Efficient Feature Matching Using Deep Neural Networks

Nicolas Filliol
MIT
Cambridge, MA 02139
nfilliol@mit.edu

Abstract

Feature detection and matching is a crucial task for many computer vision applications. Current deep-learning-based models require GPUs to support real-time inference. This work describes how pruning and quantization methods can be applied to SuperPoint and SuperGlue to reduce the number of parameters and effectively decreasing the model size and latency of the network.
https://github.com/nicofilliol/efficient-feature-matching

1 Introduction

Finding correct correspondences between points of different images is an essential step in many computer vision tasks such as Structure-from-Motion (SfM) or Simultaneous Localization and Mapping (SLAM). These correspondences are typically found by first detecting image keypoints and then matching their local features with the keypoints of the other image. To allow consistent and accurate feature matching, feature detection must be *repeatable*: the same keypoint must be detected in both images independently. Furthermore, the feature descriptor must be *reliable* and *distinctive*, so that for each keypoint, the corresponding one can be correctly recognized.

Traditionally, handcrafted feature detectors and descriptors such as SIFT (Lowe [1999]) and SURF (Bay et al. [2008]) have been used to perform this task. In recent years, neural networks have also been used to solve the problem of descriptor learning (Yi et al. [2016], DeTone et al. [2017]), often able to learn representations that are superior to the ones of handcrafted features. These features are then generally matched using a Nearest Neighbor (NN) search before being used to estimate the geometric transformation between the two images. Because NN search can be computationally very expensive, deep learning based approaches to feature matching such as SuperGlue by Sarlin et al. [2019] have become a viable alternative. Yet, latency of these models remain a problem as often GPUs are required for real-time inference. When considering potential applications in robotics, augmented and virtual reality, it is clear that there is a need for reducing memory footprint and latency of these models to enable real-time inference on weaker hardware resources, too. In this work, a state-of-the-art deep learning based feature detection model called SuperPoint proposed by DeTone et al. [2017] and a feature matching model called SuperGlue by Sarlin et al. [2019] are used to investigate potential methods for reducing the model size and speeding-up inference time with the ultimate goal to enable real-time feature detection and matching on CPU devices in the future.

2 Related Work

2.1 SuperPoint

SuperPoint (DeTone et al. [2017]) is a fully-convolutional model for feature detection and description. It operates on full-sized input images and outputs pixel-level keypoint locations and their associated fixed-length descriptors in a single forward pass.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

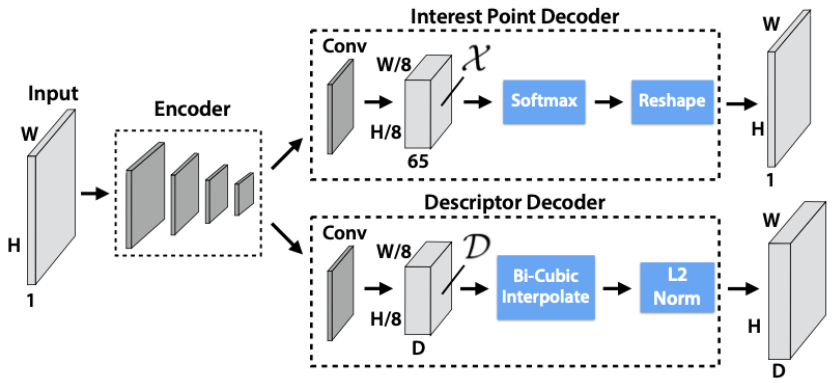


Figure 1: **SuperPoint Architecture.** The encoder reduces the dimensionality of the input and two separate decoders predict keypoint locations and descriptors. (DeTone et al. [2017])

