Where Should We Begin? A Low-Level Exploration of Weight Initialization Impact on Quantized Behaviour of Deep Neural Networks

Stone Yun
University of Waterloo
Alexander Wong
University of Waterloo
Email: {s22yun, a28wong}@uwaterloo.ca

Abstract

With the proliferation of deep convolutional neural network (CNN) algorithms for mobile processing, limited precision quantization has become an essential tool for CNN efficiency. Consequently, various works have sought to design fixed precision quantization algorithms and quantization-focused optimization techniques that minimize quantization induced performance degradation. However, there is little concrete understanding of how various CNN design decisions/best practices affect quantized inference behaviour. Weight initialization strategies are often associated with solving issues such as vanishing/exploding gradients but an often-overlooked aspect is their impact on the final trained distributions of each layer. We present an in-depth, fine-grained ablation study of the effect of different weights initializations on the final distributions of weights and activations of different CNN architectures. The fine-grained, layer-wise analysis enables us to gain deep insights on how initial weights distributions will affect final accuracy and quantized behaviour. To our best knowledge, we are the first to perform such a low-level, in-depth quantitative analysis of weights initialization and its effect on quantized behaviour.

1 Introduction

Deep Convolutional Neural Networks (CNN) have enabled dramatic advances in the field of computer vision. An explosion of recent research has demonstrated incredible results in applications such as image classification [1–3], object detection [4–7], image segmentation [8, 9] and many more. As CNNs have become an increasing part of everyday life, their usage has expanded to mobile processing. With this expansion comes issues of minimizing latency, area, and power. Fixed point quantization of CNN weights and activations has become an essential tool for running efficient CNN inference. Various works have explored different quantization algorithms [10–12] to minimize the loss of information when mapping the weights and activations of a CNN into a discretized space. With these methods, the weights and activations can be represented as an n-bit (most often 8-bit) integer rather than 32-bit floating point numbers. Consequently, simple integer multiply-accumulate (MAC) operations can be performed rather than costly floating point arithmetic. This leads to significant savings in both computation (integer arithmetic) and storage (typically 8-bits per value or less).

Mobile hardware accelerators are usually limited in the types of operations that can be massively parallelized for fast execution. Thus, more complex quantization methods are often not supported by existing hardware. As such, other works have focused on quantization-algorithm-specific optimization methods (Eg. targeting 8-bit uniform quantization). These include quantization-aware fine-tuning [10] and differential optimization of quantization parameters [13, 14], eg. finding the optimal max/min thresholds of each weight/activation tensor for minimal quantized degradation. These methods train a model that is robust to quantized perturbations by simulating the error/noise of fixed point arithmetic.

Despite significant works demonstrating ways to recover near floating-point performance using reduced precision inference, there is still little understanding of how different design decisions can affect the quantized inference behaviour of CNNs. Weights initialization strategies are often designed with the goal of solving issues such as vanishing/exploding gradient [15–17]. However, an often-overlooked aspect of weights initialization is its impact on the final trained distributions of each layer. As they determine our starting point on the loss surface, initial distributions of each weight tensor will have a profound impact on the final trained model. Gradient descent is an incremental process with many small, noisy steps. Thus, an intelligent weight initialization strategy will have significant impact on the local minima reached on our path through the loss space. With regards to quantization, this means that weight initialization choices will have significant impact on the dynamic ranges of the weights and activations in a trained CNN. Thus, affecting the noise in our system and the expected quantized inference behaviour.

We propose a framework for in-depth, fine-grained quantitative analysis of the impact of various weights initialization strategies on final accuracy and quantized behaviour. By analyzing the trained distributions of each layer's weights and activations, we can gain deep insight on how different weights initialization strategies will affect the dynamic ranges of each layer. This in turn provides insight on the quantized behaviour of a CNN. Furthermore, we analyze the effect of these different weights initializations for a small set of different CNN architectures. Thus, we are able to isolate and observe the interplay between the CNN architecture choices (the parameterization) and the weights initialization strategy (the starting point on the parameterized loss surface). To our best knowledge, we are the first to perform such a systematic, low-level, quantitative analysis of weights initialization strategies and quantized behaviour. Furthermore, our framework for fine-grained analysis is applicable to analyzing any number of CNN design choices such as layer types, batch size, learning rate schedule etc.

