Transactions Brief

Fast and Low-Power Quantized Fixed Posit High-Accuracy DNN Implementation

Sumit Walia, Bachu Varun Tej, Arpita Kabra, Joydeep Devnath, and Joycee Mekie

***Abstract*— This brief compares quantized float-point representation in posit and fixed-posit formats for a wide variety of pre-trained deep neural networks (DNNs). We observe that fixed-posit representation is far more suitable for DNNs as it results in a faster and low-power computation circuit. We show that accuracy remains within the range of 0.3% and 0.57% of top-1 accuracy for posit and fixed-posit quantization. We further show that the posit-based multiplier requires higher power-delay-product (PDP) and area, whereas fixed-posit reduces PDP and area consumption by 71% and 36%, respectively, compared to (Devnath *et al.*, 2020) for the same bit-width.**

***Index Terms*— Convolutional neural net (CNN), deep neural network (DNN), fixed-posit representation, posit number system, quantization.**

I. Introduction

Deep neural networks (DNNs)-based Internet of Things (IoTs) have gained popularity recognizing their learning and processing capabilities [2], [3]. However, since DNNs are both data- and compute-intensive, they consume enormous power while inferencing, limiting their applicability for low-power IoTs. Quantization is a promising technique explored to reduce power as well as area requirement [4]. The rise in the influence of DNNs on various applications has led to the emergence of a reduced bit length weight-based hardware accelerator [5]. Most of the general-purpose processors employ IEEE 754 floating-point (FP) standard-based real number representation [6]. However, this format has a lot of shortcomings [7]. As a requirement of its replacement, Gustafson and Yonemoto [8] have proposed a novel FP number representation called posits. posits have higher accuracy and a larger dynamic range when compared to IEEE 754 FP standard. However, posits have larger power and area consumption. Recently, a new FP representation using fixed-posit [9] is proposed, keeping the dynamic range sufficient for a given application reducing the power and area requirements.

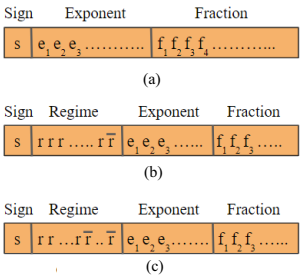

Fig. 1. Bit formats. (a) qNFP representation [1]. (b) Posit representation. (c) fixed-posit representation.

Previously, low bit-width posits were explored for DNNs by quantizing up to 8 b [10]. Devnath *et al.* [1] have rigorously analyzed IEEE 754 FP-based quantization, exploring eight DNN models; however, such exhaustive analysis has not been done using posits. Since the posit has a larger dynamic range [8] when compared to IEEE 754 FP representation, its ability to represent real numbers with fewer bits, preserving the information, makes it an appealing candidate to be used for quantization in DNNs. Similarly, fixed-posit [9] has never been explored extensively for DNN quantization. In this brief, we examine the possibility of bit quantization using posits and fixed-posits compared with the most recent work on DNN quantization [1]. We expect the comparisons with other schemes such as bfloat, FP8, and so on to follow the suite. The major contributions of this brief are as follows.

1) We have carried out rigorous analysis of DNNs for quantized weights. We have analyzed eight different DNNs using IEEE754, qNFP [1], posits [8], and fixed-posit [9].
2) We show that posit quantization achieves higher Top-1 accuracy when compared to [1] for same bit-width.
3) We further show that fixed-posit quantized multiplier reduces power-delay-product (PDP) up to 71% and area up to 36% compared to quantized non-standard floating point (qNFP) [1] with an average accuracy drop of 0.48% among the eight considered models.
4) We further show that larger DNNs can be quantized using fewer bits in fixed-posits compared to [1] than smaller DNNs with a negligible drop in Top-1 accuracy.
5) We have incorporated quantized fixed-posit multiplier along with quantized weights and quantized inputs to analyze the effect of quantization and observe negligible drop in Top-1 accuracy.

The rest of this brief is organized as follows: Section II discusses the background, Section III discusses the quantization of weights in posit, Section IV describes the evaluation and results, and finally, Section V concludes this brief.