The model (see Figure 2) has a single encoder consisting of eight 2D convolution layers with max-pooling layers to learn a lower dimensional representation of the input image. It maps the input image $I \in \mathbb{R}^{H \times W}$ to a tensor $\mathcal{B}^{H_C \times W_C \times F}$, where $H_C = H/8$ and $W_C = W/8$. It is then followed by two branches: an *interest point decoder* and a *descriptor decoder*. Both decoders contain two additional 2D convolution layers. The interest point detector first computes a tensor $\mathcal{X} \in \mathbb{R}^{H_C \times W_C \times 65}$ where H_C and W_C are the spatial output dimensions of the previous encoder. The 65 channels are used to represent a local, non-overlapping 8×8 grid of pixels plus an additional element used as a dustbin to symbolize a "no interest point". A channelwise softmax is then applied, the dustbin dimension removed and the tensor is reshaped to size $\mathbb{R}^{H \times W}$, now containing the probability of each pixel being an interest point. The descriptor head outputs a tensor $\mathcal{D}$ of size $\mathbb{R}^{H \times W \times D}$, where D is the fixed-length of the descriptor.

The final loss function to train the network is the sum of the interest point detector loss $\mathcal{L}_p$ and the descriptor loss $\mathcal{L}_d$. The model can be trained by using a pair of synthetically warped images given (a) pseudo-ground truth interest point locations and (b) the ground truth correspondence from a randomly generated homography that relates the two images. To generate the pseudo-ground truth interest point locations, DeTone et al. [2017] describe a method where they train a base detector called *MagicPoint* on a synthetic dataset consisting of simple geometric shapes with no ambiguity in the keypoint locations. In a next step, a process called *Homographic Adaption* is applied to enable self-supervised training of the detector and descriptor. By using MagicPoint, a set of pseudo-ground truth interest point location is generated for each image, after which random homographies are applied to each image. In theory, feature detection should be covariant to these homographies, but in practice a detector will not be covariant. By taking an empirical sum over all applied homographies, a new super-point detector can be trained.

2.2 SuperGlue

Sarlin et al. [2019] propose a neural network called SuperGlue, that matches two sets of local features by finding correspondences and rejecting non-matchable points. The matching can be written as a differentiable optimal transport problem and its costs are predicted by a Graph Neural Network (GNN). Inspired by the Transformer architecture, SuperGlue uses self-attention (intra-image) and cross-attention (inter-image) to take advantage of both spatial relationsips of interest points as well as their visual appearance.

Inputs to SuperGlue are two images A and B with their associated keypoint positions $\mathbf{p}$ and visual descriptors $\mathbf{d}$. Keypoint positions and descriptors can either be extracted by traditional algorithms such as SIFT or by a deep neural network like SuperPoint (see subsection 2.1). Assume image $\mathcal{A} := \{1, ..., M\}$ and $\mathcal{B} := \{1, ..., N\}$ have M and N local features, respectively. The problem of feature matching is now to find a partial assignment between these local features, and for better interpretability each correspondence should have a confidence value. For this, a partial soft assignment matrix $\mathbf{P} \in [0, 1]^{M \times N}$ is defined, where P_{mn} represents the confidence of keypoint m in $\mathcal{A}$ and n in $\mathcal{B}$ to be a match.

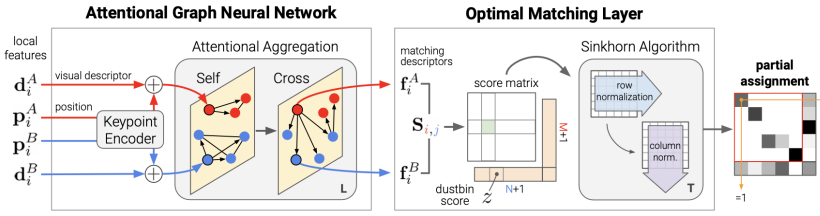


Figure 2: **SuperGlue Architecture.** SuperGlue is made up of an *attentional graph neural network* that predicts a more powerful representation $\mathbf{f}$ of local features $(\mathbf{p}, \mathbf{d})$. The following *optimal matching layer* finds the optimal partial assignment matrix $\mathbf{P}$ by using the differentiable Sinkhorn algorithm (Sarlin et al. [2019])