2 Background

In early research, neural network parameters were often randomly initialized based on sampling from a normal or uniform distribution. The respective variance and range of these distributions would be hyperparameters for the practitioner to decide. While easily taken for granted, several works such as [15–17] have provided rigorous mathematical proofs showing how intelligent weights initialization strategies can solve issues of vanishing and exploding gradients. These works define *fan_in* and *fan_out* of a fully connected layer as the input/output units respectively. For convolution, it is defined as Eq. 1 where K is the kernel width (assume square kernel). They provide mathematical proofs on their proposed fan_in/fan_out-aware initialization strategies that scale the variance of gradients at each layer. Thus, avoiding failure modes created by vanishing and exploding gradients. While the introduction of Batch Normalization (BatchNorm) [18] layers has greatly mitigated training issues involving gradient scales, the choice of "where to begin" in the parameterized loss space is still extremely relevant. An often-overlooked effect of these initialization strategies is their impact on the trained dynamic ranges of each layer. As gradient descent is a noisy, iterative process with small, incremental steps, the final dynamic ranges of each layer are profoundly impacted by their starting point.

$$fan_{in/out} = K \times K \times channel_{in/out} \tag{1}$$

3 Fine-grained Layerwise Analysis

Besides a high-level study of how different weight initializations affect 32-bit floating point (fp32) and eight-bit quantized (quint8) accuracy, we also wish to gain detailed insight on the layer-wise distributions of final trained weights and activations. This information can give us an in-depth look at how the learning dynamics of various weight initializations play out. Furthermore, the dynamic ranges of each weight/activation tensor determine the resolution of the quantized step-size and, by extension, the quantization noise in a CNN. Thus, this analysis can help explain the observed quantized inference behaviour of different trained models. We propose systematically ablating through a variety of different weight initialization strategies while tracking the dynamic ranges of each layer's weights and activations during training. In this way, we can isolate the effect of these different design choices and analyze the changing distributions at each layer. We also track the "average channel precision". Average channel precision is defined as Eq. 2. Channel precision in this context is the ratio between an individual channel's range and the range of the entire layer. [19] uses this precision quantity to algorithmically maximize the channel precisions of each layer in a network prior to quantization. It can be seen as a measure of how well the overall layer-wise quantization encodings represent the information in each channel.

$$average\_precision = \frac{1}{K}\sum_{i=1}^{K}\frac{range_i}{range_{tensor}} \tag{2}$$

For dynamic ranges of activations, we randomly sample N training inputs from our training set and observe the corresponding activation responses. To reduce outlier noise, we perform symmetric percentile clipping (Eg. top and bottom 1%) and track the dynamic range and average precision of the clipped activations. As percentile clipping has become a ubiquitous default quantization setting we feel that this method establishes a realistic baseline of what can be expected during inference-time. Finally, there is one more set of dynamic ranges that must be observed. Batch Normalization has become the best-practice in a large range of CNN algorithms. However, their vanilla form is not well-suited for mobile hardware processing. Best practice for fast CNN inference usually involves folding the scale and variance parameters of a BatchNorm layer into the preceding layer's convolution parameters prior to quantization, as shown in Eq. 3. Therefore, we must also track the dynamic range and precision of our CNN's batchnorm-folded (BN-Fold) weights. In this manner, we can iterate through various weight initializations, gaining insights at each step on the trained models and their learning dynamics as well as the final weights and activations distributions. Our method can be extended to analyze a plethora of other design choices. These can include architecture choices such as layer-type, skip/residual connections as well as training hyperparameters such as learning rate schedules, batch size, optimizers etc. Despite their simplicity, such analyses can provide deep insight on the interplay of these various design choices and perhaps yield new understanding on their interaction.

