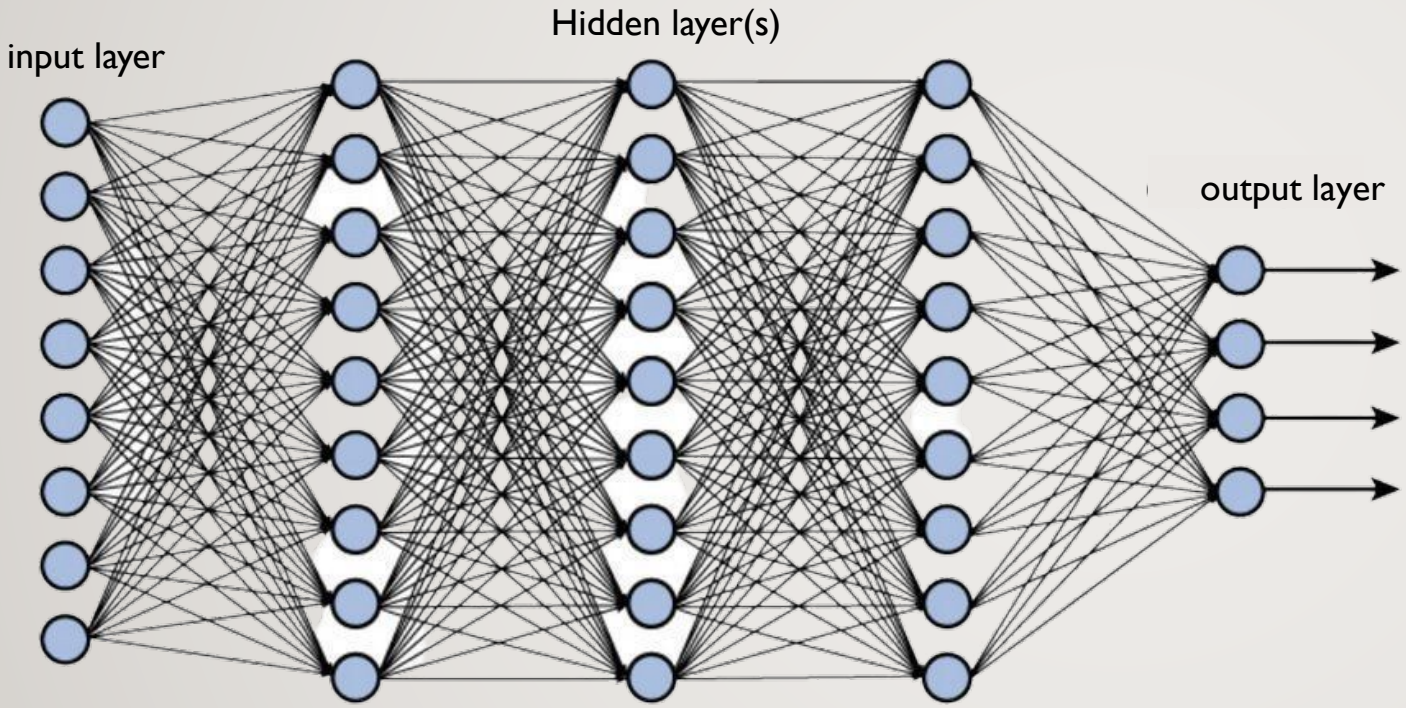
THE STATE OF SPARSITY

G.Marco Nuzzarello
Walter Maltese
Pierandrea Morelli

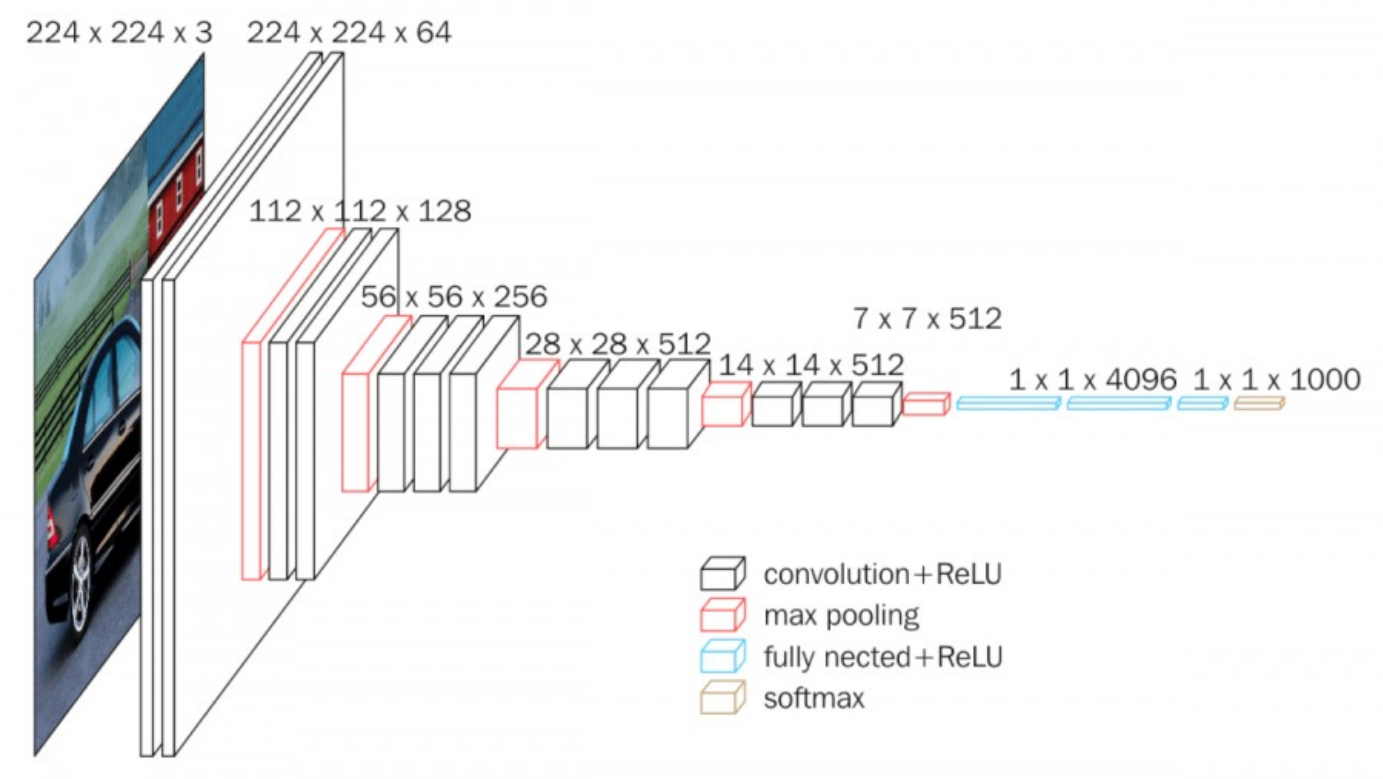
DEEP NEURAL NETWORK



- Image classification
- Machine translation
- Text-to-speech
- ..

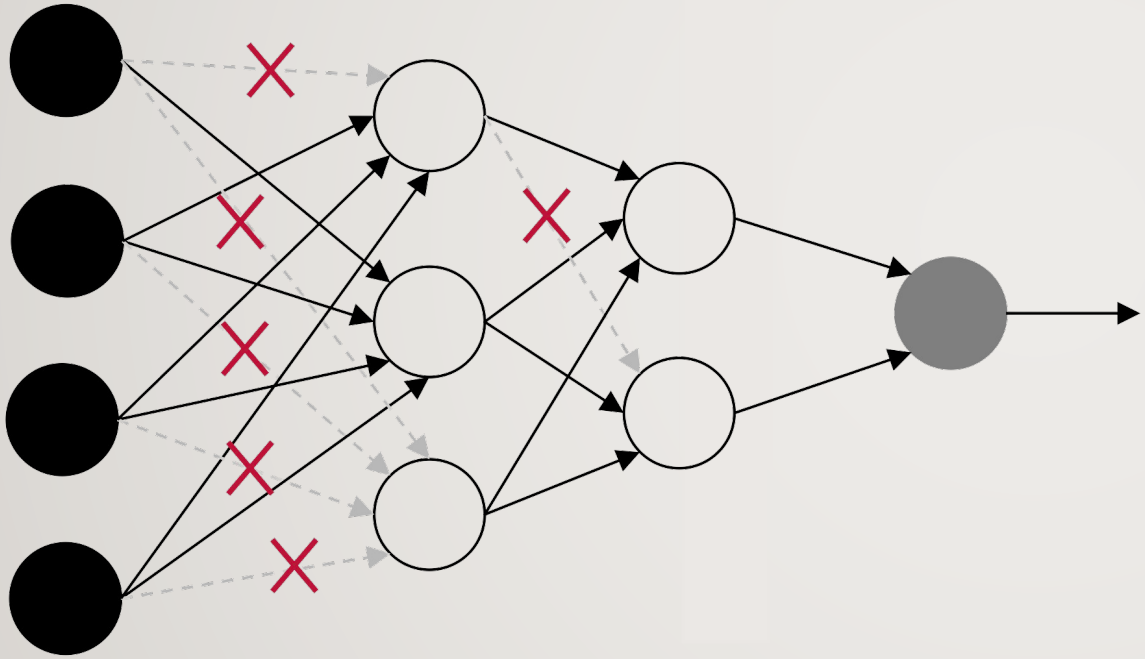
- > 10M parameters

VGGNET (2014)