The attentional graph neural network starts with a keypoint encoder which combines its visual appearance $\mathbf{d}$ and its position $\mathbf{p}$ into a high-dimensional vector using a Multilayer Perceptron (MLP). The graph used for the GNN is given by a complete graph whose nodes are the keypoints of both images and uses features $(\mathbf{p}, \mathbf{d})$ as a starting point, and there are undirected edges between each node (intra-image edges and inter-image edges). A fixed number of layers L with different parameters are chained and alternatively perform a residual message passing update by aggregating along the self or cross edges. The messages used for aggregating and node feature update are computed using the attention mechanism, typically using multi-head attention to improve the expressivity. Using a final linear projection, the matching descriptors $\mathbf{f}$ are found.

The assignment $\mathbf{P}$ can now be found by computing a score matrix $\mathbf{S} \in \mathbb{R}^{\mathbf{M} \times \mathbf{N}}$ for all possible matches and then maximizing the total score $\sum_{i,j} \mathbf{S_{i,j}} \mathbf{P_{i,j}}$ with the constraint that the sum of each row and column in $\mathbf{P}$ must be smaller than one. The score is given by the inner product $\mathbf{S_{i,j}} = \langle \mathbf{f}_i^A, \mathbf{f}_j^B \rangle \; \forall (i,j) \in \mathcal{A} \times \mathcal{B}$, but to let the network suppress some keypoints (e.g. a match might no be present in the other image), the score matrix is augmented with a dustbin for both images by using a single learnable parameter: $\bar{\mathbf{S}}_{i,N+1} = \bar{\mathbf{S}}_{M+1,j} = \bar{\mathbf{S}}_{M+1,N+1} = z \in \mathbb{R}$. Since a keypoint in A is assigned to at most a single keypoint in B or the dustbin, but each dustbin can have as many matches as there are keypoints in the other set, we can write the constraint for the augmented partial assignment matrix as $\bar{\mathbf{P}} \mathbf{1_{N+1}} = \mathbf{a}$ and $\bar{\mathbf{P}}^T \mathbf{1_{M+1}} = \mathbf{b}$, where $\mathbf{1}$ denotes a vector of all ones and $\mathbf{a} = [\mathbf{1}_M^T \; N]^T$ and $\mathbf{b} = [\mathbf{1}_N^T \; M]^T$ describe the number of expected matches for each keypoint and dustbin. The solution of the optimization problem

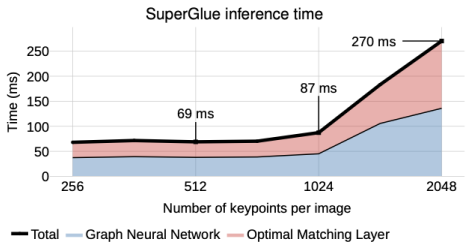
$$\max_{\mathbf{P}} \sum_{i,j} \bar{\mathbf{S}}_{i,j} \bar{\mathbf{P}}_{i,j} \tag{1}$$

can be efficiently solved with the differentiable Sinkhorn algorithm Cuturi [2013]. Because both the GNN and optimal matching layer are differentiable, backpropagation from matches to visual descriptors is possible. SuperGlue is trained in supervised manner from ground truth matches $\mathcal{M} = \{(i,j)\} \subset \mathcal{A} \times \mathcal{B}$ which can be generated by applying homographies. Including unmatched keypoints $\mathcal{I} \subseteq \mathcal{A}$ and $\mathcal{J} \subseteq \mathcal{B}$, the loss is given as:

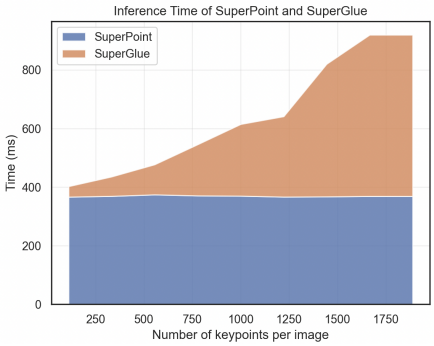
$$\mathcal{L} = -\sum_{(i,j) \in \mathcal{M}} log \bar{\mathbf{P}}_{i,j} - \sum_{i \in \mathcal{I}} log \bar{\mathbf{P}}_{i,N+1} - \sum_{j \in \mathcal{J}} log \bar{\mathbf{P}}_{M+1,j} \tag{2}$$