Manuscript received May 31, 2021; revised October 15, 2021; accepted November 14, 2021. Date of publication December 10, 2021; date of current version January 17, 2022. This work was supported in part by the Ministry of Electronics and Information Technology (MEITY) through the SMDP-C2SD and the Young Faculty Research Fellowship (YFRF) Visvesvaraya Ph.D. Scheme; in part by the Science and Engineering Research Board (SERB) under Grant CRG/2018/005013, Grant MTR/2019/001605, and Grant SPR/2020/000450; and in part by the Semiconductor Research Corporation (SRC) under Contract 2020-IR-2980. *(Sumit Walia and Bachu Varun Tej contributed equally to this work.) (Corresponding author: Sumit Walia.)*

Sumit Walia was with IIT Gandhinagar, Ahmedabad, Gandhinagar 382355, India. He is now with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92092 USA (e-mail: sumit.walia@alumni.iitgn.ac.in).

Bachu Varun Tej was with IIT Gandhinagar, Ahmedabad, Gandhinagar 382355, India. He is now with NVIDIA, Bengaluru 560048, India (e-mail: tej_bachu@alumni.iitgn.ac.in).

Arpita Kabra and Joycee Mekie are with the Department of Electrical Engineering, IIT Gandhinagar, Ahmedabad, Gandhinagar 382355, India (e-mail: arpita.kabra@iitgn.ac.in; joycee@iitgn.ac.in).

Joydeep Devnath was with IIT Gandhinagar, Ahmedabad, Gandhinagar 382355, India. He is now with Ceremorphic Inc., Hyderabad 500081, India (e-mail: joydeep.devnath@mtech2017.iitgn.ac.in).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3131609.

Digital Object Identifier 10.1109/TVLSI.2021.3131609

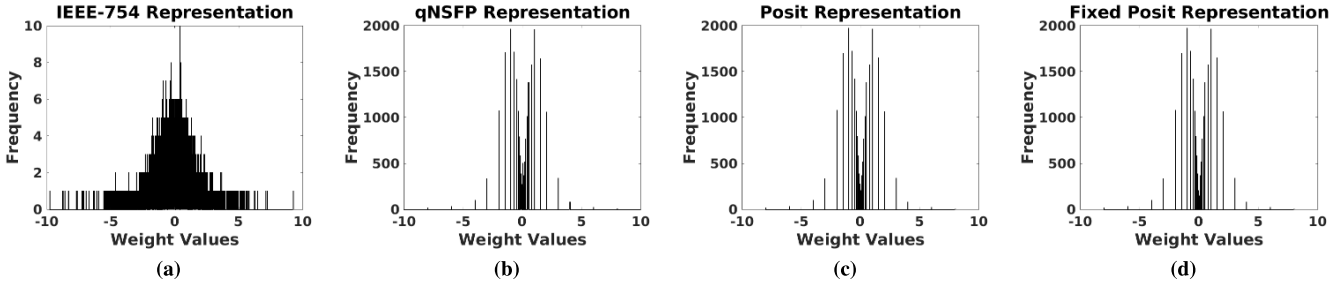

Fig. 2. Histogram of FCNN trained on MNIST weights. (a) Standard IEEE-754 representation. (b) qNFP representation ($N = 6$). (c) posit representation ($N = 6$, es $= 2$). (d) fixed-posit representation ($N = 6$, es $= 2$, regime $= 2$).