- Over 100 M parameters depending on depth

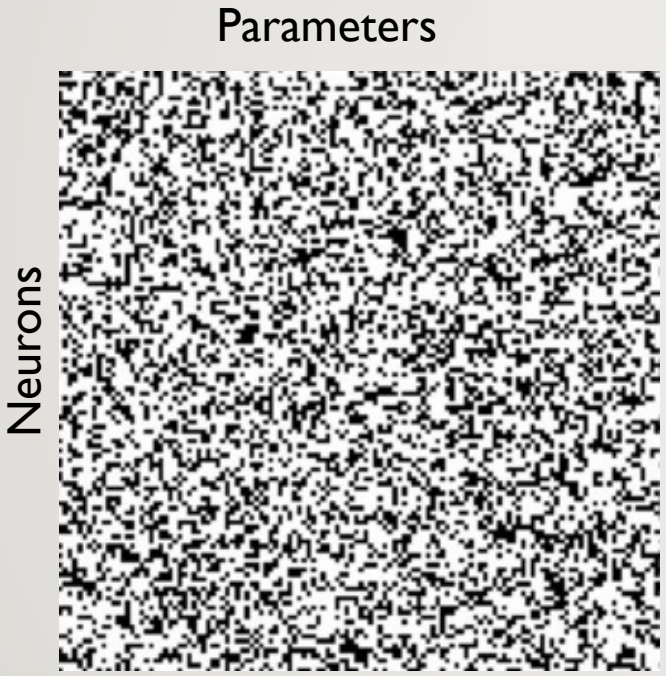
SPARSIFICATION



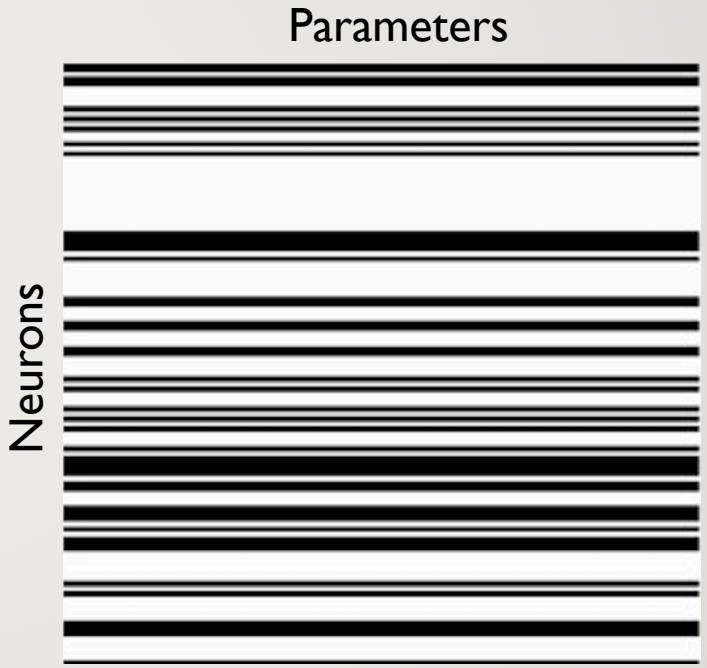
$$\begin{bmatrix} 0 & 0 & -2.5 \\ 1.12 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Sparse weights matrix

SPARSITY VARIANTS



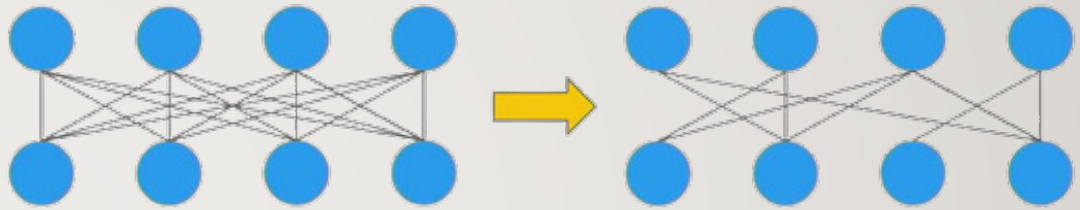
Unstructured



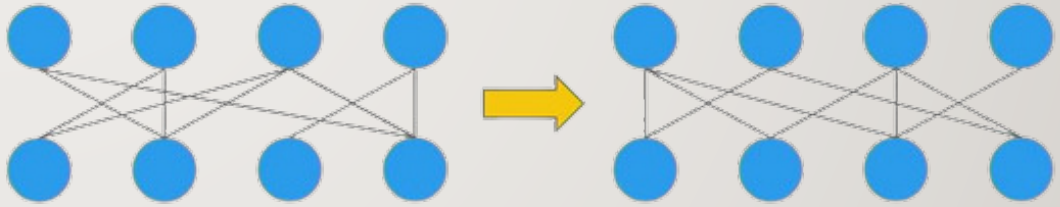
Structured

TRAINING SPARSE NEURAL NETWORKS

- **Dense-to-Sparse**: Model is dense during training. Produces sparse models that can be used for fast inference.

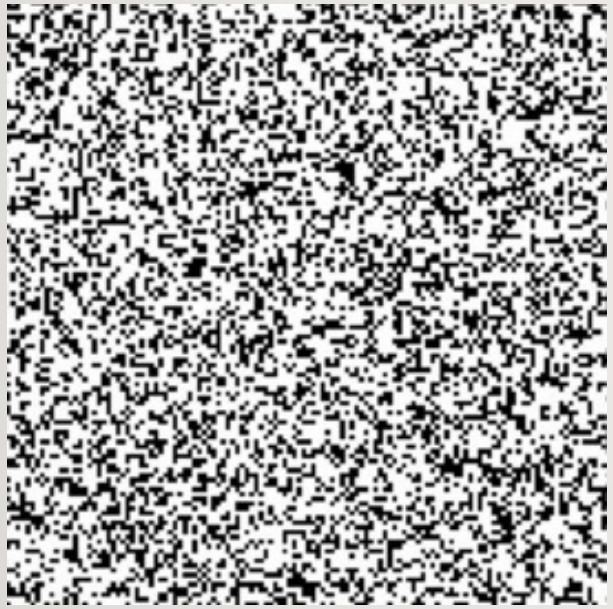


- **Sparse-to-Sparse**: Model is sparse during training. Faster training and inference. More difficult to maintain quality.

