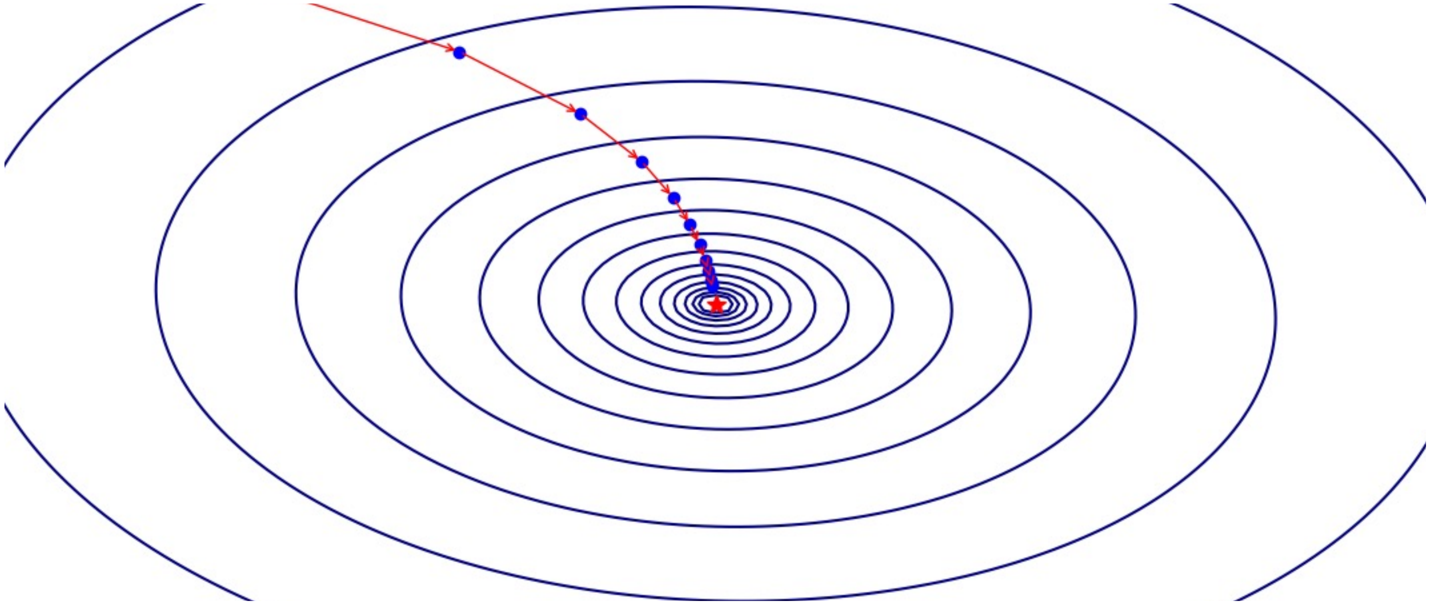


Advanced Topics in Deep Learning

Summer Semester 2024

8. Model Compression

17.06.2024

Prof. Dr. Vasileios Belagiannis

Chair of Multimedia Communications and Signal Processing

Friedrich-Alexander-Universität
Technische Fakultät

Course Topics

1. Interpretability.
2. Attention and Transformers.
3. Self-supervised Learning I.
4. Self-supervised Learning II.
5. Similarity Learning.
6. Generative Models.
7. **Model Compression.**
8. Transfer learning, domain adaptation, few-shot learning.
9. Uncertainty Estimation.
10. Geometric Deep Learning.
11. Recap and Q&A.

- The exam will be written.
- We will have an exam preparation test.

Acknowledgements

- Special thanks Arij Bouazizi, Julia Hornauer, Julian Wiederer, Adrian Holzbock and Youssef Dawoud for contributing to the lecture preparation.

Recap

- Generative models.
- Generative Adversarial Networks.
- Auto-Encoders.
- Variational Auto-Encoders.

Today's Agenda and Objectives

- Compression definition.
- Compression categories.
- Parameter quantitation.
- Parameter pruning.
- Data-free approaches.

Deep Neural Network Demands

- As the performance of the neural network increases, so does the number of model parameters.
  - This is a common trend for visual, audio and speech modalities.
- This leads to increased:
  - <u>Memory</u> and <u>computing</u> resources.
  - Execution <u>Time</u>.
  - <u>Energy</u> Consumption.
- Real-time is often only possible with high-performance <u>workstations</u>.

Model Compression

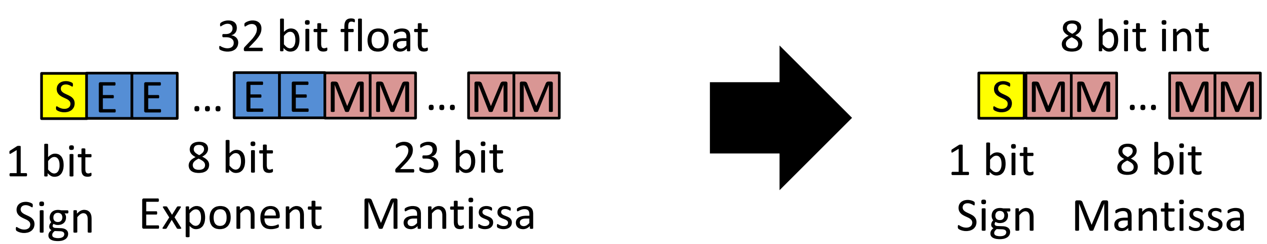
- Model compression simplifies the model in terms of parameter number and/or parameter size, while at the same time aiming not to reduce the model performance.
  - Lossless model compression retains the model performance.
  - Lossy model compression results in reduced model performance.
  - In some cases, the model performance can be improved after model compression.
- In deep learning, it is common to work with lossy approaches.
- Why can the model compression improve the results?