$$w_{fold} = \frac{\gamma w}{\sqrt{EMA(\sigma_B^2) + \varepsilon}} \tag{3}$$

4 Experiment

For our experiment we use a simple, VGG-like macroarchitecture with four variations that differ in the micro-architecture of each layer (eg. type of convolution block used, use of BatchNorm and Relu etc. See Figure 1 for the general macro-architecture and details on the different variations of convolution layers). Our four CNNs are trained and tested on CIFAR-10 with a wide variety of different weight initialization strategies. These strategies can be separated into two categories of naive, straightforward strategies and more intelligent, layer-aware methods. Furthermore, the most common random weight initializations can also be categorized by the type of sampling distribution: random sampling from uniform distributions (hereafter referred to as RandUni) and random sampling from normal distributions (hereafter referred to as RandNorm). With considerations of dynamic range in mind, we seek to select distributions for the naive methods that would roughly correspond to small, medium, and large initial weights ranges. For the layer-aware initialization strategies, we use four commonly used methods introduced in [15, 16]. Named after the authors, we call them Glorot Uniform (GlorotUni) and Glorot Normal (GlorotNorm) from [15], He Uniform (HeUni) and He Normal (HeNorm) from [16]. In these works, the distribution range (for uniform sampling) and standard deviation (for normal sampling) for each layer are calculated based on *fan_in*, *fan_out*, or some combination of the two. We choose to focus on only the convolution layers and so the fully connected layers are always initialized using Glorot Uniform initialization. Furthermore, we also keep the weight initialization of the first convolution layer constant; only Glorot Uniform initialization is used. This was to keep the very first convolution layer as constant as possible.

Based on initial results showing Glorot Uniform having the most success in fp32 accuracy, we further experiment with Modified Glorot Uniform (ModGlorotUni) weights initialization strategies. The method of computing the max/min range of the uniform sampling distribution in Glorot Uniform initialization can be generalized as Eq. 4. In the original paper, $C = 6$. Following our established method of selecting distributions corresponding to small, medium, and large initial weights ranges, we select two values of C that would roughly correspond to medium and large ranges. The original Glorot Uniform leads to fairly small ranges. See Table 1 for a detailed breakdown of the sampling methods used in each of the 48 experiments.

$$max/min = \pm\sqrt{\frac{C}{fan\_in + fan\_out}} \tag{4}$$

Each network is trained for 200 epochs of SGD with Momentum = 0.9 and batch-size = 128. Initial learning rate is 0.01 and we scale it by 0.1 at the 75th, 120th, and 170th epochs. For the activation range tracking we perform top/bottom 1% clipping computed on a random sample of 1024 training samples. Basic data augmentation includes vertical/horizontal shift, zoom, vertical/horizontal flip and rotation. We use Tensorflow for training and quantizing the weights and activations to quint8 format.

Table 1: List of the various weight initialization strategies used. For methods that require some hyperparameter selection we include the values selected.

Initialization Method	Standard Deviation	Max/Min Value	C
RandNorm Large	1	N/A	N/A
RandNorm Med	0.5	N/A	N/A
RandNorm Small	0.1	N/A	N/A
RandUni Large	N/A	+/- 1	N/A
RandUni Med	N/A	+/- 0.5	N/A
RandUni Small	N/A	+/- 0.25	N/A
ModGlorotUni Large	N/A	N/A	1296
ModGlorotUni Med	N/A	N/A	36
GlorotUni	N/A	N/A	N/A
GlorotNorm	N/A	N/A	N/A
HeUni	N/A	N/A	N/A
HeNorm	N/A	N/A	N/A

For each network we evaluate testing performance with respect to 4 metrics: fp32 accuracy, quint8 accuracy, quantized mean-squared error (QMSE), and quantized crossentropy (QCE). Results are presented in Table 2. QMSE refers to the MSE between the fp32 network outputs and the quint8 network outputs after dequantization. Similarly, QCE measures the cross entropy between the fp32 network outputs and the dequantized quint8 network outputs. While QMSE directly measures how much the quint8 network outputs deviate from the fp32 network, QCE quantifies the difference in the distribution of the network outputs. For classification tasks, the quantized network can predict the same class as the fp32 network, despite deviations in logit values, if the overall shape of the output distribution is similar. Therefore QCE can sometimes be more reflective of differences in quantized behaviour. Additionally, we also observe the percent accuracy degradation (change in accuracy divided by fp32 accuracy) of each network after quantization. Though these quantities often track together, there can be scenarios where a network with more QMSE or QCE actually has less relative quantization degradation from a pure accuracy standpoint. This is likely explained by favourable rounding within the network.

5 Discussion

We can see in Table 2 that besides affecting the final FP32 accuracy of a given CNN architecture, the weights initialization strategy also has significant impact on the QUINT8 accuracy. Particularly worth noting is the markedly improved quantized behaviour in the DWS_Conv_With_BN networks trained using RandUni_Large initialization. Equally noteworthy is the stark drop in QUINT8 accuracy observed with the DWS_Conv_With_BN networks trained with the HeNorm and HeUni weight initializations and the Regular_Conv_With_BN network trained with ModGlorotUni_Med initialization. As expected, quantized accuracy usually worsened when BatchNorm layers were introduced. This is often attributed to the increased dynamic ranges/distributional shift introduced by BatchNorm Folding.