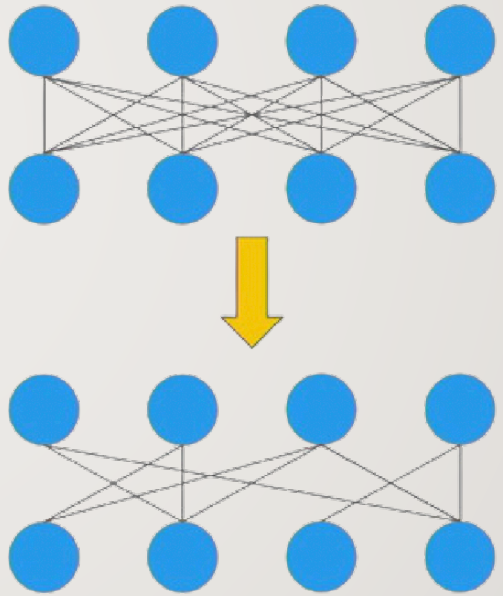


FOCUS OF THIS PROJECT

This work focuses on **unstructured sparsity** and **dense-to-sparse training**

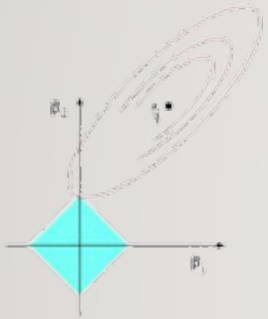


Unstructured

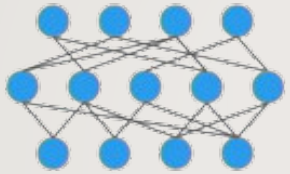


Dense-to-sparse training

SPARSIFICATION ALGORITHMS



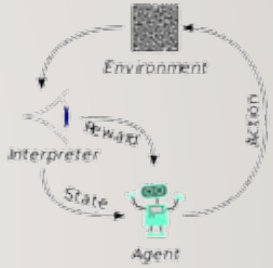
Regularization



Heuristics

Bayesian

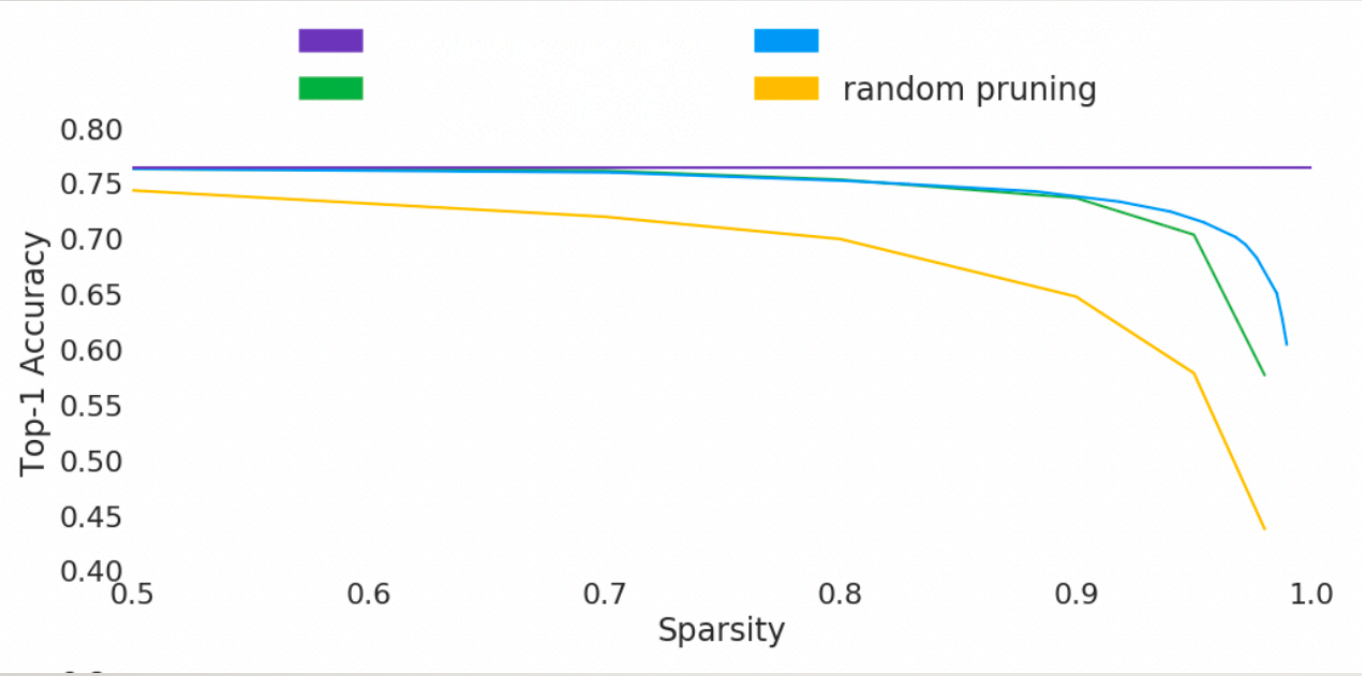
Loss Approximations



Reinforcement Learning

We chose to benchmark three of the top performing techniques: **Variational Dropout** (Kingma et al., Molchanov et al.), **L0-Regularization** (Louizos et al.) and **Magnitude Pruning** (Zhu et al.).

RANDOM PRUNING



This technique is intended to represent a lower-bound of the accuracy-sparsity trade-off curve.

VARIATIONAL DROPOUT

$$\mathcal{L}(\phi) = -D_{KL}(q_\phi(\mathbf{w})||p(\mathbf{w})) + L_\mathcal{D}(\phi)$$

$$\text{where } L_\mathcal{D}(\phi) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbf{E}_{q_\phi(\mathbf{w})}[\log p(\mathbf{y}|\mathbf{x},\mathbf{w})]$$

- Consider the setting of a dataset *D* of *N* i.i.d. samples (x, y) and a standard classification problem where the goal is to learn the parameters w of the conditional probability p(y|x, w).
- We optimize the parameters ϕ of some parameterized model qϕ(w) such that qϕ(w) is a close approximation to the true posterior distribution p(**w**|*D*) as measured by the **Kullback-Leibler divergence** between the two distributions.

L0 - REGULARIZATION

$$\theta_j = \tilde{\theta}_j z_j$$
$$\text{where } z_j \sim \min(1, \max(0, \bar{s})),\ \bar{s} = s(\zeta - \gamma) + \gamma$$
$$s = \text{sigmoid}((\log u - \log(1-u) + \log \alpha)/\beta)$$
$$\text{and } u \sim \mathcal{U}(0, 1)$$

Louizos et al. (2017b) use the following estimate for the model parameters:

$$\boldsymbol{\theta}^* = \tilde{\boldsymbol{\theta}}^* \odot \hat{\mathbf{z}}$$
$$\hat{\mathbf{z}} = \min(\mathbf{1}, \max(\mathbf{0}, \text{sigmoid}(\log \boldsymbol{\alpha})(\zeta - \gamma) + \gamma))$$

To optimize the l0-norm, we reparameterize the model weights θ as the product of a weight and a random variable

GOAL OF THIS PROJECT

- Can learned sparse architectures be trained from scratch to the same test set accuracy as those trained with joint sparsification and optimization?

- Do the top performing techniques in the literature actually perform the best on large-scale tasks?

MAGNITUDE PRUNING

- **Magnitude-based weight pruning schemes** use the **magnitude** of each weight as a proxy for its importance to model quality and remove the least important weights according to some sparsification schedule over the course of training.

VARIANCE OF DIFFERENT SCHEDULES

Many **variants** have been proposed (Collins & Kohli, 2014; Han et al., 2015; Guo et al., 2016; Zhu & Gupta, 2017), with the key differences lying in:

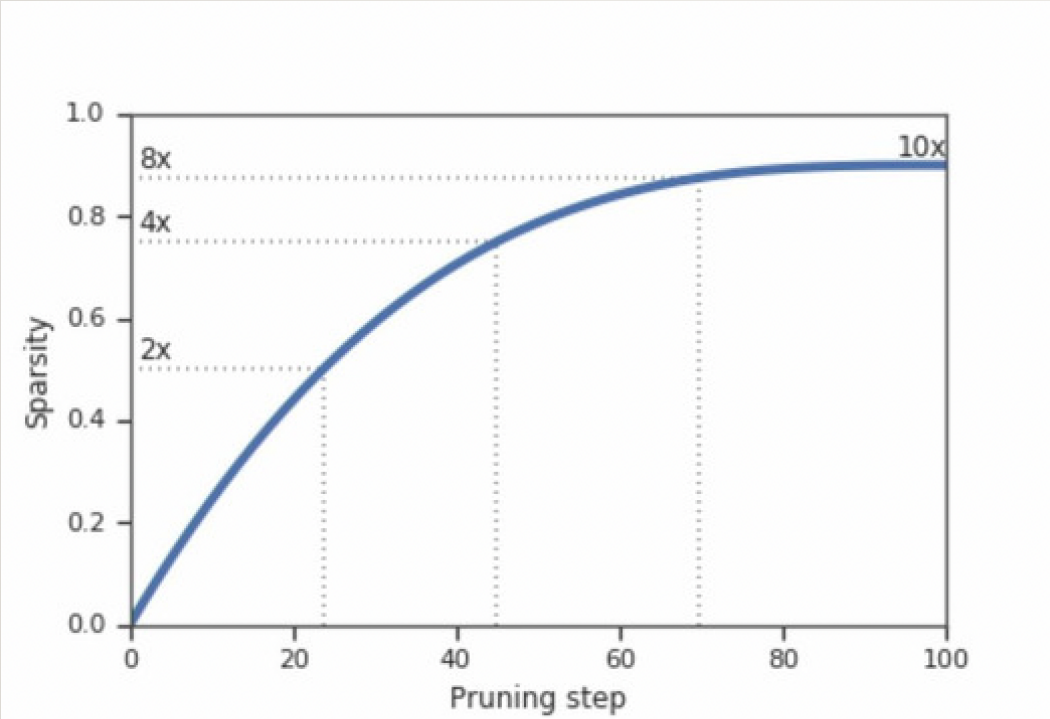
- when weights are removed;
- whether weights should be sorted (to remove a precise proportion) or thresholded (based on a fixed or decaying value);
- whether or not weights that have been pruned still receive gradient updates and have the potential to return after being pruned.

Variants have also been proposed that maintain a constant level of sparsity during optimization to enable accelerated training (Mocanu et al., 2018).

OUR IMPLEMENTATION I

- For our experiments, we use the approach introduced in **Zhu & Gupta** (2017).
- It makes use of a **gradual sparsification schedule** with sorting-based weight thresholding to maintain accuracy while achieving a user specified level of sparsification.
- These features enable **high compression ratios at a reduced computational cost**, relative to the iterative pruning and re-training approach used by Han et al. (2015), while requiring less hyperparameter tuning relative to the technique proposed by Guo et al. (2016).

OUR IMPLEMENTATION II



Sparsification schedule for magnitude pruning. The final sparsity, pruning frequency, and begin/end pruning step are hyperparameters.

- It uses this specific sparsification schedule to remove weights over the course of training;
- During training (we simulate sparsity with a binary mask), the model is fully dense, and weights that are masked can still receive gradient updates and can become unmasked again during training.

OTHER IMPLEMENTATIONS

Han et al. (2015) use iterative magnitude pruning and re-training to progressively sparsify a model. The target model is first trained to convergence, after which a portion of weights are removed, and the model is re-trained with these weights fixed to zero. This process is repeated until the target sparsity is achieved.

Guo et al. (2016) improve on this approach by allowing masked weights to still receive gradient updates, enabling the network to recover from incorrect pruning decisions during optimization. They achieve higher compression rates and interleave pruning steps with gradient update steps to avoid expensive re-training.

- Can **learned sparse architectures** be trained **from scratch** to the same test set accuracy as those trained with **joint sparsification and optimization**?

1^ GOAL OF THIS PROJECT

- Focus on whether its possible to take a sparse neural network architecture that was learned through some kind of pruning procedure and train it from scratch to the full quality that was realized when sparsification and optimization were performed jointly.
- Works like "The lottery ticket hypotesis" and "Rethinking the value of network pruning" show that **this is possible**, but they mostly focus **on smaller datasets or low sparsity levels**.
- It is possible to reproduces these phenomenas on **large scale models and datasets**?

