN and XNOR
- XNOR-Net: Scaling factors make bit-convs impossible for weights*grads (?)
- They fail to reproduce results of XNOR-Net for binary neurons! XNOR reports severe degradation for BNN...what is wrong there?
- They report strong dependence of results on activation function
- Funny gradient quantization, mapping to [0,1] plus artificial noise which they say is critical. + each image in batch has it own scaling factor
- First+last layer weights not quantized! (larger degradation)
- Question: Can you get good performance with just low precsion but without all the hocus pocus?
- They recgnize need to explicitly take into account the gradient estimate for the weight binarization!
- How do you backprop through pooling?
- Number of channels affects robustness to quantization (table 1). more complex models are more robust.
- They have problems to close gap between lp and hp models, they need to use hp initialization for lp models.
- NO efficiency evaluation

**DropConnect**

- Similar to Dropout, but they put connections to zero
- Derive bounds on generalization performance
- Regularization: weight decay, Bayesian, early stopping, dropout, weight elimination, mention more? maxnorm,
- Why is Dropconnect (and dropout) only suitable for fully conn? ok less parameters in convs, but actually dropout claims positive effects as well
- Quite intensive: different mask for each sample
- Dropout does r*a(Wx) but they do a(r*Wx) which introduces problems with proper sampling at test time
- Some formal insights to dropout
- Still it seems that they did not properly read dropout (FC issue, averaging at test time)

**Outline Talk**

- Motivation: Power wall, limitations, Horowitz, Sejnowski, why turn to spezialized hardware, no free lunch theorem? Potential computational savings by reduced precision
- Show that quantization leads to less degradation than one might expect. In constrast: it can even improve generalization under certain circumstances. Make analogy to entropy-SGD.
- Analysis of forward+backward propagation ops: Identify bottlenecks and demonstrate quantization of ops step by step.
- Illustrate concepts like Straight-Through-Estimator
- Explore configuration space: which parts are sensitive to quantization? Provide some intuition (from experiments). NN architecture, dataset, optimization, etc.
- Tricks: Show what tricks are used. Which are really necessary?
- Hidden ops & performance holes: Point out where people ignore bottlenecks and where they introduce new ones.

**Questions**

- what is good/noteworthy about QNNs?
- Where do the most significant benefits (computation and accuracy) come from (which aspects)?
- what are the major problems with QNNs theoretically/practically?
- what do we need to solve before QNNs can be fully employed?
- how could one extend/improve QNNs?
- How should I go about showing computational gains? All these papers pretty much avoid to show it explicitly.
- How does the ¡¡¿¿ shift thing work?
- ZipML applies quantization scheme only to input?
- XNOR-Net...how trustworthy?
-