TABLE I
ACCURACY COMPARISON OF DNN MODELS ACROSS DIFFERENT WEIGHT QUANTIZATION TECHNIQUES

Network (DataSet)	layers	Base Top-1 Acc(%)	qNFP [1]		Posit			Fixed Posit			Memory Reduced w.r.t (%)	
			Acc(%)	(*N*)	Acc(%)	(*N*,*es*)	Acc Drop(%)	Acc(%)	(*N*,*es*,*regime*)	Acc Drop(%)	IEEE-754	qNFP
FCNN (MNIST)	3	95.5	95.2	6	**95.23**	(6,2)	0.27	95.19	(6,2,2)	0.31	81.25	0
					94.50	(5,2)	1.00				**84.375**	**16.67**
ISOLET (MNIST)	4	94.4	94.89	6	**94.16**	(6,2)	0.24	94.16	(6,3,2)	0.24	81.25	0
					93.97	(5,2)	0.42				**84.375**	**16.67**
CNN (cifar10)	19	83.00	82.77	8	**83.43**	(8,2)	-0.43	82.78	(8,4,2)	0.22	75	0
					82.79	(7,2)	0.20				78.125	12.5
VGG-16 (ImageNet)	16	71.21	70.95	8	**71.07**	(8,2)	0.14	70.94	(8,3,2)	0.27	75	0
					70.43	(7,2)	0.77				78.125	12.5
					66.29	(6,2)	4.92				**81.25**	**25**
VGG-19 (ImageNet)	19	71.19	70.85	8	70.92	(8,2)	0.29	70.84	(8,3,2)	0.35	75	0
					70.73	(7,2)	0.48				78.125	12.5
					66.39	(6,2)	4.82				**81.25**	**25**
ResNet-50 (ImageNet)	50	74.78	74.49	10	**74.71**	(10,2)	0.06	73.53	(10,4,2)	1.25	68.75	0
					74.45	(9,2)	0.33				71.875	10
					71.66	(8,2)	3.128				**75**	**20**
Inception-V3 (ImageNet)	48	77.06	76.9	10	**76.99**	(10,2)	0.06	76.09	(10,4,2)	0.97	68.75	0
					76.87	(9,2)	0.18				71.875	10
					75.83	(8,2)	1.23				**75**	**20**
Xception (ImageNet)	71	78.29	78.05	10	**78.28**	(10,2)	0.004	77.31	(10,4,2)	0.98	68.75	0
					78.08	(9,2)	0.20				71.875	10
					77.25	(8,2)	1.04				**75**	**20**

II. BACKGROUND

A DNN can be considered a flexible mapping from the input to the output feature state space. This mapping's parameters (weights) tunes to the desired form governed by the data fed during the training phase. The weights define the characteristic of the network. However, a DNN can comprise a million weights. Therefore, compression techniques are equipped to operate such massive networks. Since the DNNs are resilient to some amount of error, compression technique such as quantization is used to decrease the network size, preserving the accuracy from falling below a certain limit. These DNN weights are commonly represented and stored using IEEE-754 FP standard representation in a general-purpose processor. However, recently, many new formats are devised to represent an FP number. These formats are specifically designed to favor a particular application. In this brief, we have evaluated three such representations, mentioned as follows: 1) qNFP representation [1]; 2) posit format; and 3) fixed-posit format. Three terms define qNFP [1] format: Sign bit, exponent bits, and mantissa bits as shown in Fig. 1(a). qNFP follows the same principle as the IEEE-754 FP standard, except the fact that bit-width is variable. Any real number a, represented in qNFP, is given by $a = (-1)^S \times 2^{\text{exponent}-\text{bias}} \times (1.\text{mantissa})$.

A posit [8] representation is defined by a two-tuple *{N, es}* where N and es represent bit-width and the maximum number of exponent bits, respectively. Unlike IEEE 754 standard, posit representation is divided into four parts: sign, regime, exponent, and mantissa as shown in Fig. 1(b). Any real number a, represented in posit format, is given by $a = (-1)^S \times (2^{2^{\text{es}}})^k \times 2^{\text{exponent}} \times (1.\text{mantissa})$, where k value is decided by the regime bits.

A fixed-posit [9] representation is defined by a three-tuple *{N, es, regime}*, where N, *es*, and *regime* represent the total number of bits, the maximum number of exponent bits, and regime bits, respectively. Unlike posits, length of fixed-posit regime bits is fixed. The regime bits representation in fixed-posit is a sequence of 1's and 0's, similar to posits. But, if the representation needs fewer bits than the fixed length of regime bits, complement bits fill the remaining bits as shown in Fig. 1(c).