BENCHMARKS ON SMALL-SCALE MODELS

- **Lottery ticket hypothesis (Frankle & Carbin, 2018)**: "ones you perform pruning to learn a sparse weight mask, you can take that **final weight mask** as well as the **same weight inizalitation** that was used when the weight mask was learned, **and train it from scratch** without changing the sparse topology at all, **to the same quality** that was **achieved when pruning and optimization were performed jointly**". They provide extensive experimental results, but only on MNIST and CIFAR (small datasets).
- **Rethinking Pruning (Liu et al, 2018)**: similar to the Lottery ticket hypothesis, but it states that "**is only required the final weight mask**, and not the same initialization, **to match the same quality** (achieved when pruning and optimization were performed jointly)".

They both experiments on ImageNet, but only up to 60% sparsity.

Lottery Ticket Hypothesis:
Both final mask & same weight initialization are needed. Results on MNIST and CIFAR.

Rethinking Pruning:*
Only final mask is required. Results on ImageNet but only up to 60% sparsity.

BENCHMARKS ON LARGE-SCALE MODELS

The "scratch paper" (Liu et al. – Rethinking Pruning) introduces a few variances in their experiment:

- The **scratch-e** experiments are trained for the same number of training steps as the standard model;
- The **scratch-b** experiments have the number of training steps doubled, to account for the fact that these models have much less FLOPs;
- The **augmented initialization** is a tweak that is applied to the random weight initialization, owning to the fact that the initial model is sparse in this case.

TAKEAWAYS 1

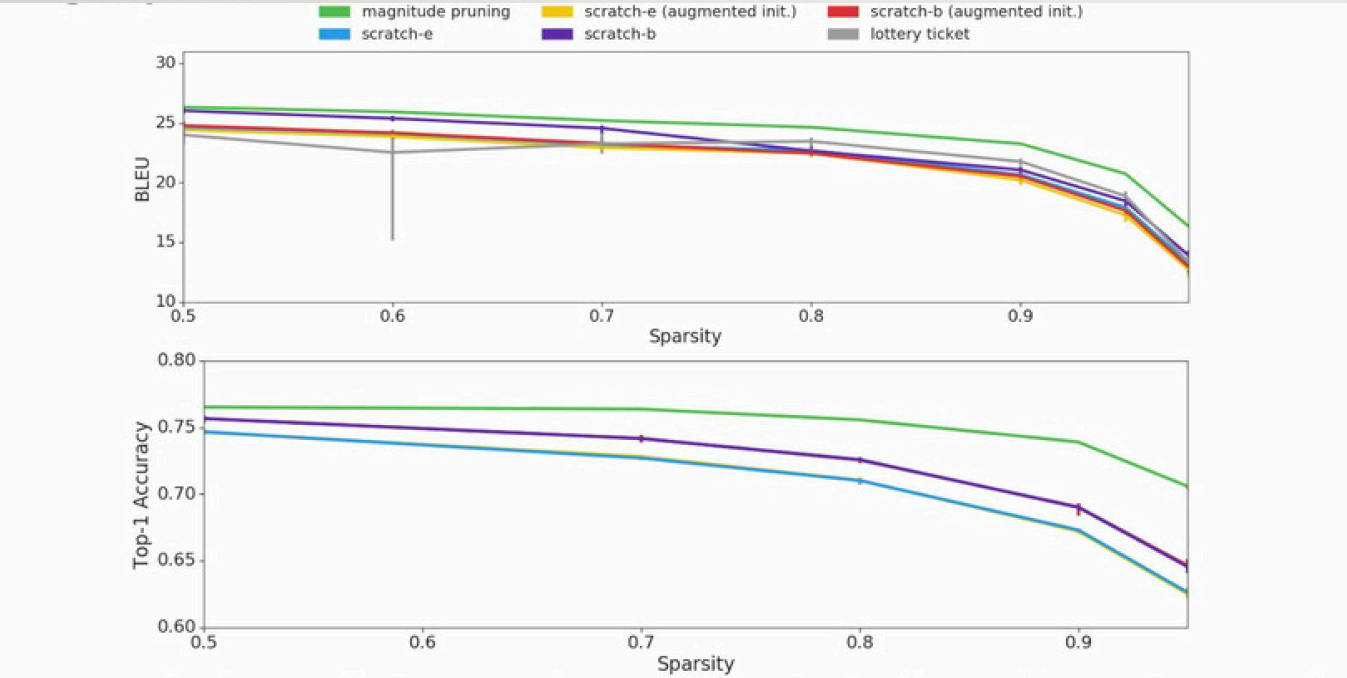


Figure 6: Scratch and Lottery Ticket Experiments with Magnitude Pruning. Top: Transformer results. Bottom: ResNet-50 results. Across all experiments, training from scratch does not match performance achieved with joint sparsification and optimization.

- For *Transformer*, we took the **best magnitude pruning hyperparameters** found in previous experiments, trained 5 replicas of those and, for each of those, we trained 5 replicas with the lottery ticket experiment. Every point on the plot is **25** Transformer models, with error bars being the min/max performances found.
- For *Resnet-50*, we followed the **same procedure**, but with 3 experiments and other 3 experiments respectively. Consequently, each point on the plot is **9** experiments total, with error bars across all of the experiments and all sparsity levels.

The **stratch-b experiments** (trained for twice as long) **did better**, but we were **unable to match the quality of** a well-tuned **magnitude pruning baseline on these large scale models**.

TAKEAWAYS II

- Across both models, we observed that **doubling the number of training steps did improve the quality of the results** for the scratch experiments, but was **not sufficient to match** the test set performance of the **magnitude pruning** baseline.
- **As sparsity increased**, we observed that **the deviation** between the models trained with magnitude pruning and those trained from scratch **increased**.
- For both models, **we did not observe a benefit** from **using the augmented weight initialization** for the scratch experiments.
- For the **scratch experiments**, our results are consistent with the negative result observed by (Liu et al., 2018) for ImageNet and ResNet-50 with unstructured weight pruning.
- For **unstructured weight sparsity**, it seems likely that the phenomenon observed by **Liu et al. (2018)** was produced by a combination of **low sparsity levels** and **small-to-medium sized tasks**.

GOAL OF THIS PROJECT

- **Can learned sparse architectures be trained from scratch to the same test set accuracy as those trained with joint sparsification and optimization?**

- Do the top performing techniques in the literature actually perform the best on large-scale tasks?

GOAL OF THIS PROJECT

- Can learned sparse architectures be trained from scratch to the same test set accuracy as those trained with joint sparsification and optimization?

✓ **NO, Training from scratch does not match the performaces of joint optimization & sparsification on large-scale tasks.**

- Do the top performing techniques in the literature actually perform the best on large-scale tasks?

GOAL OF THIS PROJECT

- Can **learned sparse architectures** be trained **from scratch** to the same test set accuracy as those trained with **joint sparsification and optimization**?

- Do the top performing techniques in the literature actually perform the best on large-scale tasks?

GOAL OF THIS PROJECT

- Can **learned sparse architectures** be trained **from scratch** to the same test set accuracy as those trained with **joint sparsification and optimization**?

✓ **NO**, **Training from scratch does not match the performaces of joint optimization & sparsification on large-scale tasks**.

- Do the top performing techniques in the literature actually perform the best on large-scale tasks?

BONUS I:

THE BENEFITS OF SPARSITY

PERFORMANCE BENEFITS FROM SPARSITY

- *"FASTER CNNS WITH DIRECT SPARSE CONVOLUTIONS AND GUIDED PRUNING"*
- *"Escoin: Efficient Sparse Convolutional Neural Network Inference on GPUs"*
- *"Efficient Neural Audio Synthesis"*
- *"GPU Kernels for Block-Sparse Weights"*

And countless other examples…

FASTER CNNS WITH DIRECT SPARSE CONVOLUTIONS AND GUIDED PRUNING

- The **number of parameters** needed in **CNNs** are often **large** and undesirable. Consequently, various methods have been developed to **prune a CNN** once it is trained.
- Nevertheless, the resulting CNNs offer **limited benefits**. While pruning the fully connected layers reduces a CNN's size considerably, it does not improve **inference speed** noticeably, as **the compute heavy parts lie in convolutions**.
- **Solutions** proposed:

  1) an **high performance sparse convolution design** to **realize simultaneously size economy and speed improvement** while pruning CNNs**,** thanks to an efficient general **sparse-with-dense matrix multiplication** implementation that is applicable to convolution of feature maps with kernels of arbitrary sparsity patterns;

  2) a **performance model** that **predicts sweet spots of sparsity levels** for different layers and on different computer architectures.

ESCOIN: EFFICIENT SPARSE CONVOLUTIONAL NEURAL NETWORK INFERENCE ON GPUS

- **Weight pruning** can compress DNN models by removing redundant parameters in the networks, but it **brings sparsity in the weight matrix**, and therefore **makes the computation inefficient on GPUs**. Although pruning can remove more than 80% of the weights, it actually **hurts inference performance (speed)** when running models **on GPUs**.
- **Solution** proposed: **Escort**, an efficient **sparse CNN method customized for GPU**'s data-parallel architecture. Instead of lowering the convolution onto matrix multiplication, we choose to **directly compute the sparse convolution**.
  To **take advantage of GPU's** tremendous **computational horsepower**, we customize the dataflow and apply a series of optimization techniques based on the **understanding of the memory access pattern**. Escort is implemented using **CUDA** and it is evaluated on NVIDIA GPUs.