3 Methods

Sarlin et al. [2019] present a detailed analysis of inference time of their SuperGlue network (see Figure 3a). For 512 and 1024 keypoints per image, SuperGlue runs at 14.5 FPS and 11.5 FPS. However, performance on a CPU is strongly limited as experiments on a Apple M2 chip show (refer to Figure 3b). On top of that, we must also account for the inference time of the SuperPoint model that is required to detect the keypoints or alternatively a handcrafted feature detector like SIFT which are typically even slower (DeTone et al. [2017]). This turns out to be very significant for smaller number of keypoints. Even in the best case, the model can be used for inference at 2.5 FPS which is by far not fast enough for applications in robotics or mixed reality. From the results, we also see that the SuperGlue model with its 12M parameters is relatively big, and to make it more applicable for smaller edge devices, a reduction of the model size is desireable.



(a) **SuperGlue Inference Time.** Run-time of SuperGlue and its two major blocks on a NVIDIA GeForce GTX 1080 GPU across 500 runs. (Sarlin et al. [2019])



(b) **Inference Time on Apple M2.** Run-time of SuperPoint and SuperGlue.

Figure 3: **Latency Measurements.** Inference time for SuperGlue and SuperPoint on GPU (left) and CPU (right).

In order to improve latency and model size of SuperPoint and SuperGlue, two common methods have been tested: *pruning* (subsection 3.1) and *quantization* (subsection 3.2).

3.1 Pruning

There are two common approaches to pruning: *fine-grained* or *structured* pruning. The latter has the advantage, that it can lead to a direct speed-up on common general-purpose hardware, while for former method requires specialized hardware to gain speed-up. To determine layers that could potentially be pruned and with which pruning ratio, several sensitivity experiments have been conducted and are presented in Figure 4 and Figure 5 in Appendix A. The main conclusion that can be made from this simple experiment are: i) SuperPoint layers are very sensitive to pruning and ii) SuperGlue layers are very insensitive to pruning. This suggests that aggressive pruning in the SuperGlue model could be possible.

3.1.1 Fine-Grained Pruning

A fine-grained pruning approach has been applied to both the SuperPoint and SuperGlue model as described by Han et al. [2015b]. Based on a pretrained model, for each layer the smallest p percent of the weights (absolute value) are removed (set to zero), where p is the pruning ratio. After removing these unimportant connections, the model is trained for a couple more epochs, a process called finetuning. It must be noted, however, that unfortunately so far I wasn't able to setup a working training loop for the SuperPoint model which is compatible with the SuperGlue implementation although I tried very hard. That means that there was no way to finetune the SuperPoint parameters which are already very sensitive to pruning anyways. Finetuning for SuperGlue was performed for three epochs using a subset of the COCO2017 dataset Samet et al. [2020a].

3.1.2 Channel-Level Pruning

Because channel-level pruning (Mao et al. [2017]) can achieve direct speed-up also on common hardware, it is particularly interesting. SuperGlue contains hardly any convolutional layers, so channel-level pruning can only be applied to SuperPoint. First, the input channels for each layer's weight matrix are sorted by some importance metric (e.g. Frobenius norm) and then the $p \cdot \#\text{Input Channels}$ with least importance are removed. The according output channels of the previous layer must be pruned, too.