III. QUANTIZATION USING FIXED-POSITS

Reducing a set of infinite continuous values to a set of finite discrete values is referred to as quantization. Therefore, quantization gives the freedom to represent real numbers using reduced bit-width. Moreover, real numbers represented using fewer bits decreases area and power requirement while performing mathematical operations required by a DNN. This method further reduces the memory utilization to store millions of weights in each DNN layer. To understand the effect of quantization, we have plotted the weight distribution using four various number representation of a three-layer fully connected neural network (FCNN) model trained on the MNIST dataset. Fig. 2(a) shows the histogram plot of weights represented using single-precision (32 b) IEEE-754 FP standards. Most of the earlier works have focused on quantizing weights. Apart from this, we have a framework with fixed-posit multiplier within DNN

TABLE II
ACCURACY COMPARISONS USING QUANTIZED WEIGHTS, QUANTIZED INPUTS, AND QUANTIZED FIXED-POSIT MULTIPLIER FRAMEWORK

Network (dataset)	Base Top-1 Acc (%)	Fixed-posit	
		{N,es,regime}	Acc (%)
FCNN (MNIST)	95.5	6,2,2	94.12
LeNet5 (MNIST)	98.49	6,2,2	98.14
ISOLET (MNIST)	94.4	6,3,2	94.16
ResNet18 (CIFAR10)	82.21	10,4,2	80.73
VGG-16 (ImageNet)	72.28	8,3,2	68.08

inference engine. Distribution of quantized weights using qNFP, posit, and fixed-posit are shown in Fig. 2(b)–(d), respectively. The effect of quantization can be seen by the decrease in the number of levels on the x-axis and an increase in the frequency of each weight. For qNFP, we have chosen the bit-width equals 6, as shown in Fig. 2(b). {6,2} and {6,2,2} formats are employed for posit and fixed-posit representation, respectively. Table I represents the accuracy of various models with different number representations where only weights are quantized. We analyze performance by calculating accuracy drop after weight quantization of pre-trained models while inference comparing with non-quantized inference accuracy (Top-1 accuracy) as shown in Table I.

From Table I, the Top-1 accuracy of three-layer FCNN model trained on the MNIST dataset drops from 95.5% to 95.2%, 95.23%, and 95.19% for qNFP, posit, and fixed-posit-based representation, respectively. When compared to qNFP, posit-based quantization results in higher accuracy, however increases PDP by 2×. But, fixed-posit-based quantization consumes lower PDP with accuracy within the 0.01% range of Top-1 accuracy than qNFP.

In this work, we have only quantized DNN weights by deploying a wrapper around them and have not adulterated the DNN architecture. We can observe from Fig. 2 that the values are denser around zero on the whole number line. Hence, we require more quantized levels near zero, leading to nonlinear quantization. Since general-purpose processors deploy IEEE-754 FP standard-based representation of real numbers, we develop algorithms to convert 32 b (single-precision) representation to qNFP, posit, and fixed-posit format and vice versa. This methodology invariably introduces nonlinear quantization as required. We have used the following steps for weight quantization: 1) we first extract the weights for a pre-trained DNN model; 2) next, we develop an algorithm to convert real numbers from IEEE-754 FP representation to one among the three formats; 3) then, we pass the weights extracted from step-i through the developed algorithm to generate quantized weights; 4) we further develop an algorithm to convert the quantized weights into IEEE-754 FP format; and 5) finally, quantized weights replace the original weights in the pre-trained DNN model.

We have analyzed weight quantization on the following DNN models: FCNN trained on MNIST [11] and ISOLET [12] datasets, convolutional neural net (CNN) [13] trained on Cifar10 [14], VGG16 [15], VGG19 [16], Inception V3 [17], ResNet 50 [18], and Xception [19] trained on ImageNet [20] dataset.

A. DNN Inferencing Using Posit Multiplier

We have also implemented a framework with a fixed-posit multiplier. The comparison results of fixed-posit accuracy with Base Top-1 accuracy using this framework is reported in Table II. This framework converts both weights and inputs into quantized fixed-posits and also makes use of fixed-posit multiplier these quantized fixed-posits. We do expect the accuracy to slightly drop for a fixed quantization (N bits) for large networks. However, to the best of our knowledge, this is the first attempt to incorporate quantized fixed-posit multiplier hardware behavior in DNN inferencing, along with weights. Earlier work [1] only transforms weights into respective format, but the arithmetic calculations are done in IEEE-754 single-precision representation. As shown in Table II, for the same N (reported in Table I), the accuracy does not drop for FCNN, LeNeT5, and ISOLET when compared to the base accuracy. However, for large NNs, we need to add few more bits. For example, Resnet-18 needs $N = 10$, instead of $N = 8$ in [1]. Interestingly, for denser networks like VGG-16, N does not change.