EFFICIENT NEURAL AUDIO SYNTHESIS

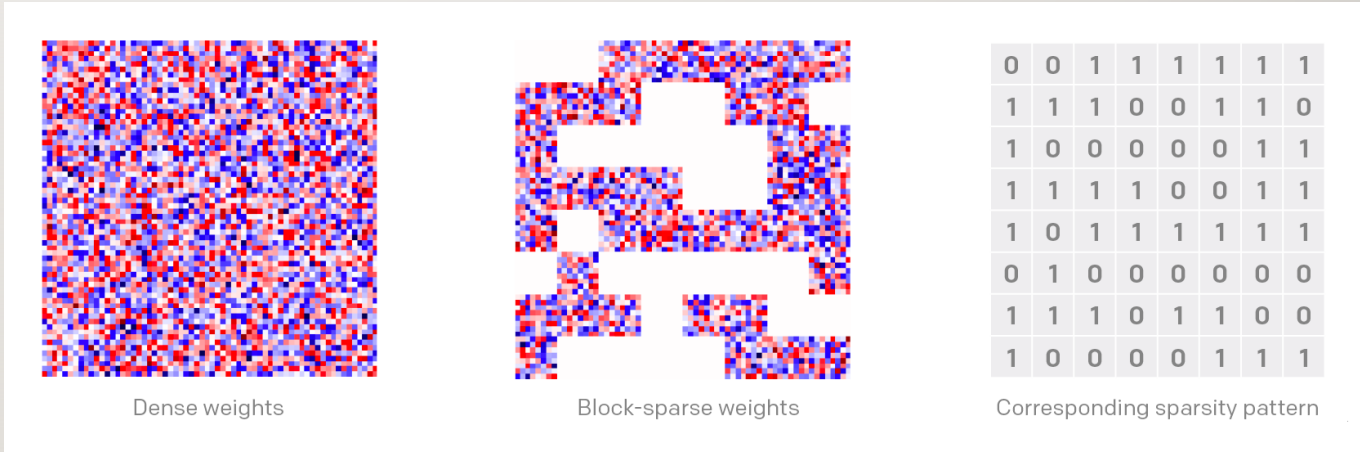
- It is introduced **WaveRNN**, a simple and powerful **recurrent network** for the **sequential modelling of high fidelity audio**, and demonstrated a **high performance** implementation of this model **on GPUs**.
- It is shown that **large sparse models have much better quality than small dense models with** the **same number of parameters** (and this relationship holds for sparsity levels beyond 96%) and demonstrated that sampling time is proportional to parameter count.
- The **small number of weights in** a **Sparse WaveRNN** makes it **possible to sample high-fidelity audio on a mobile low-power CPU** in real time.

GPU KERNELS FOR BLOCK-SPARSE WEIGHTS I

- One issue in the development of model architectures and algorithms in the field of deep learning has been the **lack of an efficient GPU implementation for sparse linear operations**.
- OpenAI released **highly-optimized GPU kernels for** an underexplored class of neural network architectures: **networks** with **block-sparse weights**.
- The **computational cost** of matrix multiplication and convolution **with sparse blocks** is **only proportional to the number of non-zero blocks**. **Sparsity** enables, for example, **training of neural networks** that are much **wider and deeper** than otherwise possible with a given parameter and computational budget.

GPU KERNELS FOR BLOCK-SPARSE WEIGHTS II

- The **kernels** allow **efficient usage of block-sparse weights** in fully connected and convolutional layers. The **sparsity** is **defined at the level of blocks**, and have been **optimized for block sizes** of **8x8**, **16x16** or **32x32**. At the block level, the sparsity pattern is completely configurable.
  Since **the kernels skip computations of blocks that are zero**, the **computational cost** and the **cost for storing the parameters** is only **proportional to the number of non-zero weights**, not the number of input/output features.
- Attained state-of-the-art results in **text-sentiment analysis** and **generative modelling** of text and images.



BONUS II:

PUSHING THE LIMITS OF MAGNITUDE PRUNING

MAGNITUDE PRUNING V2 I

- Given that a **uniform distribution of sparsity is suboptimal**, and the significantly **smaller resource requirements for** applying **magnitude pruning** to ResNet-50 it is natural to wonder how well magnitude pruning could perform if we were to **distribute the non-zero weights more carefully** and **increase training time**.

  We'll see later these results…

BUT… HOW ABOUT MAGNITUDE PRUNING VS OTHER (MORE COMPLEX) TECHNIQUES ?

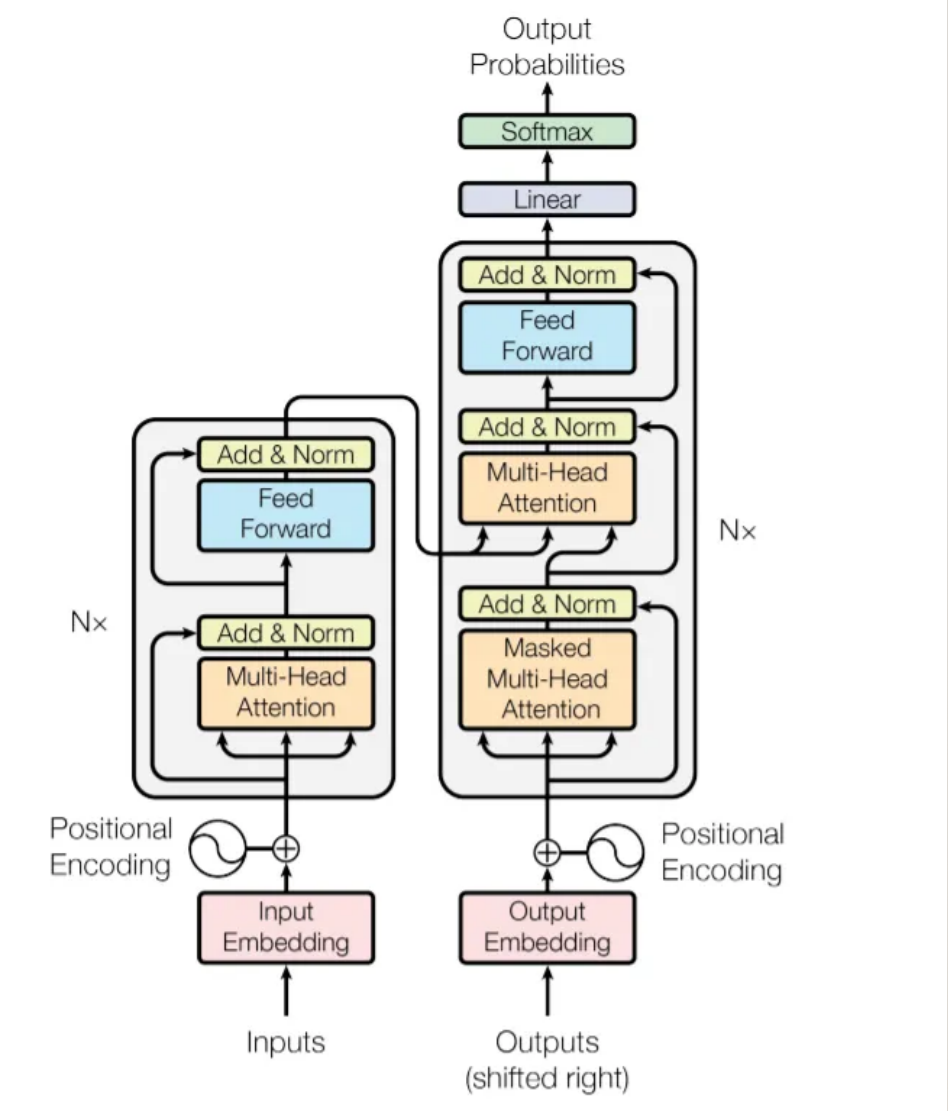
EXPERIMENTAL FRAMEWORK

Transformer on WMT14 EN-DE and RESNET-50 on ImageNet

- Costant number of training steps for all techniques
- Sparsity levels 50-98%
- Extensive Hyperparameter tuning

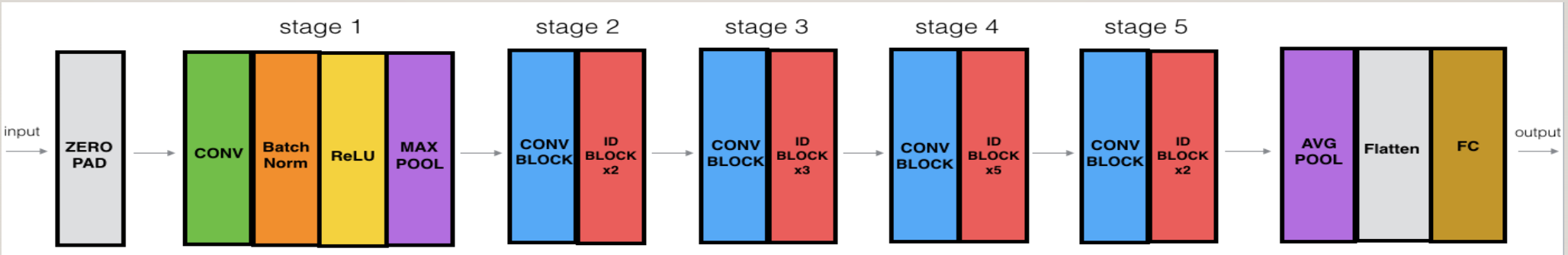
TRANSFORMER

- Transformer is an encoder-decoder architecture. The encoder consists of a set of encoding layers that processes the input iteratively one layer after another and the decoder consists of a set of decoding layers that does the same thing to the output of the encoder.
- The function of each encoder layer is to process its input to generate encodings, containing information about which parts of the inputs are relevant to each other. It passes its set of encodings to the next encoder layer as inputs.
- Each decoder layer does the opposite, taking all the encodings and processes them, using their incorporated contextual information to generate an output sequence.
- To achieve this, each encoder and decoder layer makes use of an attention mechanism, which for each input, weighs the relevance of every other input and draws information from them accordingly to produce the output.
- Each decoder layer also has an additional attention mechanism which draws information from the outputs of previous decoders, before the decoder layer draws information from the encodings.
- Both the encoder and decoder layers have a feed-forward neural network for additional processing of the outputs, and contain residual connections and layer normalization steps.