3.1.3 Results

In the following, the results of three experiments are presented: (i) aggressive fine-grained pruning of only the SuperGlue network, (ii) less aggressive pruning of both SuperPoint and SuperGlue, (iii) channel-based pruning for SuperPoint and fine-grained pruning for SuperGlue. Table 1 shows the effect of pruning on the number of (non-zero) parameters and the corresponding model size. With

very aggressive pruning for SuperGlue only, a total model size reduction of 5x can be achieved, but also for less aggressive combined pruning in both SuperPoint and SuperGlue, a significant model size reduction is achievable. Table 2 shows the evaluation results of the models after finetuning for three epochs on a smaller version of the COCO 2017 dataset (Samet et al. [2020b]). It shows precision and recall of the matches. Furthermore, homography estimation is performed using RANSAC and the mean reprojection error of the four corners of the image is computed and the area under the cumulative error curve (AUC) is reported. These metrics are of special interest, as they evaluate a downstream task. Precision and recall on the other hand are highly dependent on the number of keypoints predicted by SuperPoint. The pruned SuperPoint models (without finetuning) tend to predict fewer keypoints which might help to artificially increase precision and recall because only the high confident keypoints are left.

Table 1: Effect of Pruning on Parameters and Model Size

Model	**Params (M)**			**Model Size (MiB)**		
	SuperPoint	SuperGlue	Total	SuperPoint	SuperGlue	Total
Dense	1.3	12.02	13.32	4.96	45.87	50.83
Fine-grained (SuperGlue only)	1.3	1.33	2.63	4.96	5.08	10.04
Fine-grained (both)	1.18	3.71	4.89	4.48	14.16	18.64
Channel + Fine-grained	1.1	6.09	7.19	4.18	23.24	27.42

Table 2: Evaluation of Pruned Models

Model	**AUC@5**	**AUC@10**	**AUC@25**	**Precision**	**Recall**
Dense	32.02	49.09	68.94	78.64	91.67
Fine-grained (SuperGlue only)	28.77	48.92	71.04	77.07	85.95
Fine-grained (both)	33.19	50.71	70.75	80.34	90.67
Channel + Fine-grained	8.28	20.99	42.20	47.11	62.60

3.2 Quantization

Another common approach to decrease model size is quantization of the model. Instead of using 32-bit floating point numbers, the idea is to quantize the weigths to n-bit integers where n is typically of chosen to be 4, 8 or 16. To typical approaches to quantization are *k-means-based weight quantization* (Han et al. [2015a]) and *linear quantization* (Jacob et al. [2017]).

A linear quantization approach has been implemented based on Lab 02 and extended to support 1D convolution layers. However, linear quantization at this point was unsuccessful and results in zero precision and recall. The weight distribution of most layers is very narrow but with tails extending to large values (see Figure 6 in Appendix A). The high range of values that must be covered reduces resolution for smaller weight values significantly and the results show that that it doesn't suffice.

Instead, k-means-based quantization could be used to initialize more centroids around zero to increase resolution there. The tools for k-means-based quantization do not work with the SuperGlue model at this point of time because of its special structure inlcuding the optimal matching layer. Unfortunately, there was no more time to start with a custom implementation.

4 Conclusion

In recent years, deep-learning-based feature detectors and matchers have become more popular thanks to their superior performance compared to handcrafted algorithms in specific applications. However, these models remain computationally very intensive and GPUs are typically required to run them in real-time. This project demonstrates with the example of SuperPoint and SuperGlue that there is great

potential to reduce the number of parameters significantly, thus reducing model size and speeding-up the inference time on specialized hardware. The presented results show that aggressive pruning ratios are possible and with an improved training procedure, original performance could be recovered and a direct speed-up on common hardware could be achieved by applying higher channel-pruning ratios.

References

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014. Similarity Matching in Computer Vision and Multimedia.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances. 2013.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2017.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015a.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015b.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017.

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. IEEE, 1999.

Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J. Dally. Exploring the regularity of sparse structure in convolutional neural networks, 2017.