IV. EVALUATION AND RESULTS

We have compared qNFP, posit, and fixed-posit multipliers based on PDP and area utilization.

A. PDP and Area Evaluation

For our experiments, we design the multipliers using Verilog. We synthesize them on 28-nm FDSOI technology node at 1.0-V supply voltage and at 25 °C using the Synopsys design compiler (DC) for maximum frequency of operation. We compute the total power by testing the multiplier on 100k numbers generated uniformly at random within the range 2^{-126}–2^{127}.

We have divided our experiments into two phases. In the former part, we procure the 100k trace of uniformly distributed input operands for multiplication and consequently obtain netlist for all the designs simulated at maximum possible frequency for respective designs. In the second phase, we convert the trace into switching activity interchange format (SAIF) files. We obtain power values by simulating all the netlist generated in the former phase at the lowest frequency among all designs. Along with power, we obtain area and delay results for all the designs using DC, as shown in Table III.

B. Comparison of Posit With qNFP

It can be inferred from Table II that posit quantization with {N,2} configuration obtains higher accuracy for seven out of eight models when compared to qNFP-based quantization. We have evaluated posit quantization performance by generating accuracy for bit-width from N as mentioned in [1] up to $N-2$ as shown in Table II. On an average, posit {N,2} configuration has an accuracy within the range of 0.08% Top-1 accuracy. Furthermore, as es increases from 2 to 3, it is observed that the accuracy for posit {N,3} configuration drops, although remaining within the range of 0.34% Top-1 accuracy.

Table III shows PDP and area utilization by multipliers based on posit formats. For the IEEE 754 FP standard-based multiplier, the PDP and area are 1578.42 μW–μs and 2012 μm^2, respectively. Overall, posit {N,2} configuration-based multiplier consumes 95% and 88% less PDP and area, respectively, compared to the IEEE 754 FP standard-based multiplier, but more PDP and area compared to the qNFP-based multiplier. But, as es bits increased from {es = 2} to {es = 3}, we observe performance benefits for posit {$N-1$,es = 3} configuration as shown in Table III. We observe that larger DNNs like Xception can be compressed more than smaller DNNs like three-layered FCNN or VGG-16 with minimal accuracy drop along with PDP lowering.

C. Comparison of fixed-Posit and Posits

The fixed-posit format has a dynamic range lower than the posits; however, it reduces the PDP and area utilization for a higher bit-width-based multiplier [9]. We evaluated fixed-posit format performance for lower bit-width. Table III includes qNFP, posit, and fixed-posit configurations only for maximum accuracy for each DNN model. Although we expect accuracy to drop drastically for fixed-posit-based quantization compared to posit quantization due to the lower dynamic range, the accuracy remains with the range

TABLE III
PDP AND AREA COMPARISON OF DNN MODELS ACROSS DIFFERENT QUANTIZATION TECHNIQUES

Network	qNFP			Posit			Fixed-posit				
	N	PDP (μW-μs)	Area (μm^2)	{N,es}	PDP (μW-μs)	Area (μm^2)	{N,es,regime}	PDP (μW-μs)	Area (μm^2)	PDP reduced (%)	Area reduced (%)
FCNN-MNIST	6	20.69	75.07	{6,2}	42.92	165.48	{6,2,2}	17.54	122.23	**59.13**	**26.13**
FCNN-ISOLET	6	20.69	75.07	{6,2}	42.92	165.48	{6,3,2}	18.8	118.8	**56.19**	**28.2**
CNN-CIFAR10	8	30.87	112.77	{8,2}	86.55	246.04	{8,4,2}	25.84	160.91	**70.14**	**34.6**
VGG-16	8	30.87	112.77	{8,2}	86.55	246.04	{8,3,2}	24.77	168.09	**71.38**	**31.68**
VGG-19	8	30.87	112.77	{8,2}	86.55	246.04	{8,3,2}	24.77	168.09	**71.38**	**31.68**
ResNET-50	10	88.89	230.27	{10,2}	104.51	324.76	{10,4,2}	31.56	206.9	**69.8**	**36.29**
INCEPTION-V3	10	88.89	230.27	{10,2}	104.51	324.76	{10,4,2}	31.56	206.9	**69.8**	**36.29**
XCEPTION	10	88.89	230.27	{10,2}	104.51	324.76	{10,4,2}	31.56	206.9	**69.8**	**36.29**