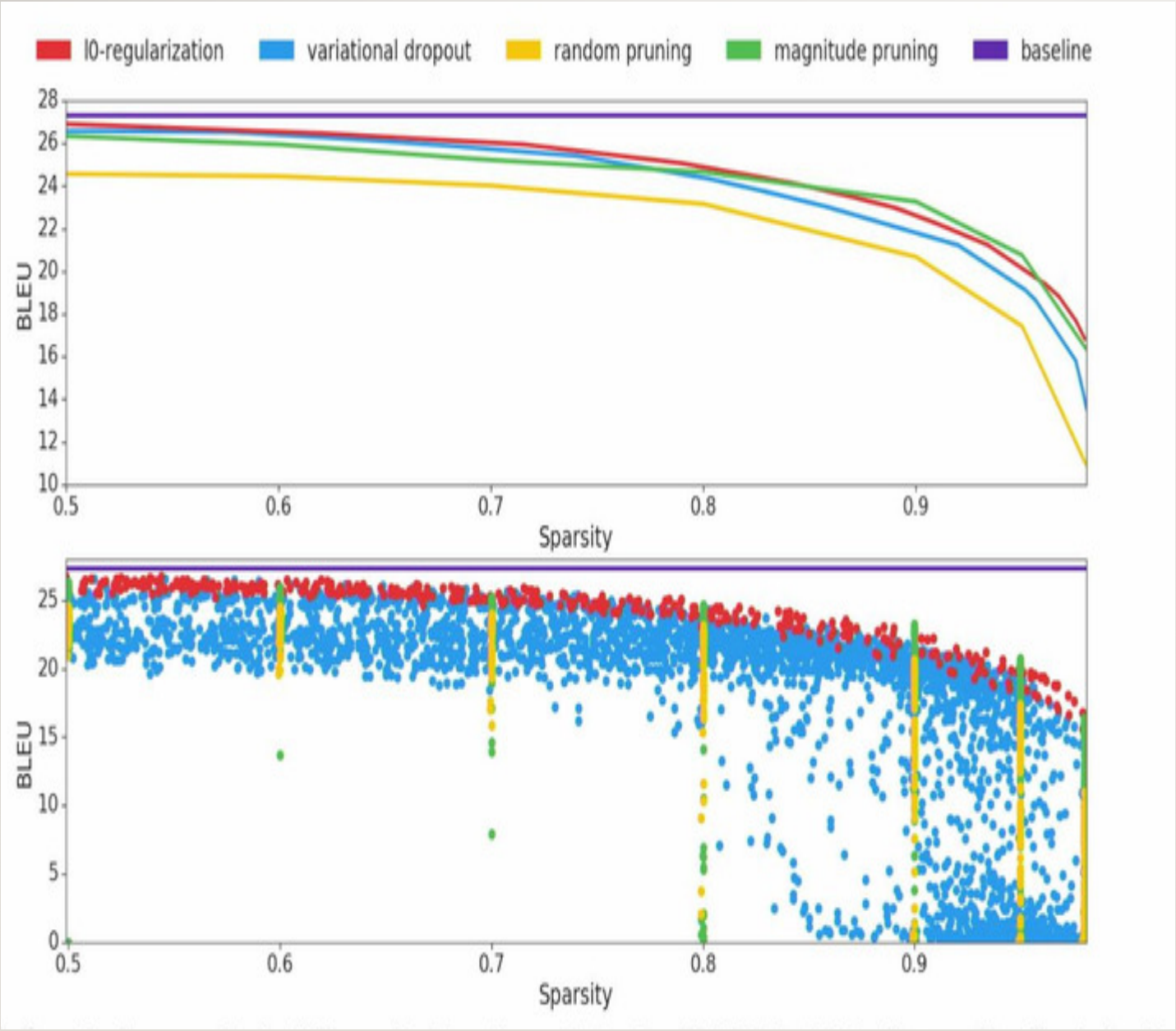
RESNET-50

- The winner of ImageNet challenge in 2015
- The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully.
- Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients, as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient extremely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.
- ResNet first introduced the concept of skip connection to add the output from an earlier layer to a later layer. This helps it mitigate the vanishing gradient problem
- The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters



SPARSE NEURAL MACHINE TRANSLATION

- We adapted the Transformer model for neural machine translation to use these four sparsification techniques
- Trained the model on the WMT14 English-German dataset.
- We sparsified all fully-connected layers, which make up 99.87% of all of the parameters in the model.
- This setup got a baseline BLEU score (Machine-Translation quality parameter) of 27.29 averaged across five runs.
- We extensively tuned the remaining hyperparameters for each technique

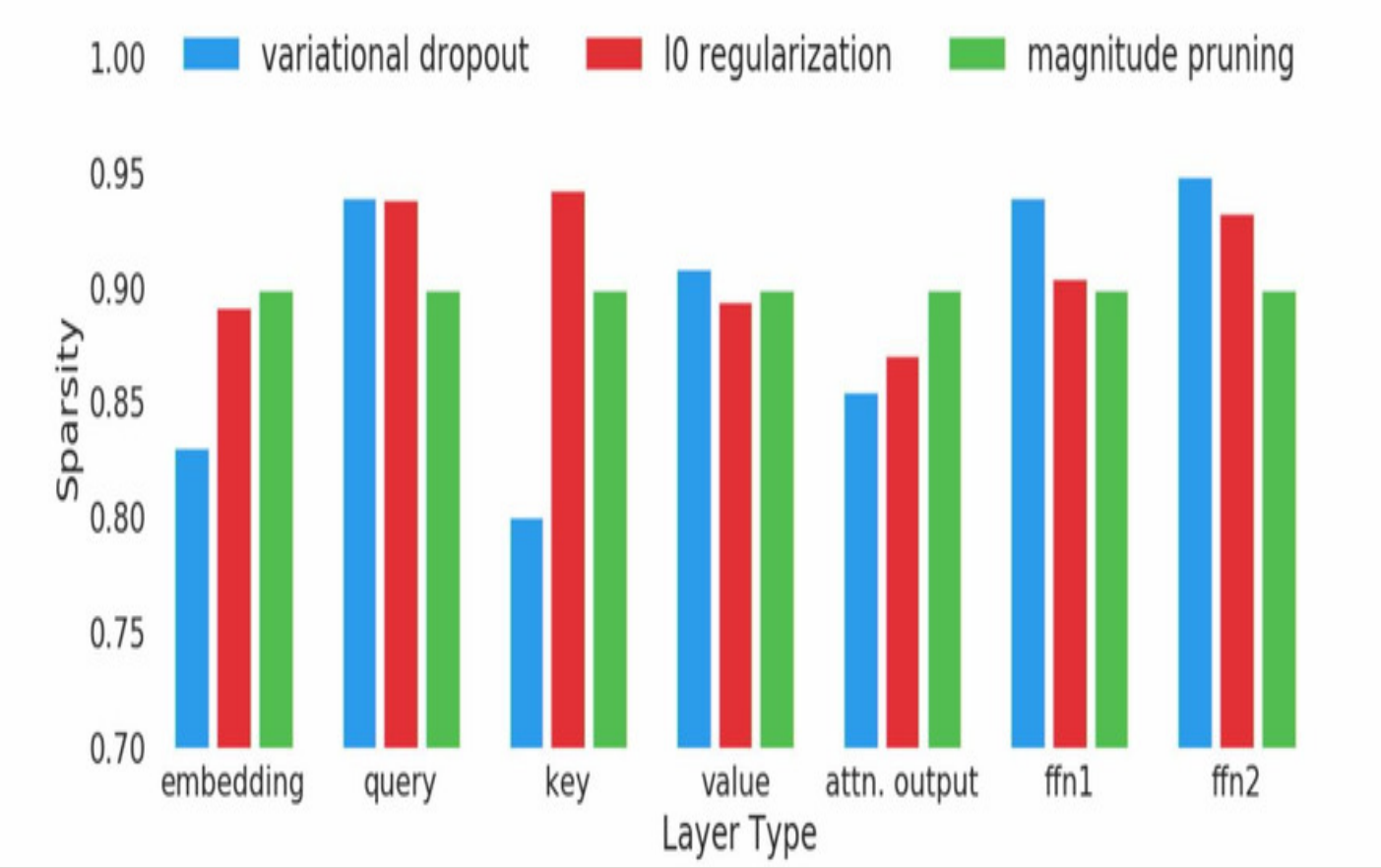


TRANSFORMER ON WMT14 EN-DE

Sparsity-BLEU trade-off curves for the Transformer.

