Project Proposal Top-KAST: Top-K Always Sparse Training

Top-KAST is a generic method that preserves constant sparsity throughout training (in both the forward and backward-passes). Hardware support for sparsity is growing (think Nvidia’s Ampere architecture). Motivation: larger sparse networks outperform smaller dense networks (e.g. WaveRNN). Status quo: Mimic sparsity by using dense kernels with hard masking

* Group members: Lisa Wimmer, Sven Lorenz, Andreas Klaß
* **Define** the different **building** **blocks** that need to be implemented:
  + Create a new layer module?
  + Rewrite the optimizer => could become very low-level
  + Forward Pass:   
    set of weights A used to compute output
  + Backward Pass
    - Construct gradient vector to update weights in superset B ⊃ A
  + Constructing alpha parameter / selection process periodically:
    - Implement several possibilities to calculate norms (e.g. Euclidean)
  + Auxiliary exploration regularization loss
* Clearly defines what **requirements** the final **code** must fulfill
  + Pretty much nothing except Tensorflow or Pytorch
  + Generic procedure that can be applied to any network
  + Python 3.7+
  + Language: Pytorch (?)
* Must define how the **framework** can and will be **test**ed

By reproducing the paper results:

* + - ImageNet
    - Transformer-XL
* What do we start with? Which code parts already exist?
  + <https://github.com/google-research/rigl> (tensorflow!)  
    -> similar method which the authors mention a lot
  + Pseudocode:
* Particularly **challenging** parts:
  + Custom sparse kernels (was das?)
  + Rewriting optimizers directly is very low-level
  + Writing generic forward-backward passes that can be applied everywhere is challenging. Compatibility with generic modules is not trivial.
  + In general Top-KAST can be implemented by modifying the parameters used in the forward pass and applying a gradient with respect to only some of the weighs in the backward pass
* Main **contribution**:
  + We actually wrote the code for the paper
  + We reproduced the results from the paper
  + Could be added to the pytorch library as a new layer module or option

Related works and references

<https://cs.stanford.edu/~matei/papers/2020/sc_sparse_gpu.pdf>

Pseudocode:

Algorithm 1 TopKAST ####################

// First perform a Top-K

dense\_params = initialise()

fwd\_params = TopK(dense\_params, X%)

bwd\_params = TopK(dense\_params, Y%)

just\_bwd\_set = set(bwd\_params) - set(fwd\_params)

...

// Output with just the TopK

params output = model(fwd\_params, input)

loss = loss\_fn(output)

// Exploration L2 Loss

loss += l2(fwd\_params) + l2(just\_bwd\_set) / (X/100)

...

// Update only the bwd params

bwd\_params = bwd\_params - grad(loss, bwd\_params)