While each CNN architecture is trained on twelve different initialization methods, Regular_Conv_No_BN only has four results. This is because the other initialization methods had issues of exploding gradients. Their results were ommitted. Most of the DWS_Conv_No_BN experiments also did not learn but suffered from vanishing gradient issues instead. However, in our analyses we found that these vanishing gradients were not necessarily caused by a deep architecture leading to the gradient progressively vanishing during backpropagation. Instead, we observed a "vanishing activations" type phenomenon wherein the activations of the final Depthwise Separable Convolution block are exceedingly small. Thus, no gradients are able to propagate past the fully connected layers. Figure 2 shows a plot of the network activations in

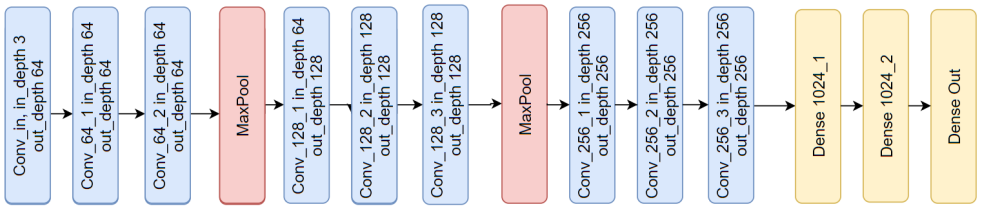

Fig. 1: **General Macroarchitecture of the CNN**. For our analysis we use a fixed macro-architecture so that we can isolate the interactions between various weight initialization strategies and a few different convolutional layer choices. We train four variations of this macro-architecture determined by the type of conv-block used at each layer: Regular_Conv_With_BN, Regular_Conv_No_BN, DWS_Conv_With_BN, and DWS_Conv_No_BN. These respectively correspond to using regular convolution followed by BatchNorm and Relu, regular convolution followed by only Relu and no BatchNorm, depthwise separable convolution blocks with BatchNorm and Relu after each convolution layer (same as the MobileNets block in [20]), and finally depthwise seperable convolution with only Relu and **no** BatchNorm after each convolution layer. The very first convolution layer stays fixed for all architectures, but follows the With/Without BatchNorm behaviour of the rest of the layers.

Table 2: Detailed results for each combination of weight initialization strategy and CNN architecture. The initialization strategies that suffered from vanishing/exploding gradients are ommitted. DWS_Conv_No_BN_GlorotUni is kept for illustrative purposes.

