# Shedding Light on Variational Autoencoders

J.C. Ruiz Vargas, S.F. Novaes, R. Cobe, R. Iope, S. Stanzani, T.R. Tomei
São Paulo State University (Unesp)
Center for Scientific Computing (NCC), São Paulo
Email: [jruizvar, novaes, rmcobe, rogerio, silvio, trtomei]@ncc.unesp.br

***Abstract*—Deep neural networks provide the canvas to create models of millions of parameters to fit distributions involving an equally large number of random variables. The contribution of this study is twofold. First, we introduce a diffraction dataset containing computer-based simulations of a Young's interference experiment. Then, we demonstrate the adeptness of variational autoencoders to learn diffraction patterns and extract a latent feature that correlates with the physical wavelength.**

## I. Introduction

The pupil of the human eye is a circular aperture through which light diffracts, giving us the ability to resolve objects at large distances and to distinguish text characters. Due to the diffraction of light you can enjoy reading this article as well as the subtitles on your favorite movies.

Young's double-slit experiment [1] is a classical demonstration of the physical wave nature of light revealed by diffraction patterns. A diffraction pattern consists of dark regions of destructive interference and bright regions of constructive interference. Diffraction phenomena occur when the light passes through a slit comparable in size with the wavelength $\lambda$, resulting in a diffraction pattern that depends on the geometry of the experimental setup.

In the realm of computer vision an image $X(n_h, n_w, n_c)$ is described by a tensor of height $n_h$, width $n_w$ and $n_c$ channels. The normalized intensity of each pixel takes values in the interval $[0, 1]$ allowing a probabilistic interpretation. Since the intensity of a pixel is a random variable, the image itself is the manifestation of an stochastic event. Think of the set of all images $\mathbb{A}$ and the subset of diffraction images $\mathbb{B} \subset \mathbb{A}$. The chances of drawing an element $X \in \mathbb{B}$ are very low if we sample from a uniform distribution. However, if we have an idea of how diffraction images are distributed, the chances of drawing $X \in \mathbb{B}$ are enhanced.

Using variational autoencoders [2], it is possible to shrink the representation of an image from a tensor of $n_h \times n_w \times n_c$ axes into a vector of lower dimension. This transformation is accomplished by the encoding layer, yielding a representation known as the latent space. Subsequently, a point in the latent space is upscaled by the decoder into the original high dimensional space, generating synthetic images with hopefully the same patterns as the training set.

Generative adversarial networks [3] and variational autoencoders are two different strategies for learning latent spaces of image representations, each with its own characteristics. Autoencoders are appropriate for learning latent spaces that are well structured, where specific directions encode a meaningful axis of variation in the data. Adversarial networks generate images that can be highly realistic [4], but the latent space may not have as much structure and continuity.

Generative models have been applied successfully in computer vision for the generation of handwritten digits [5] and pose person images [6]. Applications in the domain of physical sciences envision generative models to reduce the processing time of particle physics simulations [7].

This study investigates the capability of a variational autoencoder to capture the distribution of the wavelength $\lambda$, given a set of diffraction patterns generated by the simulation of a Young's interference experiment. In addition, we want to encourage the usage of generative models in scientific applications that rely on expensive computational simulations, in order to speed up their analysis workflow.

The remainder of this article is organized as follows. Section II explains briefly the autoencoders's architecture and how they work. In Section III, we give details of the experiment's geometry and the simulation of the diffraction patterns. In Section IV-A, we describe the specific architecture of the deep learning model and its training curve. Section IV-B displays some examples of synthetic patterns produced by the trained model. In Section V, we present performance benchmarks using some of the latest Intel processors. Finally, Section VI summarizes the conclusions and outlooks.

## II. Autoencoders

Deep neural networks provide the canvas to create models of millions of parameters, which in principle could fit any distribution defined over a random variable $X$.