Main Compression Methodologies

- Parameter pruning.
  - It deals with the removal of redundant parameters.
- Parameter quantisation.
  - It deals with mapping a large (or infinite) set of continuous values to a smaller set of discrete finite values.
- Knowledge distillation.
  - It is the compression of a large model into a smaller model.
- Low-rank approximation / factorization.
  - It deals with the approximation of redundant parameters by a linear combination of a smaller set of parameters.
- Neural architecture search.
  - It deals with finding the model architecture that maximises the performance of the training set.

Parameter Quantisation

- The aim of the quantization is to reduce the precision of the weights of the model.
- For example, reducing the precision from 32-bit floating point to 8-bit integer lowers the computational effort and memory demand of the model by a factor of 4.
- However, quantization is an irreversible process. A quantized 8-bit model cannot be converted back to the original 32-bit model due to the loss of information.



Quantisation Precision

- Neural networks can be quantized to different degrees of precision, where precision is the bit width of the quantized model weights.
- It is common for neural networks to be trained and executed with 32-bit floating point precision. They can be pruned from 32-bit floating point to different levels of precision. Common approaches are:
  - 16-bit floating point: It does not normally reduce the performance of the model.
  - 8-bit integer: Performance loss is unavoidable.
  - 1-bit*: The weights of binary neural networks have a 1-bit precision.
  - Mixed-precision**: The layers in the neural network have different precisions regarding their influence on the output.

*Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations." Advances in neural information processing systems 28 (2015).
**Dong, Zhen, et al. "Hawq: Hessian aware quantization of neural networks with mixed-precision." CVPR 2019

Quantisation Precision (Cont.)

- Pruning methods can be grouped to:
  - Uniform and non-uniform quantization.
  - Symmetric and asymmetric quantization.
  - Layer-wise and channel-wise quantization.
- There are also different approaches based on the training protocol: quantization before, during or after training.
- Data-free quantisation does not require access to the training data.

Uniform and non-uniform quantization

- The real continuous values r are <u>mapped</u> to the discrete lower-precision domain Q (orange dots on the y-axis).
- For the uniform quantization*, we have:
  - The distances between the orange dots are the <u>same</u>.
  - $Q(r) = Int\left(\frac{r}{S}\right) - Z$, where S is a real-valued <u>scaling</u> factor and Z an integer zero point.
  - $S = \frac{\beta - \alpha}{2^b - 1}$, where $[\alpha, \beta]$ are the clipping range and b the bit width.
- For the non-uniform quantization*:
  - The distances between the orange dots can <u>vary</u>.

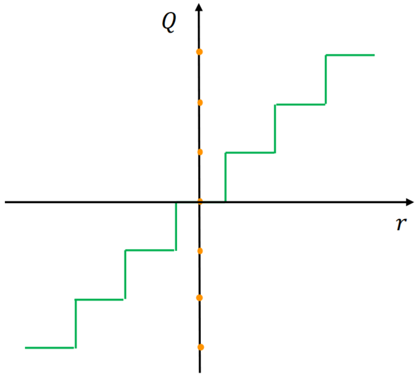


Image Source: https://arxiv.org/pdf/2103.13630.pdf

Uniform Quantization

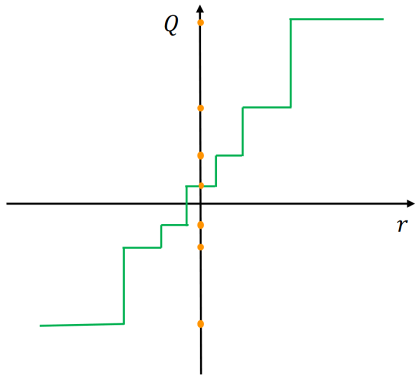


Image Source: https://arxiv.org/pdf/2103.13630.pdf

Non-Uniform Quantization

*Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." *arXiv preprint arXiv:2103.13630* (2021).

Symmetric and asymmetric quantization

- For the symmetric quantization*:
  - The clipping range $[\alpha, \beta]$ is equally distributed around zero.
  - $Z = 0$ and $\alpha = -\beta$.
- For the asymmetric quantization*:
  - The clipping range $[\alpha, \beta]$ is not equally distributed around zero and has to be determined by calibration.
  - The min and max value of the signal *r* can be used for the calibration. *r* could be the parameters of the layer to quantize.
  - $\alpha = r_{min}$ and $\beta = r_{max}$.

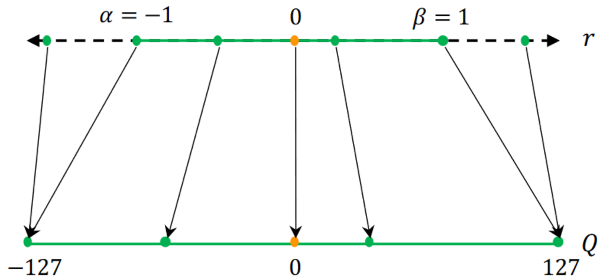


Image Source: https://arxiv.org/pdf/2103.13630.pdf

Symmetric Quantization

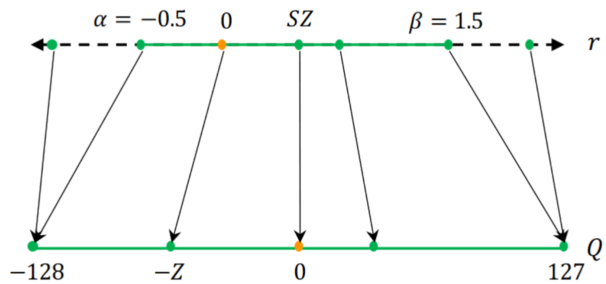


Image Source: https://arxiv.org/pdf/2103.13630.pdf

Asymmetric Quantization

*Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." arXiv preprint arXiv:2103.13630 (2021).