Network Architecture	FP32 Accuracy	QUINT8 Accuracy	QMSE	QCE	Percent Accuracy Decrease
DWS_Conv_No_BN_GlorotUni	10.00	10.00	0.000	2.303	0.00
DWS_Conv_No_BN_ModGlorotUni_Large	74.42	70.70	0.009	1.109	5.00
DWS_Conv_No_BN_RandNorm_Large	69.94	63.07	0.006	0.893	9.82
DWS_Conv_No_BN_RandNorm_Med	74.68	73.26	0.006	0.951	1.90
DWS_Conv_No_BN_RandUni_Large	75.22	72.77	0.004	0.872	3.26
DWS_Conv_With_BN_GlorotNorm	80.10	69.76	0.014	1.127	12.91
DWS_Conv_With_BN_GlorotUni	81.04	71.02	0.012	1.054	12.36
DWS_Conv_With_BN_ModGlorotUni_Large	76.33	68.64	0.012	1.789	10.07
DWS_Conv_With_BN_ModGlorotUni_Med	80.16	70.86	0.014	1.011	11.60
DWS_Conv_With_BN_HeNorm	79.48	55.56	0.033	2.400	30.10
DWS_Conv_With_BN_HeUni	80.51	62.49	0.024	1.786	22.38
DWS_Conv_With_BN_RandNorm_Large	74.93	66.32	0.010	1.358	11.49
DWS_Conv_With_BN_RandNorm_Med	77.99	66.32	0.016	1.694	14.96
DWS_Conv_With_BN_RandNorm_Small	80.61	70.12	0.013	1.464	13.01
DWS_Conv_With_BN_RandUni_Large	76.60	74.18	0.003	0.811	3.16
DWS_Conv_With_BN_RandUni_Med	78.40	67.82	0.016	1.993	13.49
DWS_Conv_With_BN_RandUni_Small	79.02	64.25	0.017	1.452	18.69
Regular_Conv_No_BN_GlorotNorm	87.03	84.46	0.005	0.585	2.95
Regular_Conv_No_BN_GlorotUni	86.89	85.51	0.003	0.403	1.59
Regular_Conv_No_BN_HeNorm	86.20	85.56	0.001	0.228	0.74
Regular_Conv_No_BN_HeUni	86.20	85.89	0.006	0.485	0.36
Regular_Conv_With_BN_GlorotNorm	89.34	86.33	0.005	0.340	3.37
Regular_Conv_With_BN_GlorotUni	88.53	88.33	0.002	0.207	0.23
Regular_Conv_With_BN_ModGlorotUni_Large	60.35	57.03	0.005	1.920	5.50
Regular_Conv_With_BN_ModGlorotUni_Med	84.60	60.08	0.029	3.217	28.98
Regular_Conv_With_BN_HeNorm	86.87	86.30	0.003	0.311	0.66
Regular_Conv_With_BN_HeUni	87.88	86.47	0.004	0.693	1.60
Regular_Conv_With_BN_RandNorm_Large	55.41	45.43	0.009	2.070	18.01
Regular_Conv_With_BN_RandNorm_Med	59.57	56.55	0.001	1.465	5.07
Regular_Conv_With_BN_RandNorm_Small	80.19	68.96	0.017	2.016	14.00
Regular_Conv_With_BN_RandUni_Large	58.69	58.15	0.002	1.577	0.92
Regular_Conv_With_BN_RandUni_Med	67.03	66.15	0.002	1.260	1.31
Regular_Conv_With_BN_RandUni_Small	76.28	75.80	0.002	0.888	0.63

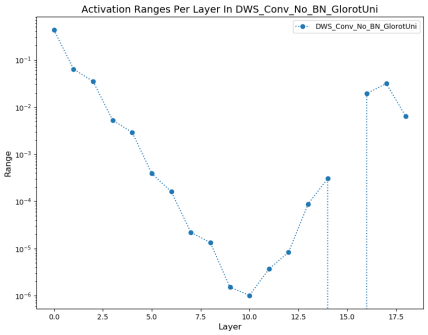

Fig. 2: **Vanishing Act!** In this figure we can see how the activation ranges of DWS_Conv_No_BN become increasingly small until they practically disappear. Consequently, gradients are not able to propagate past the fully-connected layers (final three points on the graph).

DWS_Conv_No_BN_GlorotUni. For illustrative purposes, we keep the DWS_Conv_No_BN_GlorotUni result and omit the rest. The normalization introduced by BatchNorm alleviates this issue as expected. One could consider an additional interpretation of BatchNorm as adding capacity to the network in the form of a learned **explicit scaling**. Scaling that would otherwise be too difficult for the convolution parameters to learn in addition to extracting features. We seek to follow-up on this hypothesis in future works. While we focus on the variations in quantized behaviour in this work, the varying FP32 accuracies are also worthy of close study. Our method sets out a framework through which we can systematically study these phenomena.

To better understand why we are observing the given quantized behaviour, we can use the proposed fine-grained analysis and inspect the distributions of each model layer-by-layer. With regards to the significantly improved quantized accuracy for DWS_Conv_With_BN_RandUni_Large, we observe in Figure 3 (top) that weights ranges don't necessarily tell the whole story. Despite having generally larger weights ranges, we start to see several other key areas in which the RandUni_Large layers stand out. For example, while the two He-initialized models tend to have a spike in the BN-Fold weights range at layer 2, RandUni_Large actually decreases in range. Furthermore, when we compare the BN-Fold weights precisions we also see a drop in precision for the other networks at layer 2 while the precision for RandUni_Large increases. With the activations, we see that all of the activation ranges increase at layer 2 while activation precisions decrease. However, RandUni_Large experiences a significantly smaller drop in activation precision. Thus, suggesting that RandUni_Large has a much higher retention of information in those crucial early stages of low-level feature extraction. Analyzing the change in the layerwise distributions during training might explain *why* we observe such a wide range of behaviour caused by varying weight initialization. It would also be worthwhile to observe the relative change in range/precision after BatchNorm folding. This would be a proxy for observing the distributional shift of the weights. While it is intractable to pinpoint any single reason, our layer-level analysis reveals a rich set of interactions that slowly build a detailed picture of each network's system dynamics as well as inter-network trends. We could further expand our analysis to use more rigorous, yet scalable statistical methods of analysis. For example, we know that a uniformly distributed tensor would best utilize the quantized steps of our given discretization method. Thus, computing the KL-divergence between a given weight/activation tensor and its corresponding uniform distribution (ie. a uniform distribution with the same bounds as the tensor) is a potential metric to explore. Overall, from these initial analyses, we see that taking a fine-grained, systematic approach to analyzing various design choices can yield detailed insights on the learning dynamics of a CNN.