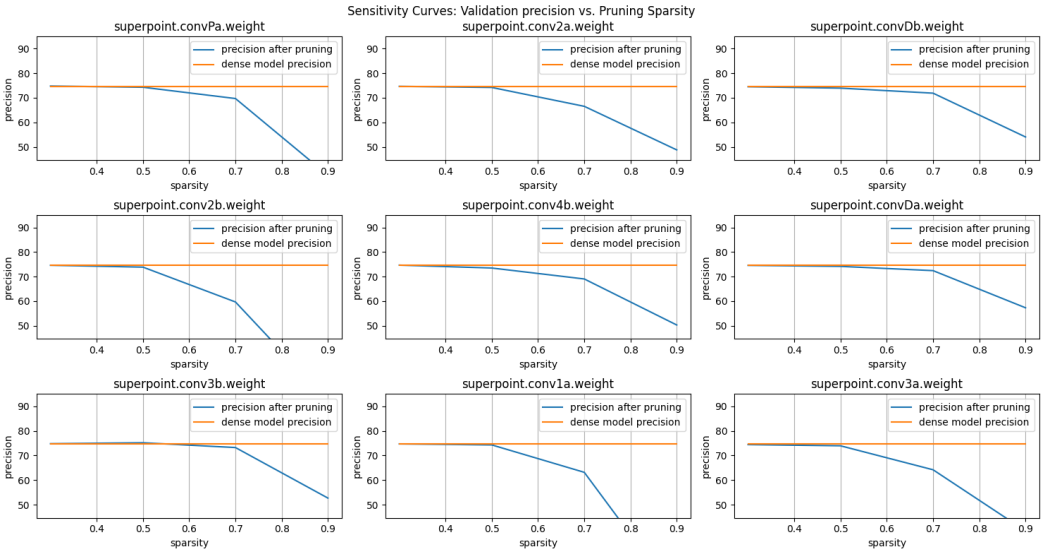
Nermin Samet, Samet Hicsonmez, and Emre Akbas. Houghnet: Integrating near and long-range evidence for bottom-up object detection. In *European Conference on Computer Vision (ECCV)*, 2020a.

Nermin Samet, Samet Hicsonmez, and Emre Akbas. Houghnet: Integrating near and long-range evidence for bottom-up object detection. In *European Conference on Computer Vision (ECCV)*, 2020b.

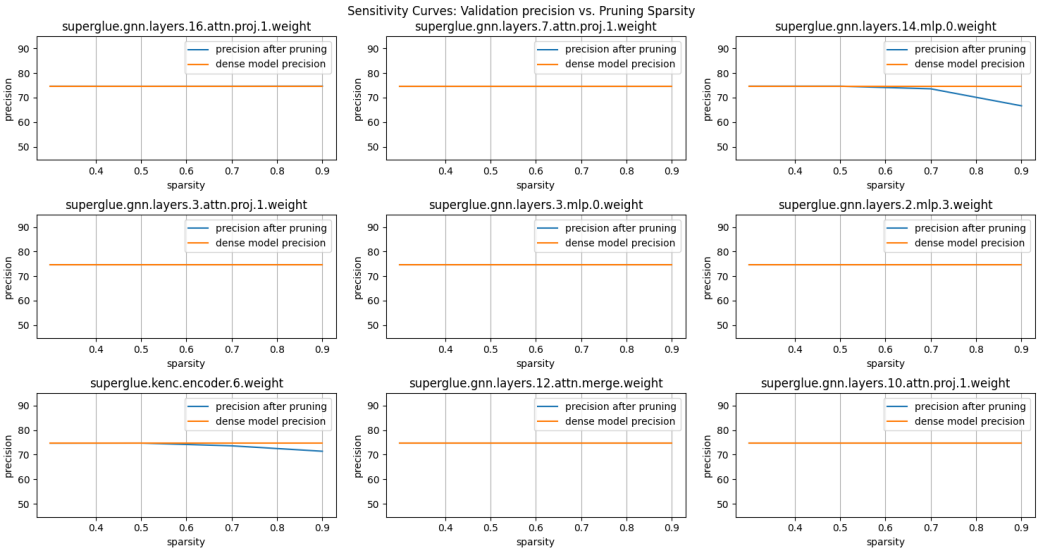
Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks, 2019.

Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform, 2016.