Layer-wise and channel-wise quantization

- Layer-wise quantization* uses a single clipping range for all filters in a layer.
- In channel-wise quantization, the clipping range is adjusted separately for each <u>channel</u>.
  - This results in <u>more</u> efficient use of the available bit width and less information loss during quantization.

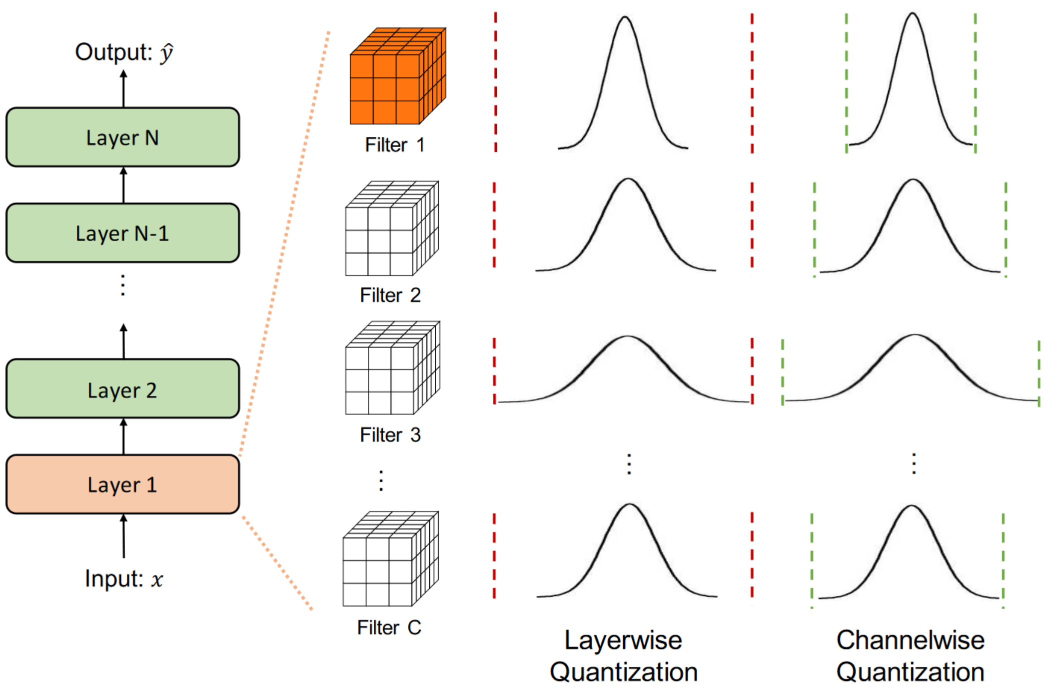
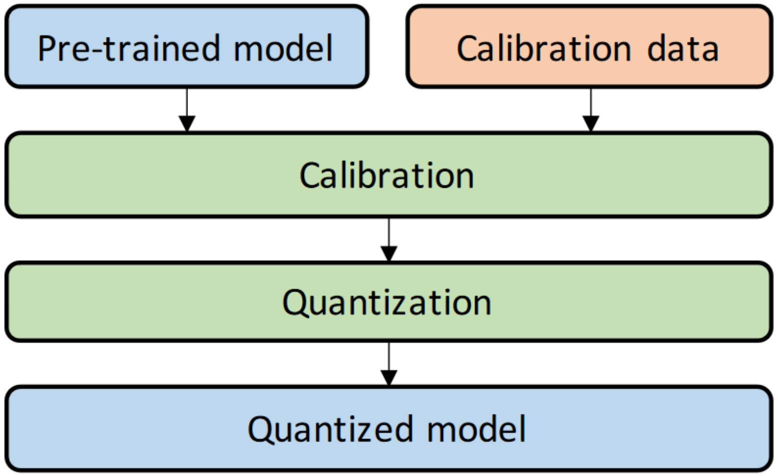


Image Source: https://arxiv.org/pdf/2103.13630.pdf

*Gholami, Amir, et al. "A survey of quantization methods for efficient neural network inference." *arXiv preprint arXiv:2103.13630* (2021).

Training-based Quantization Strategies (Cont.)

- Post-Training Quantization*: The model is quantized without fine-tuning. This reduces the needed data, which are only necessary for the calibration.



*https://medium.com/mlearning-ai/master-the-art-of-quantization-a-practical-guide-e74d7aad24f9#ecaa

Post-Training Quantization

- Post-Training Quantization (PTQ) does <u>not</u> apply fine-tuning to model parameters and activations, reducing computational and time overhead.
- The model, which is quantized with PTQ, <u>forgets</u> knowledge due to the reduced bit width (lower bit width → lower accuracy).
- PTQ requires only a <u>subset</u> of the training dataset and can even work with unlabelled data during the calibration. In the calibration the clipping range $[\alpha, \beta]$, the scaling factor S, and the zero offset Z are calculated.
- Models quantized with PTQ have often a <u>lower</u> accuracy compared to models quantized with quantization-aware training. Whereas PTQ has a <u>minimal</u> computational overhead.

Post-Training Quantization (Cont.)