6 Conclusion

We conduct the first in-depth, quantitative study of the impact of weight initialization strategies on final quantized inference behaviour of various basic CNN architectures. We show that in addition to affecting final floating point accuracy, a well-chosen weight initialization can also significantly affect a CNN's quantized accuracy. Future work includes further exploration of the interaction of BatchNorm with initial weight distributions, analysis of other intelligent initialization strategies, and analysis of weight initialization's impact on more complex architectures.

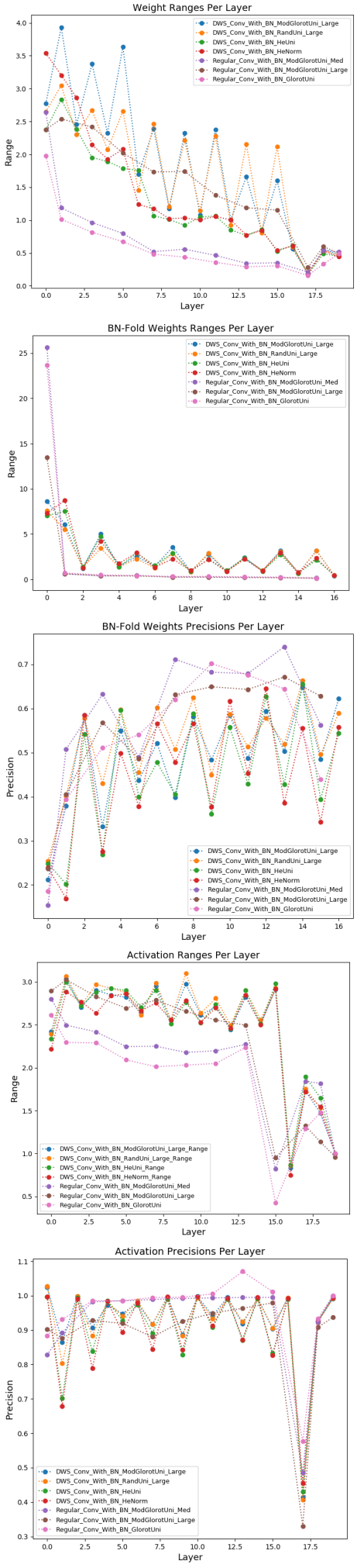

Fig. 3: A more low-level, focused look. Directly comparing a subset of the architectures. In order from top to bottom, the plots show: weights ranges per layer, BN-Fold weights ranges, BN-Fold weights precisions, activations ranges, and activation precisions. As we look to analyze any anomalies or unexpected behaviour, our fine-grained approach allows us to gain much more detailed insight as to what dynamics are at play when we introduce quantization noise.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[3] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[4] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[5] B. Wu, A. Wan, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 446–454.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[7] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017.

[10] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *CoRR*, vol. abs/1712.05877, 2017.

[11] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *CoRR*, vol. abs/1603.01025, 2016.

[12] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev, "Compressing deep convolutional networks using vector quantization," *CoRR*, vol. abs/1412.6115, 2014.

[13] S. R. Jain, A. Gural, M. Wu, and C. Dick, "Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware," *CoRR*, vol. abs/1903.08066, 2019.

[14] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: parameterized clipping activation for quantized neural networks," *CoRR*, vol. abs/1805.06085, 2018.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: JMLR Workshop and Conference Proceedings, 13–15 May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[17] B. Hanin and D. Rolnick, "How to start training: The effect of initialization and architecture," 2018.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[19] M. Nagel, M. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, nov 2019, pp. 1325–1334.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.