Autoencoders [8] are unsupervised neural networks that aim to generate an output as identical as possible to the input. Autoencoders are not intended to copy the full extent of the input but they are designed to perform an incomplete copy. Such a requirement is accomplished by introducing a bottleneck between the input and the output. In fact, autoencoders can be used for dimensionality reduction and data compressing.

The architecture of an autoencoder includes two modules, the encoder and the decoder. The last layer of the encoder is the bottleneck, also known as the latent space. The bottleneck captures the most representative features in the data, which is quite convenient for feature extraction [9]. Moreover, autoencoders are differentiable feedforward networks that can be trained via backpropagation [10].

A linear activation autoencoder can learn to copy the input to the output provided a latent space with the dimension of

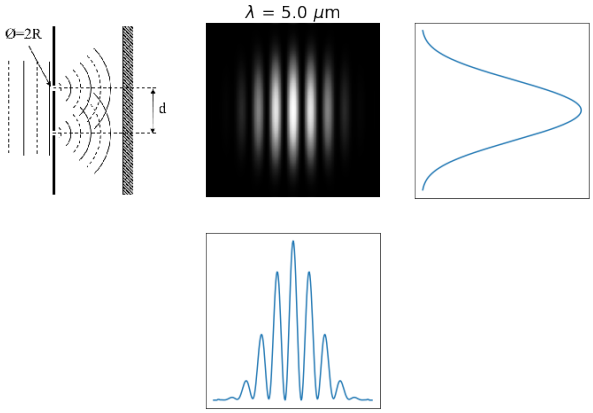


Fig. 1. Diffraction pattern produced by a monochromatic light passing through two circular apertures. The intensity profile displays peaks of constructive interference and valleys of destructive interference.

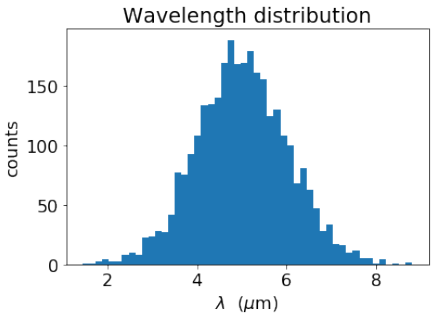


Fig. 2. Distribution of the wavelength in the diffraction dataset.

the input. To prevent this behaviour, the dimension of the bottleneck should be lower than the input in order to select effectively the most important features. The encoding is said to be undercomplete if the bottleneck has lower dimension than the input; conversely, the encoding is overcomplete if the bottleneck has dimension equal or larger than the input. In this study we use undercomplete encoding to learn the most important feature contained in the diffraction dataset to be described in Section III.

Variational autoencoders (VAE) are a kind of generative model proposed by Kingma and Welling in 2013 [2], and by Rezende, Mohamed, and Wierstra in 2014 [11]. Variational autoencoders increase the chances of drawing images according to the distribution learned from a training set, and they can handle unsupervised problems since the objective function does not require any additional label besides the input $X$.

In contrast to common autoencoders, which generate a fixed compact representation of the input data, VAEs learn the parameters for the probability distribution of the input data. During the training phase, a point in the latent is drawn from a distribution requiring two hyper-parameters: the mean and the standard deviation. Therefore, the decoder component samples from this distribution to generate synthetic images.

## III. Diffraction Dataset

Consider the setup of a double slit diffraction experiment as shown in Figure 1. A plane light wave –such as the one generated by a laser– with wavelength $\lambda$ shines upon an opaque screen $S_1$. Two circular apertures of radius $R$ = 0.15 mm have been opened, vertically separated by a distance $d$ = 1.2 mm. On the other side of $S_1$, centered upon the midpoint of the two apertures, a photosensitive screen $S_2$ of area 20×20 mm$^2$ is excited by the light that goes through the apertures; the distance between $S_1$ and $S_2$ is $r_0$ = 50 cm. $S_2$ is divided into discrete elements (pixels) arranged in a square grid of 256×256, producing grayscale images with the pixel value proportional to the intensity of the incident light. Notice that for the visible and near infrared sectors of the electromagnetic spectrum, $d^2/R\lambda \sim 1$, therefore the experiment is in the so-called Fresnel diffraction regime.