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- Linear quantization takes the min/max value as α/β → This lead to high quantization error.
- The clipping defines α/β and therefore ignores outliers.
- Outlier Channel Splitting (OCS)* doubles a predefined number of channels to involve outliers in the quantization. Later the split channels are added and multiplied by 0.5. The weights are split by the following equation: $OCS(w) = \binom{(w-0.5)/2}{(w+0.5)/2}$.

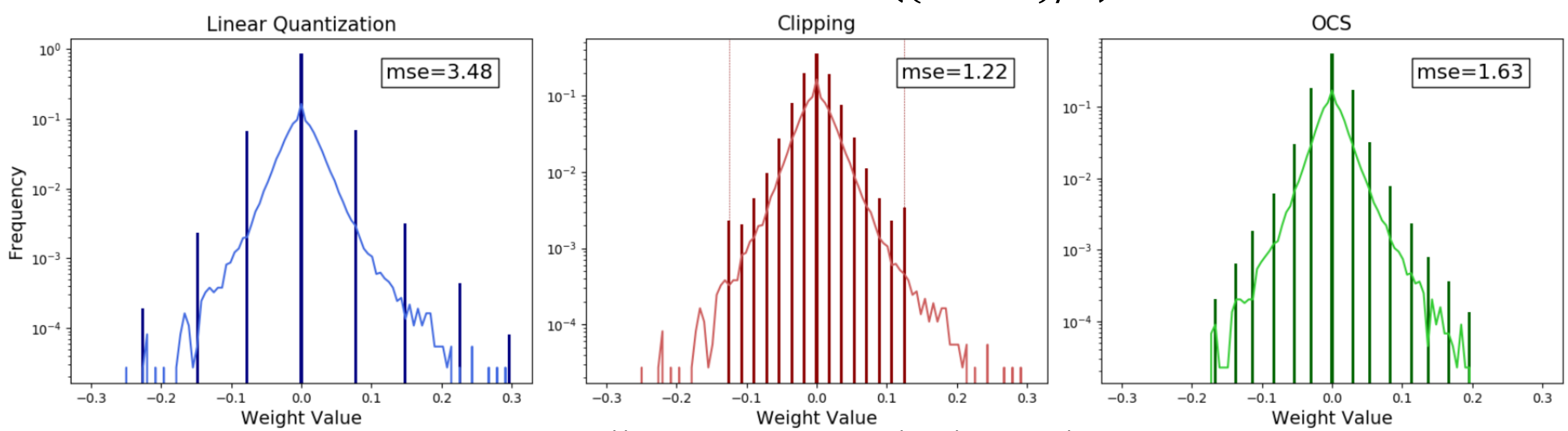
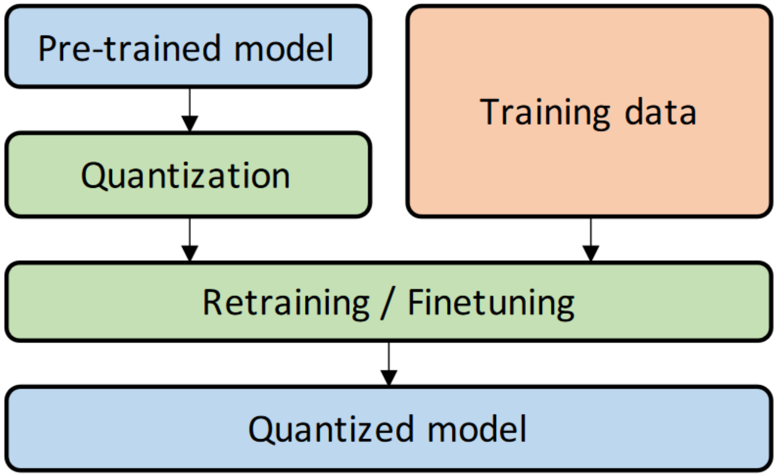


Image Source: https://proceedings.mlr.press/v97/zhao19c/zhao19c.pdf

Zhao, Ritchie, et al. "Improving neural network quantization without retraining using outlier channel splitting." ICML 2019.

Training-based Quantization Strategies

- Quantization-Aware Training*: After quantization, the model is fine-tuned with the training data and the model regains the loss of knowledge.



*Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Quantization-Aware Training

- During PTQ, the accuracy drops due to the reduced weight accuracy. A solution to recover the lost knowledge is to fine-tune the quantized model.
- The Quantization-Aware Training (QAT) needs access to the training dataset.
- Standard training is not possible for quantized models in integer precision because of the non-differentiable quantization operator.
- QAT uses floating point precision for the forward and backward passes to perform the weight update. After the weight update, the weights are pseudo-quantized. This means that they are represented by a floating point number but have the value of a quantized weight. The forward path is performed with the pseudo-quantized weights.

Quantization-Aware Training (Cont.)

Gradients can be approximated using the Straight Through Estimator (STE)*. The forward pass is performed with the quantized model. For the backward pass, the rounding operator is approximated with an identity function.