A Appendix

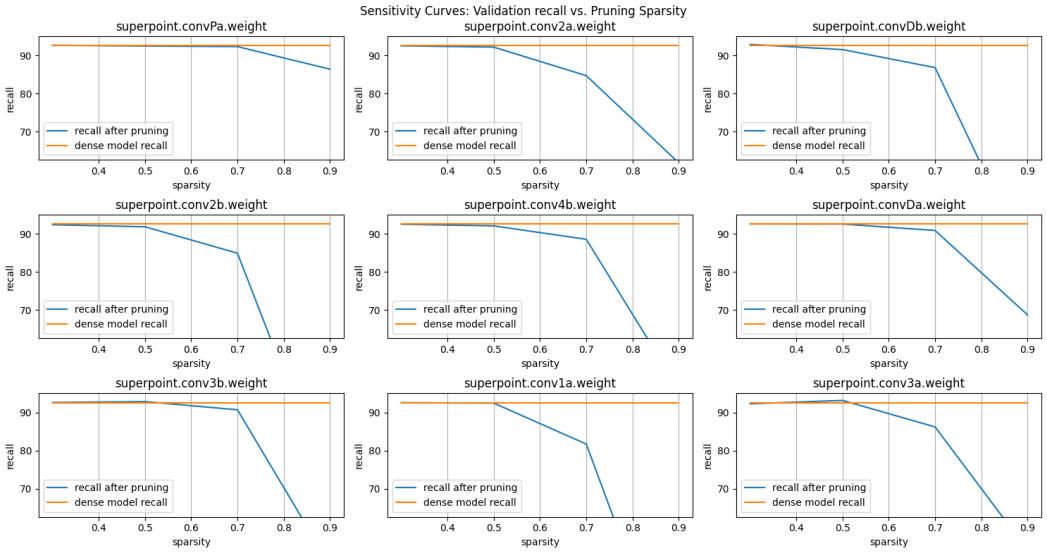


(a) SuperPoint

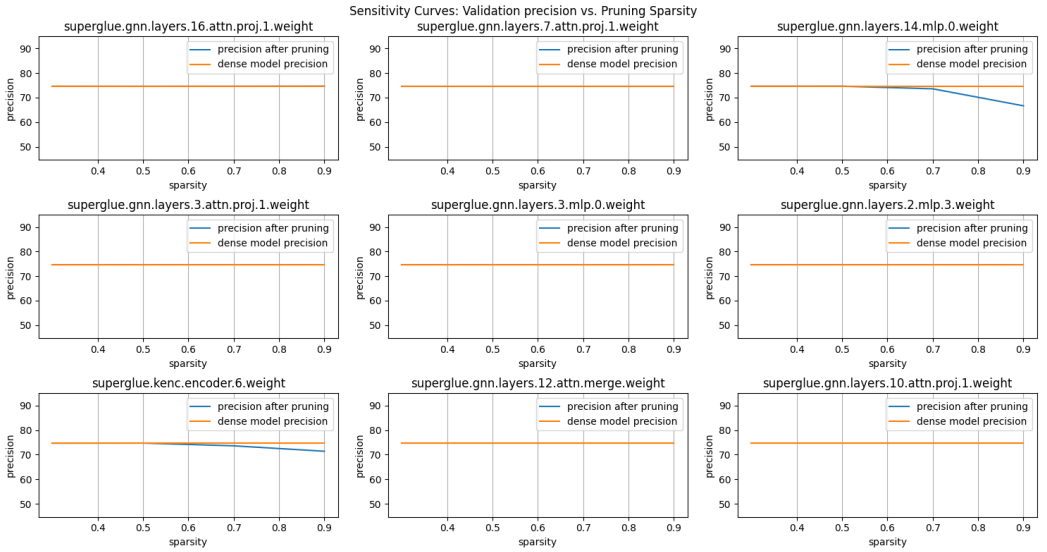


(b) SuperGlue
(Pruned curve is hidden below dense curve)

Figure 4: **Precision Sensitivity.** Influence on final precision when applying fine-grained pruning to individual layers.