- Top: Pareto frontiers for each of the four sparsification techniques applied to the Transformer.
- Bottom: All experimental results with each technique. Despite the diversity of approaches, the relative performance of all three techniques is remarkably consistent. Magnitude pruning notably outperforms more complex techniques for high levels of sparsity.
- Transformer with MP trains 1.24x and 1.65x faster than l0 and VD

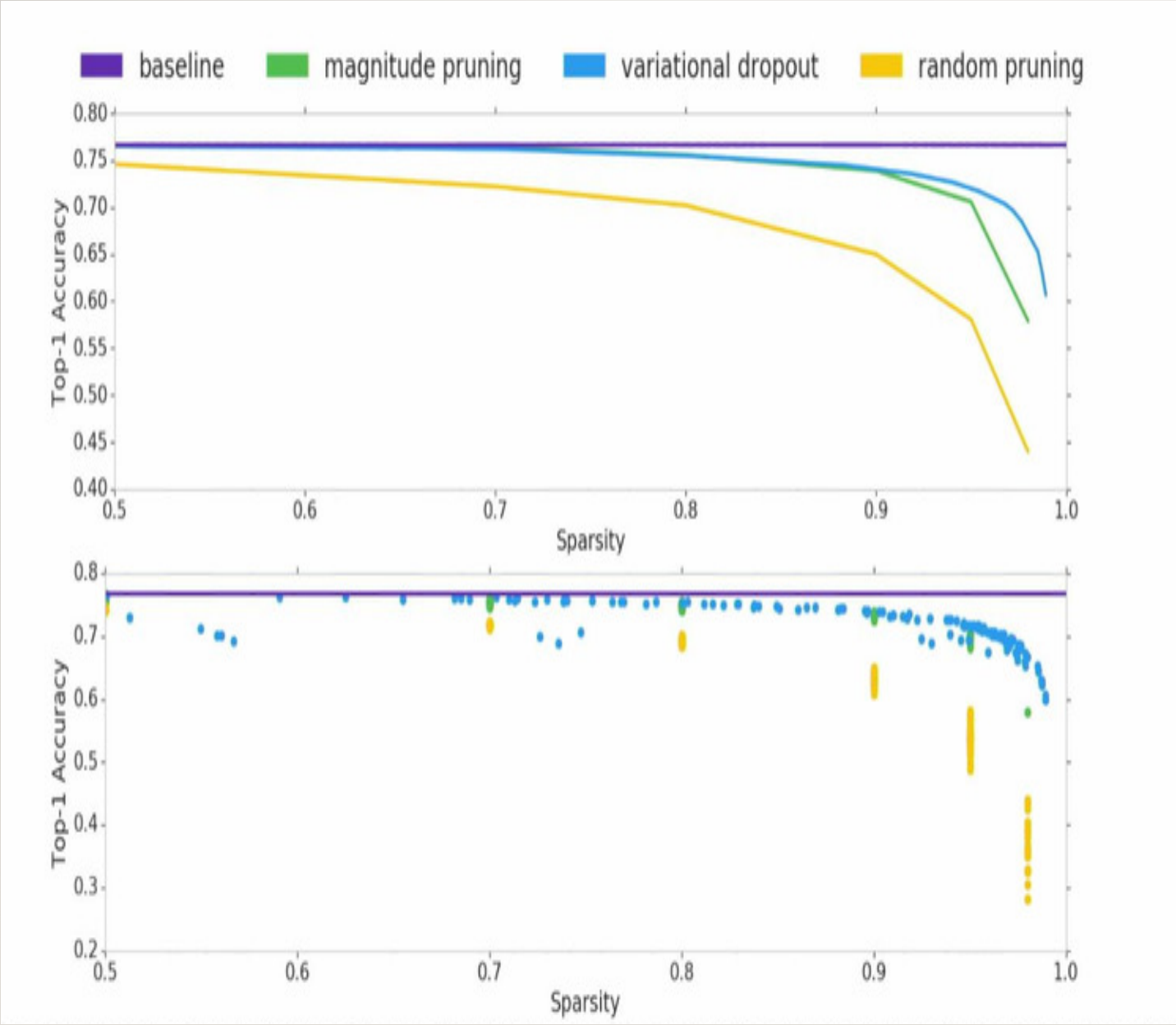
TRANSFORMER ON WMT14 EN-DE



Average sparsity in Transformer layers.

- Distributions calculated on the top performing model at 90% sparsity.
- The ability to learn non-uniform distributions of sparsity does not lead VD & l0 achieving superior performance.

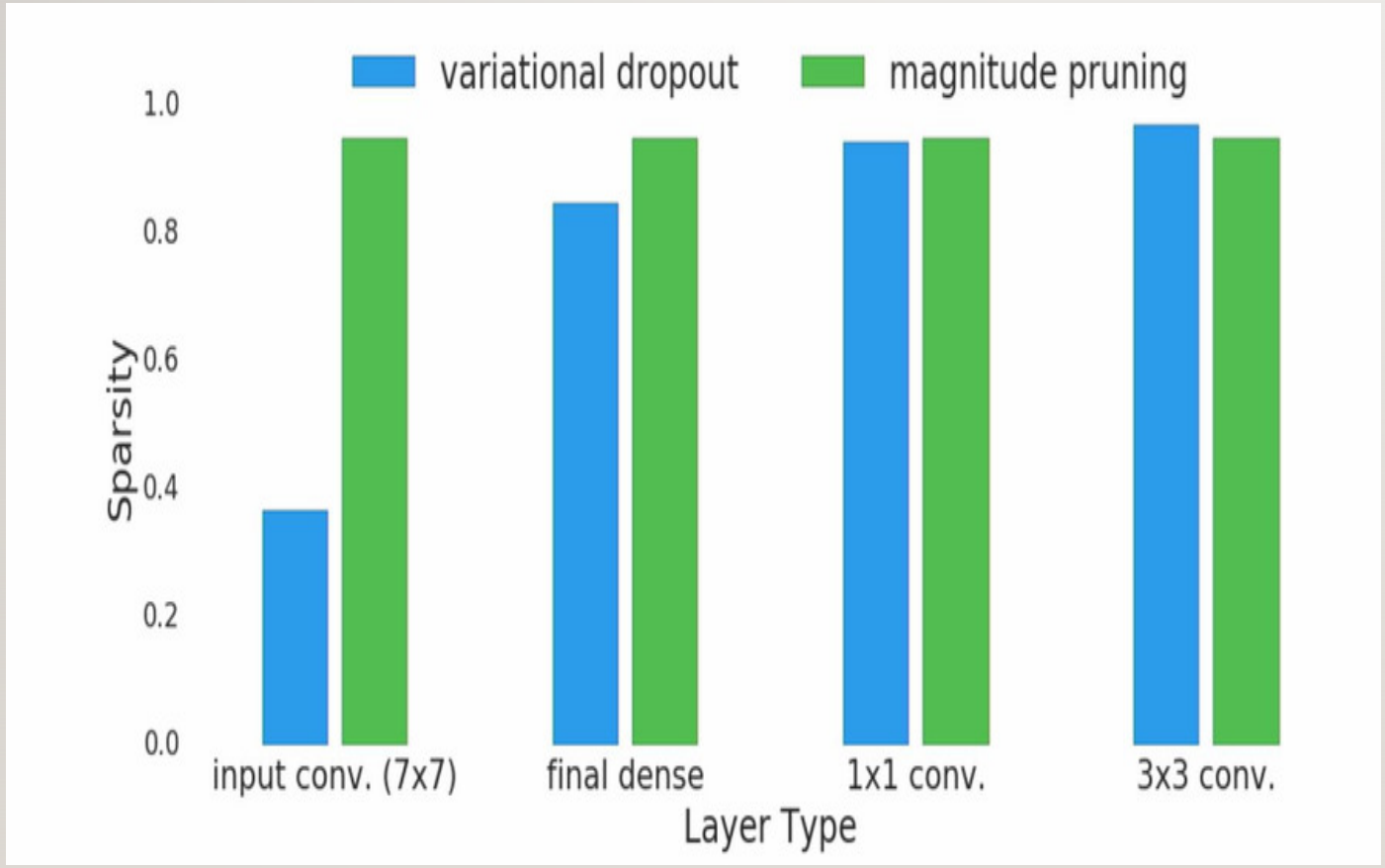
RESNET-50 ON IMAGENET



Sparsity-accuracy trade-off curves for ResNet-50

- Top: Pareto frontiers for variational dropout, magnitude pruning, and random pruning applied to ResNet-50.
- Bottom: All experimental results with each technique.
- We observe large variation in performance for variational dropout and l0 regularization between Transformer and ResNet-50
- Magnitude pruning and variational dropout achieve comparable performance for most sparsity levels, with variational dropout achieving the best results for high sparsity levels.

RESNET-50 ON IMAGENET



Average sparsity in ResNet-50 layers.

- Distributions calculated on the top performing model at 95% sparsity for each technique.
- Variational dropout is able to learn non-uniform disributions of sparsity, decreasing sparsity in the input and output layers that are known to be disproportionately important to model quality.