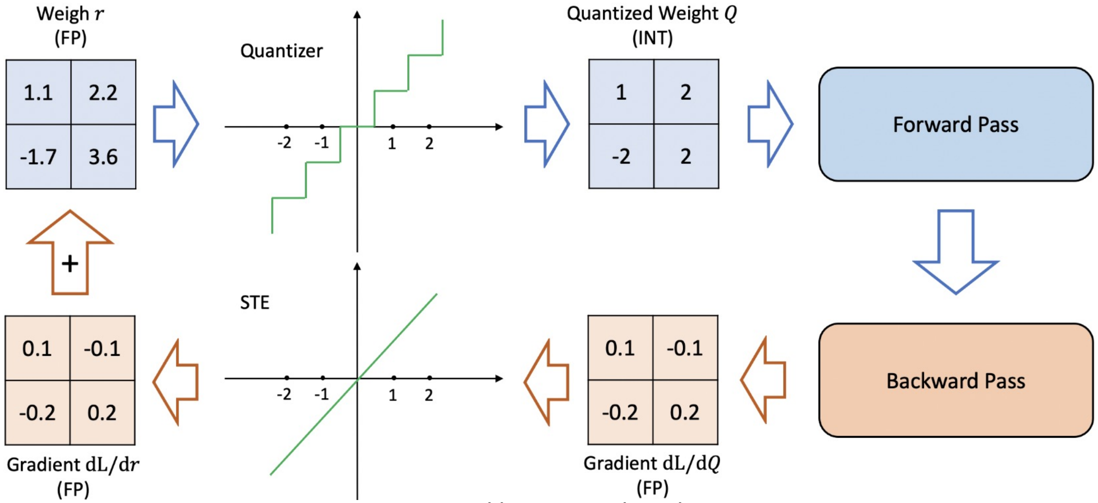


Image Source: https://arxiv.org/pdf/2103.13630.pdf

*Bengio, Yoshua, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation." *arXiv preprint arXiv:1308.3432* (2013).

Data-Free Quantization

- Training data is <u>not</u> always available to determine the clipping range or to fine-tune the quantised model.
- Real data is replaced by synthetic data which can be generated in different ways:
  - Generative Adversarial Networks (GAN)*: The pre-trained model is used as a discriminator to train the GAN to produce images that can be classified by the pre-trained model.
  - Batch Normalisation Statistics**: Backpropagation directly on a noise image using the stored mean and variance of the batch normalisation layers to calculate a loss.
- The advantage of the batch normalisation statistics approach over the GAN method is that the distribution of the real training data set is taken into account in the synthetic images.

*Li, Bowen, et al. "Dfqf: Data free quantization-aware fine-tuning." Asian Conference on Machine Learning. PMLR, 2020.
**Cai, Yaohui, et al. "Zeroq: A novel zero shot quantization framework." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

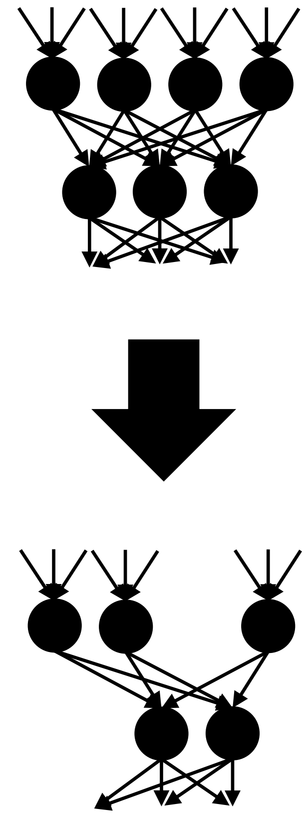
Data-Free Quantization (Cont.)

- ZeroQ* is a data-free mixed-precision quantization approach.
- The batch normalization Statistics are used to generate synthetic images by minimizing the following condition:
  - $\min_{x^r} \sum_{i=0}^{L} ||\tilde{\mu}_i^r - \mu_i||_2^2 + ||\tilde{\sigma}_i^r - \sigma_i||_2^2$
  - where x^r is the synthetic image and $\tilde{\mu}_i^r / \tilde{\sigma}_i^r$ are the mean and standard deviation of the synthetic image, while μ_i / σ_i are stored in the batch normalization.
- The precision of each layer is determined by using a <u>sensitivity</u> measure, where layers with a high sensitivity have a higher precision than layers with a low sensitivity.
- The sensitivity for a layer is calculated with the Kullback-Leibler divergence between the original model and the model where the specific layer is quantized with the desired precision.

*Cai, Yaohui, et al. "Zeroq: A novel zero shot quantization framework." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