of 0.48% for all eight models. Furthermore, fixed-posit quantization accuracy remains within the range of 0.57% of Base Top-1 accuracy. We also observed that multiplier based on fixed-posit format benefits by 71.38% and 36.29% in PDP and area utilization, respectively, compared to posit format-based multipliers as shown in Table III.

V. CONCLUSION

In this brief, we have proposed posit and fixed-posit-based quantization of pre-trained DNN models. posits provides higher accuracy and memory reduction than IEEE 754 standard and qNFP [1]. Our evaluation shows that posit quantization achieves accuracy within the range of 1% when compared to IEEE 754 standard. We show that while the posit multiplier has high PDP, the fixed-posit multiplier have 71% and 36% lower PDP and area, respectively, compared to the posit-based multiplier. The accuracy obtained with fixed-posit is within 0.57% range of Base Top-1 accuracy and 0.48% accuracy based on posit quantization. Furthermore, we show that even with quantized fixed-posit inputs, weights, and multipliers, the DNN accuracy does not drop significantly when compared to the Top-1 base accuracy.

ACKNOWLEDGMENT

The authors would like to thank Dr. Chandan Jha, IIT Bombay, Mumbai, India, and Rajesh Kumar, Marvell Technology, Bengaluru, India, for help with machine learning experiments.

REFERENCES

[1] J. K. Devnath, N. Surana, and J. Mekie, "A mathematical approach towards quantization of floating point weights in low power neural networks," in *Proc. 33rd Int. Conf. VLSI Design*, 2020, pp. 177–182.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[3] C. Razafimandimby, V. Loscri, and A. M. Vegni, "A neural network and IoT based scheme for performance assessment in Internet of Robotic Things," in *Proc. IEEE 1st Int. Conf. Internet Things Design Implement. (IoTDI)*, Apr. 2016, pp. 241–246.
[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.
[5] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
[6] *Ieee Standard for Floating-Point Arithmetic*, Standard 754-2008, 2008, pp. 1–70.
[7] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991.
[8] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, 2017.
[9] V. Gohil, S. Walia, J. Mekie, and M. Awasthi, "fixed-posit: A floating-point representation for error-resilient applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 10, pp. 3341–3345, Oct. 2021.
[10] J. Lu, C. Fang, M. Xu, J. Lin, and Z. Wang, "Evaluations on deep neural networks training using posit number system," *IEEE Trans. Comput.*, vol. 70, no. 2, pp. 174–187, Feb. 2021.
[11] CorochannNote. *MNIST Dataset Introduction*. Accessed: Dec. 20, 2020. [Online]. Available: https://corochann.com/mnist-dataset-introduction-1138.html
[12] D. Dua and C. Graff. (2019). *UCI Machine Learning Repository*. Accessed: 15, 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/isolet
[13] B. Graham, "Fractional max-pooling," 2014, *arXiv:1412.6071*.
[14] Corochann. *CIFAR-10, CIFAR-100 Dataset Introduction*. Accessed: Dec. 20, 2020. [Online]. Available: https://corochann.com/cifar-10-cifar-100-dataset-introduction-1258.html
[15] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Accessed: Dec. 10, 2020. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/
[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2016, pp. 2818–2826.
[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2016, pp. 770–778.
[19] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[20] J. Brownlee. *Introduction to Machine Learning With Scikit-Learn*. Accessed: Dec. 1, 2020. [Online]. Available: https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/