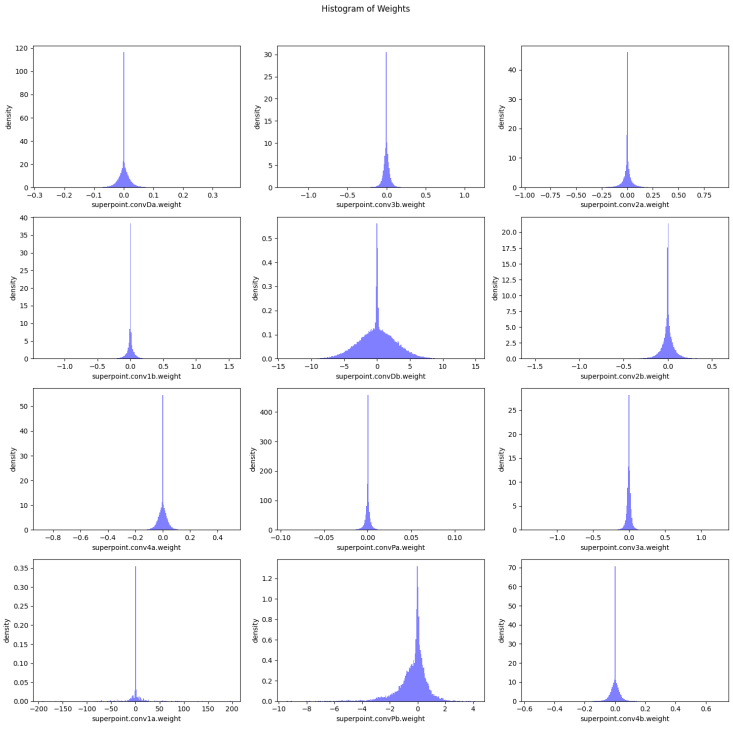


(a) SuperPoint

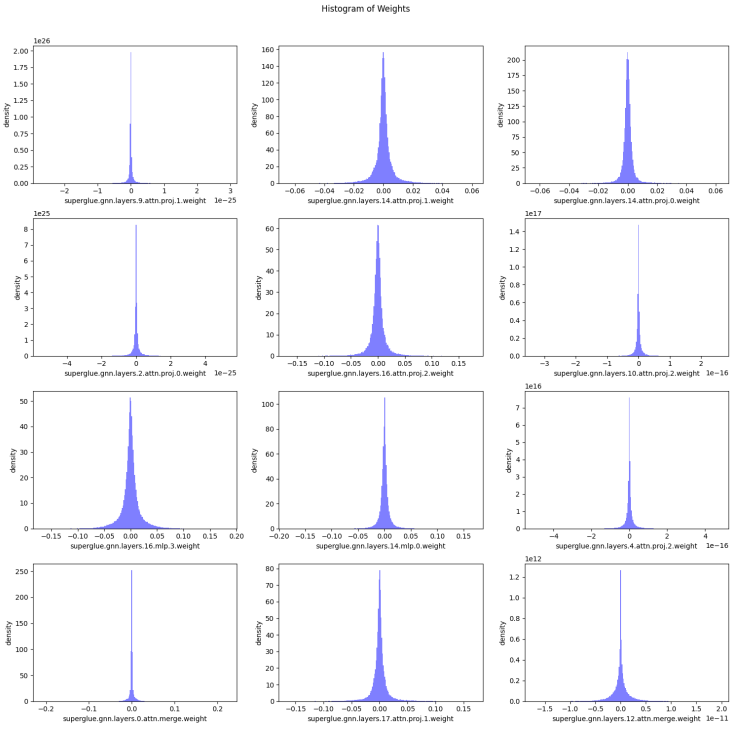


(b) SuperGlue
(Pruned curve is hidden below dense curve)

Figure 5: **Recall Sensitivity.** Influence on final recall when applying fine-grained pruning to individual layers.



(a) SuperPoint



(b) SuperGlue

Figure 6: **Weight Distribtuion.** Weight distribution in twelve randomly sampled layers.