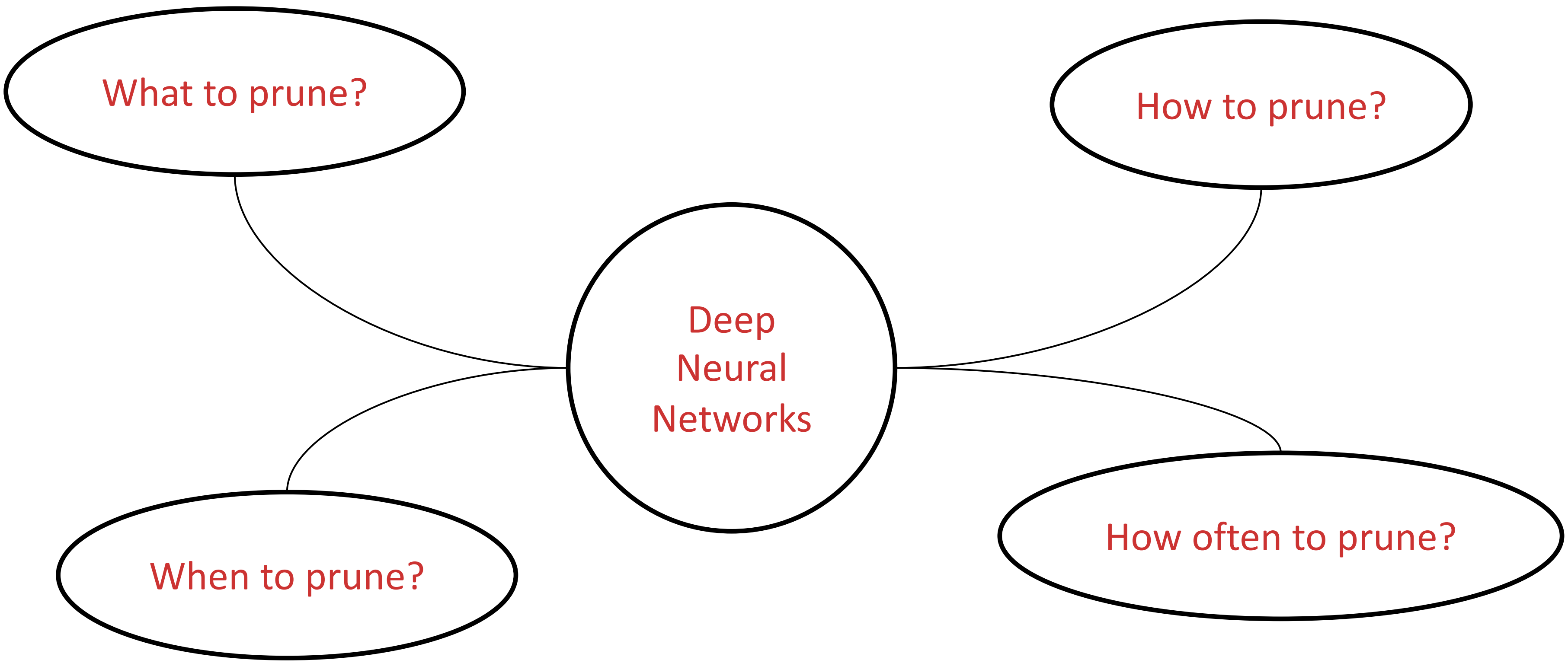
Parameter Pruning

- Neural networks are over-parameterised and some parameters are unnecessary or redundant.
- The goal of pruning is to <u>reduce</u> the number of model parameters in the neural network by removing the redundant or unnecessary parameters.
- The pruning affects the model performance and knowledge can be <u>forgotten</u>.
- Pruning a specific layer also influences the following layers.



Pruning Categories

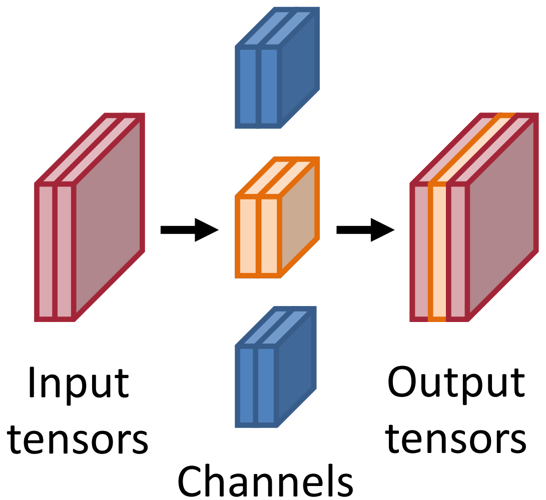
- Pruning can be grouped into different categories too.



Structured and Unstructured Pruning

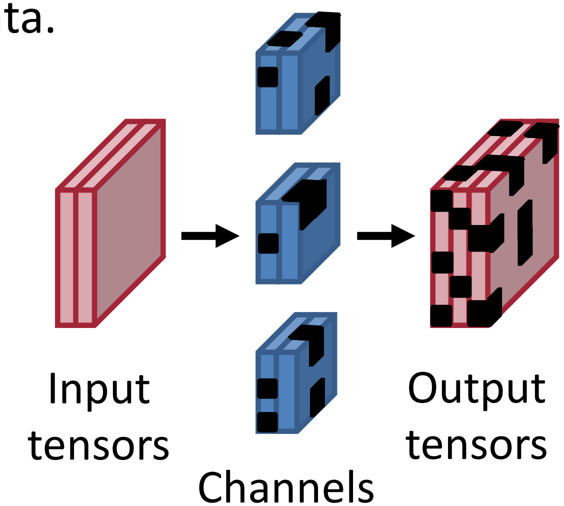
Structured pruning*:

- It removes whole channels from the neural network (also removes the connected structures).
- It reduces the execution time on standard hardware because of the fewer kernels.



Unstructured Pruning**:

- It removes single (sparse) weights from the neural network (removes unconnected structures).
- It is difficult to optimise on standard hardware because of the scattered nature of the computations. It also requires special libraries for sparse data.