MAGNITUDE PRUNING V2

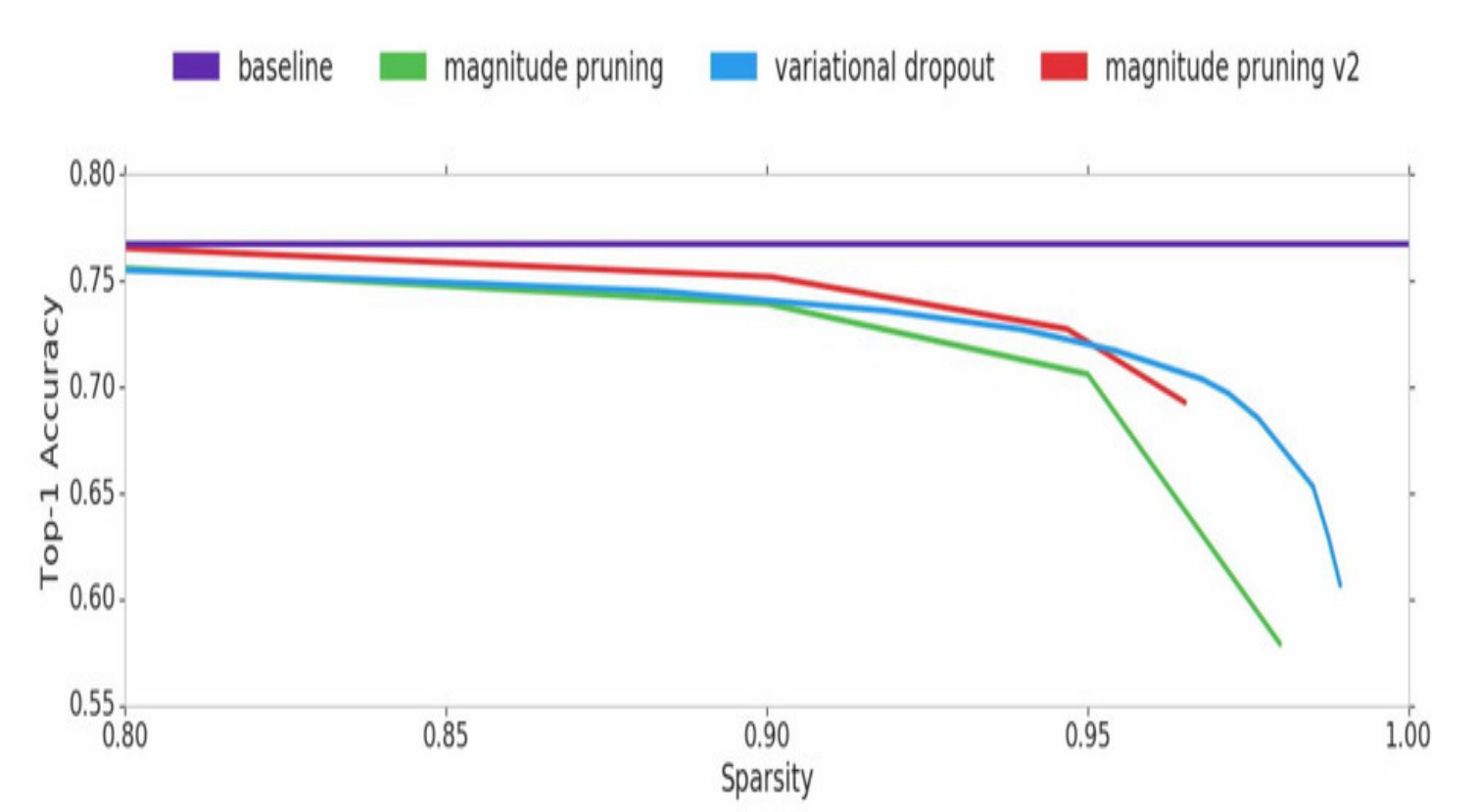
If we look at the distribution of sparsity across the layers, we saw that variational dropout learns to induce significally less sparsity in the first and last layer of the ResNet-50.

This make a lot of sense for this architecture, because the first and last layers in ResNet-50 have very few parameters, relatively to the rest of the model.

Given this importance of the distribution of sparsity, and the fact that we know that magnitude pruning can induce a user-specified level of sparsity, as well as the fact that magnitude pruning is significally more efficient that variational dropout

It is natural to wonder **how well magnitude pruning could perform if we were to distribute the non-zero weights more carefully and increase training time**.

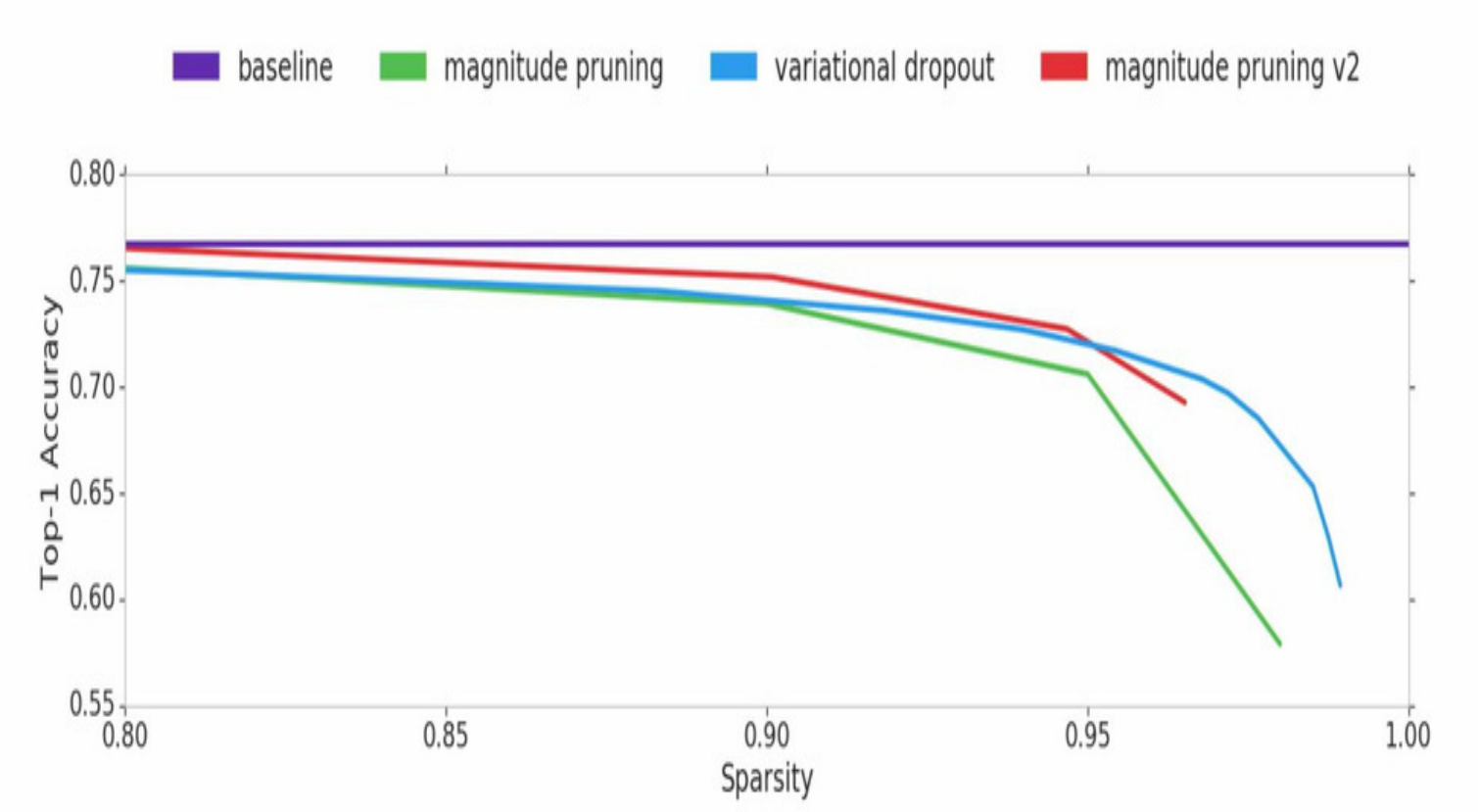
RESNET-50 ON IMAGENET



Sparsity-accuracy trade-off curves for ResNet-50 with modified magnitude pruning

- Altering the distribution of sparsity across the layers and increasing training steps by 1.5x yield significant improvement for magnitude pruning.
- We leave the first convolutional layer fully dense, and only prune the final fully-connected layer to 80% sparsity

SPARSIFICATION BENCHMARKS TAKEAWAYS



Sparsity-accuracy trade-off curves for ResNet-50 with modified magnitude pruning: 1.5x training time, 0.80% sparsity in the first and last layer

- Magnitude Pruning outperforms across all except the highest sparsity levels while using less resources than variational dropout
- On-par model sparsity and top-1 accuracy without extra complexity and computational requirements of CNN

LIMITATIONS OF THIS STUDY

Hyperparameter exploration.

- The number of possible settings vastly outnumbers the set of values that can be practically explored, and we cannot eliminate the possibility that some techniques significantly outperform others under settings we did not try.

Neural architectures and datasets.

- Transformer and ResNet-50 were chosen as benchmark tasks to represent a cross section of large-scale deep learning tasks with diverse architectures. We can't exclude the possibility that some techniques achieve consistently high performance across other architectures. More models and tasks should be thoroughly explored in future work.

GOAL OF THIS PROJECT

- Can learned sparse architectures be trained from scratch to the same test set accuracy as those trained with joint sparsification and optimization?

✓ NO, Training from scratch does not match the performaces of joint optimization & sparsification on large-scale tasks.

- **Do the top performing techniques in the literature actually perform the best on large-scale tasks?**

GOAL OF THIS PROJECT

- Can learned sparse architectures be trained from scratch to the same test set accuracy as those trained with joint sparsification and optimization?

✓ NO, Training from scratch does not match the performaces of joint optimization & sparsification on large-scale tasks.

- Do the top performing techniques in the literature actually perform the best on large-scale tasks?

✓ **No. Magnitude pruning performs on-par or better and requires less resources.**

THANKS FOR YOUR ATTENTION

G.Marco Nuzzarello
Walter Maltese
Pierandrea Morelli