The diffraction dataset was produced by running 3000 simulations [12] of the double slit experiment keeping all the geometrical parameters fixed but letting the wavelength $\lambda$ to vary according to the distribution in Figure 2. Since the diffraction pattern depends on $\lambda$, the images in the diffraction dataset are not identical themselves. The resolution of the grayscale images is $256 \times 256$ pixels, which accounts for $3000 \times 256 \times 256 \approx 200$ million random variables in the whole dataset. We use the diffraction dataset to train the model described in Section IV-A.

## IV. Analysis

In this section we present the architecture of the model (Section IV-A), the synthetic patterns generated by the decoder as well as the distribution of the latent coordinate represented by the encoder (Section IV-B). Finally, we discuss the results and further ideas for improvements (Section IV-C).

### A. Model Description

The architecture of a variational autoencoder contains two pieces. The first component is the encoder that reduces the representation of the input images, followed by the decoder that restores the output of the encoder into the shape of the input images. The layers in the encoder with trainable parameters are 4 convolutions and 3 fully connected. In the decoder the trainable layers are composed by one fully connected, one transposed convolution and one regular convolution.

The detailed architecture of the model is shown in Tables I and II, which in total contains about 36 million trainable parameters. Let us recall that this deep learning model aims to find the probability distribution that best fits the 200 million random variables in the diffraction dataset. In order to accomplish the learning process, we trained the model on 2048 samples and validated it on 952 samples during 10 epochs. The evolution of the learning curve is shown in Figure 3, and we can observe that the model succeeded on optimizing the objective loss function.

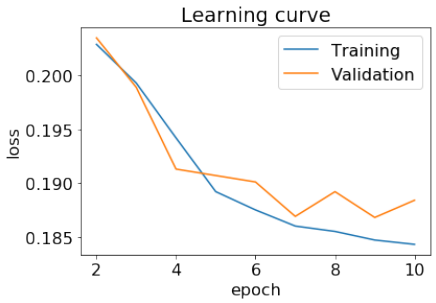


Fig. 3. Minimization of the loss during the training phase of the model.

TABLE I
ARCHITECTURE OF THE ENCODER

| Layer | Output Shape | Param # |
|---|---|---|
| InputLayer | (None, 256, 256, 1) | 0 |
| Conv2D | (None, 256, 256, 32) | 320 |
| Conv2D | (None, 128, 128, 64) | 18496 |
| Conv2D | (None, 128, 128, 64) | 36928 |
| Conv2D | (None, 128, 128, 64) | 36928 |
| Flatten | (None, 1048576) | 0 |
| Dense | (None, 32) | 33554464 |
| Dense | (None, 1) | 33 |
| Dense | (None, 1) | 33 |
| Lambda | (None, 1) | 0 |
| Total parameters: 33,647,202 | | |

TABLE II
ARCHITECTURE OF THE DECODER

| Layer | Output Shape | Param # |
|---|---|---|
| InputLayer | (None, 1) | 0 |
| Dense | (None, 1048576) | 2097152 |
| Reshape | (None, 128, 128, 64) | 0 |
| Conv2DTr | (None, 256, 256, 32) | 18464 |
| Conv2D | (None, 256, 256, 1) | 289 |
| Total parameters: 2,115,905 | | |

### *B. Results*

We generated synthetic patterns using the trained model to evaluate its quality. This procedure requires only the decoder part of the model, which is basically a function that receives a single number and produces a matrix of shape $256 \times 256$. We denoted the input of the decoder as $z$ and explored values between 0 and 4 in steps of 0.5. The generated patterns obtained for the different values of $z$ are shown in Figure 4. As we increase the value of $z$ the bright regions in the generated pattern become more prominent. This behaviour is the same observed in the diffraction dataset when examining increasing values of $\lambda$.