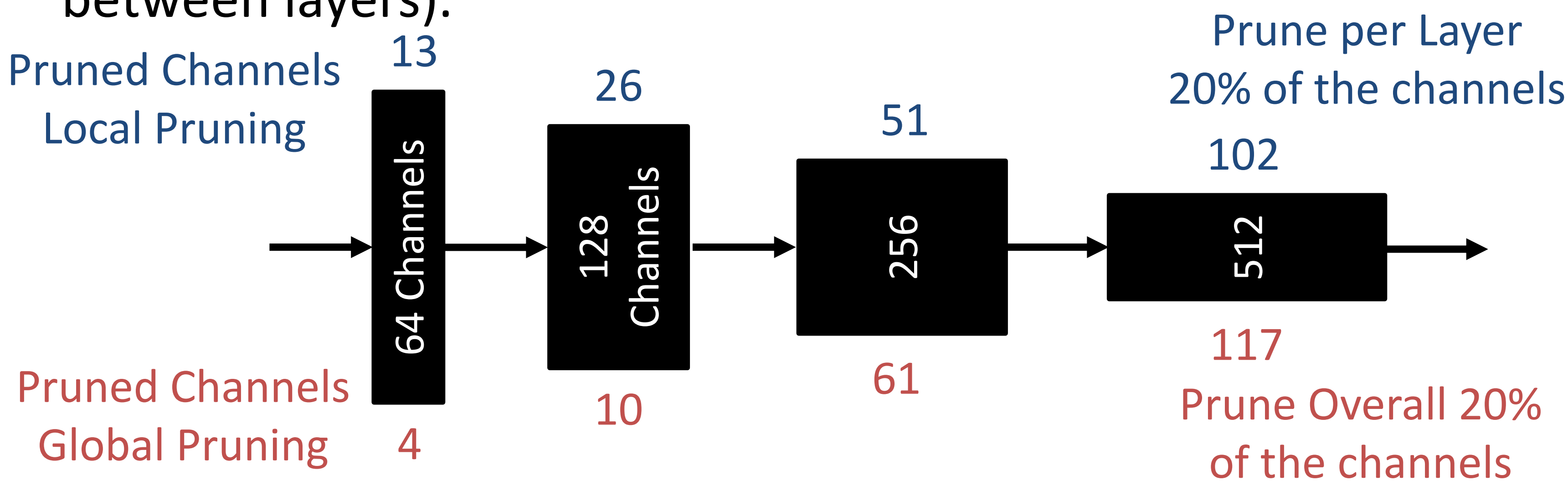


*Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).
**Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems 28 (2015).

Local and Global Pruning

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- Local Pruning*: The approach removes the same amount of parameters in each layer of the neural network (e.g. 20% of channels in each layer).
- Global Pruning*: The approach deletes the desired number of parameters across the network (the pruning ratio may differ between layers).



*https://towardsdatascience.com/neural-network-pruning-101-af816aaea61

Magnitude-based and Gradient-based Pruning

- Magnitude-based Pruning.
  - Magnitude-based pruning assumes that larger weights have a higher influence on the overall network output. Therefore, bigger weights are pruned less than smaller weights. A magnitude-based method is the L1 pruning*, where for each channel the sum of the weights is calculated. The channels with smaller sums are removed during the pruning.
- Gradient-based Pruning.
  - Gradient-based pruning uses gradients to decide which parameters to prune. Liu and Wu** compute gradients using the training data set and the loss function. They use the mean gradients of a feature map to decide whether the corresponding channel should be pruned. Channels with low gradients are pruned.

*Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).
**Liu, Congcong, and Huaming Wu. "Channel pruning based on mean gradient for accelerating convolutional neural networks." Signal Processing 156 (2019): 84-91.

Learning-based and Information-based Pruning

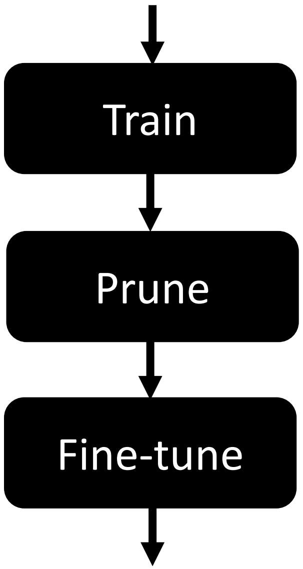
- Learning-based Pruning.
  - In learning-based pruning approaches, the importance of the parameters is learned. An example of a learning-based approach is GDP*, which uses a gating function to decide whether a channel should be pruned. During training, the gating function is pushed to 0 or 1 for different channels, and the channels with a 0 gate can be removed after training.
- Information-based Pruning.
  - An example for information-based pruning is HRank**. In Hrank, for each output feature map the matrix rank is calculated. It is claimed that the feature map is important if it has a high rank and thus remove the weights corresponding to feature maps with a low rank.

*Guo, Yi, et al. "Gdp: Stabilized neural network pruning via gates with differentiable polarization.", ICCV 2021.
** Lin, Mingbao, et al. "Hrank: Filter pruning using high-rank feature map.", CVPR 2020.

One-shot and Iterative Pruning

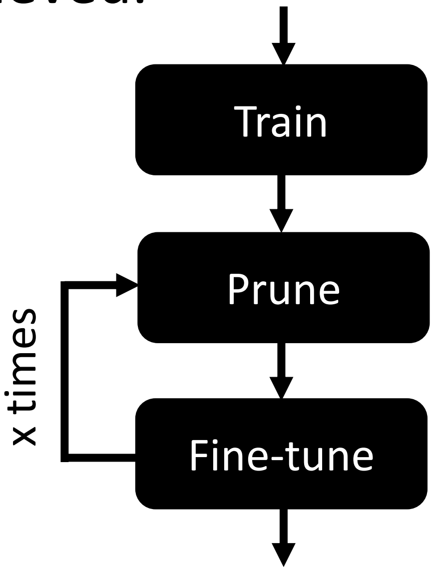
One-shot Pruning*:

The desired pruning sparsity is achieved in one step and all parameters to be pruned are removed in one step.



Iterative Pruning*:

Iterative pruning removes a small fraction of the parameters in each iteration until the desired pruning sparsity is achieved.



*https://roberttlange.com/posts/2020/06/lottery-ticket-hypothesis/

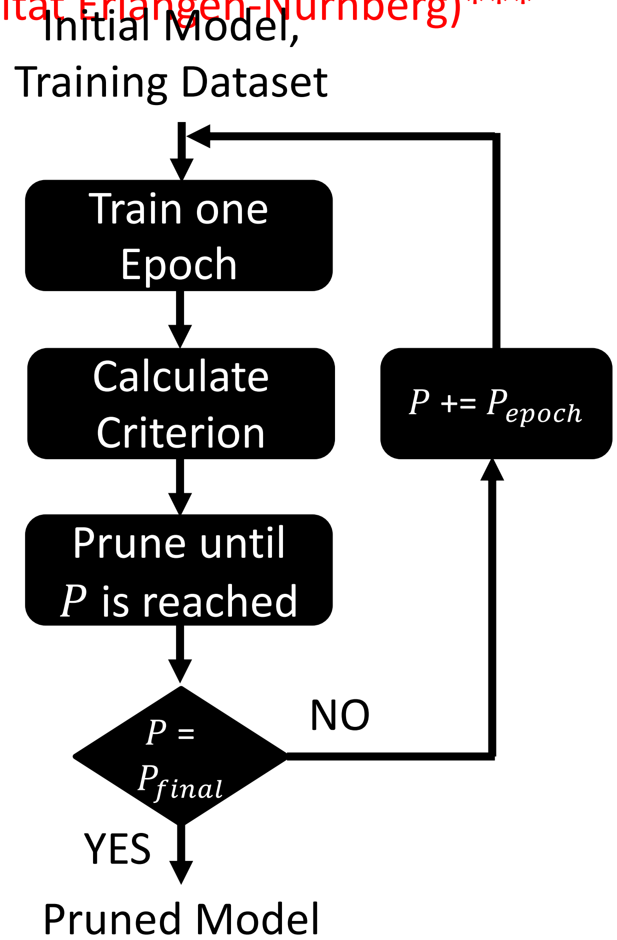
Pruning before Training

- Pruning before training results in reduced computational complexity because pre-training can be skipped and only the pruned network is trained.
- SnIP* prunes a randomly initialized network in an unstructured way by calculating a sensitivity for each weight w with the gradients g.
  - It propagates one batch D trough the randomly initialized network and calculate the loss to get the gradients g. The sensitivity s_j for the j weight is then calculated as follows:
  - $s_j = \frac{|g_j(\boldsymbol{w};D)|}{\sum_{k=1}^{m} |g_k(\boldsymbol{w};D)|}$,
  - where m is the total number of parameters.
  - The approach assumes that if the <u>gradient</u> has a <u>high absolute value</u> it has a high importance on the overall result and should be kept.
  - To get the pruned model the weights with the smallest sensitivity are removed from the model.

* Lee, Namhoon, Thalaiyasingam Ajanthan, and Philip HS Torr. "Snip: Single-shot network pruning based on connection sensitivity." *arXiv preprint arXiv:1810.02340* (2018).

Pruning during Training

- The network parameters are removed iteratively during the neural network training*.
- The pruning rate P is increased every epoch by P_{epoch} until the final pruning rate P_{final} is reached.
- It is used as pruning criterion the L1 norm**.

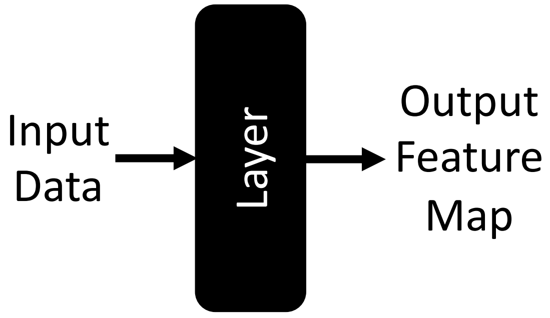


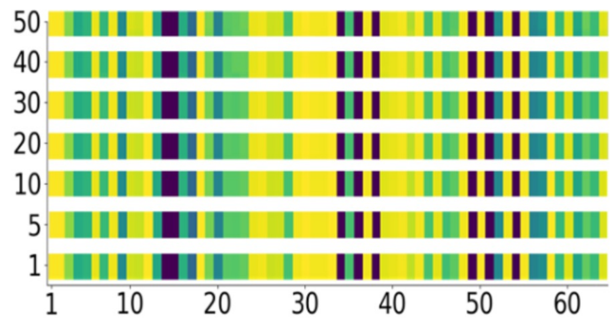
* Roy, Sourjya, et al. "Pruning filters while training for efficiently optimizing deep learning networks." 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020.
**Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).

Pruning after Training

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- The parameters are pruned after the pre-training and then the pruned model is fine-tuned. HRank* is this kind of approach.
  - It considers not only the internal knowledge of the model, but also the input data during pruning.
  - HRank defines the importance of a channel according to the matrix rank of the output feature map.
  - The rank is calculated with the Singular Value Decomposition (SVD)**.
  - The mini-Batch for the rank calculation of each channel is enough because of low variance.
  - Finally, the channels with a low rank are removed.





X-axis: channel, Y-axis: number of batches (128 samples/batch). Color represents the rank size.
Image Source: https://arxiv.org/pdf/2002.10179.pdf

* Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. CVPR 2020.
**Wall, Michael E., Andreas Rechtsteiner, and Luis M. Rocha. "Singular value decomposition and principal component analysis." A practical approach to microarray data analysis (2003): 91-109.

Data-free Parameter Pruning

- Data-free pruning approaches, like DFNP* do not need access to the training dataset for the fine-tuning.
  – DFNP generates synthetic data with the help of a generator network and uses the generated data in a Student-Teacher method to transfer knowledge from the unpruned to the pruned model.
  – The generator is trained by using the unpruned model as a discriminator based on adversarial training.
- When is a data-free approach meaningful?

*Tang, Jialiang, et al. "Data-free network pruning for model compression." *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021.

Study Material

- *Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. CVPR 2020.*
- *Li, Hao, et al. "Pruning filters for efficient convnets." arXiv preprint arXiv:1608.08710 (2016).*
- *Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.*
- *https://roberttlange.com/posts/2020/06/lottery-ticket-hypothesis/*

Next Lecture

Few-shot Learning