We also found the probability distribution of the latent variable encoded by the model. We did so by passing the whole diffraction dataset into the trained encoder. As shown in Figure 5, the latent variable follows an asymmetric distribution with mean value around $z = 1$ in a range between $-1.4$ and $4.3$. We observed a clear correlation between the latent coordinate $z$ and the physical wavelength $\lambda$, as verified in Figure 6.

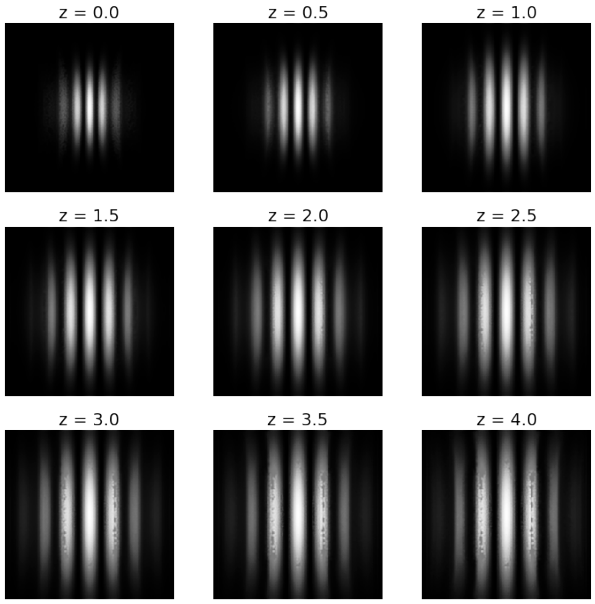


Fig. 4. Synthetic patterns generated by the decoder for different values of the latent coordinate.

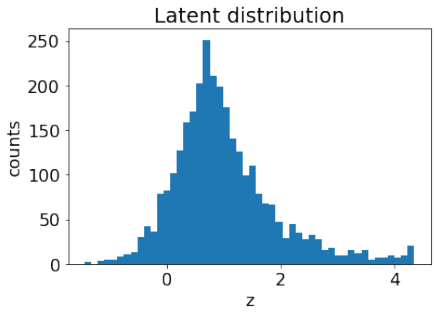


Fig. 5. Distribution of the latent variable encoded by the model.

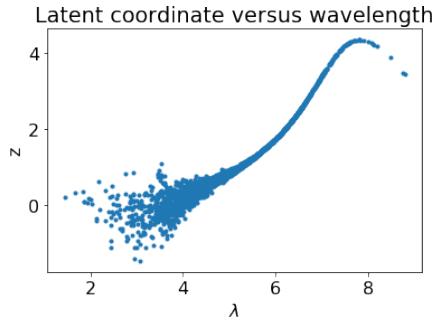


Fig. 6. Correlations between the latent coordinate and the wavelength.

### C. Discussion

We constrained the latent space to one dimension but the generalization to spaces of higher dimensions is possible as long as we keep it smaller than the input data, otherwise, the autoencondar might learn to copy the input to the output as discussed in Section II.

As explained in Section III, the diffraction dataset includes images with a wavelength $\lambda$ drawn from a gaussian distribution. Therefore, it is not surprising to observe a distribution of $z$ with a similar shape. The most remarkable result is the correlation between $\lambda$ and $z$, which displays a region of lower correlation ($z < 1$) and a region of higher correlation ($z > 1$).

The correlation between the latent space and a physical property opens the possibility to analyze more complex scenarios. For instance, the experimental resolution is an important uncertainty that needs to be considered in a real life experiment. Systematic uncertainties, which are always present due to detector calibrations and light source imperfections, may spoil the nice correlation we obtained with our simplified setup. Therefore, further studies are needed to test the robustness of a variational autoencoder against uncertainties.

## V. Performance Evaluation

Deep learning applications are computationally intensive tasks that rely on high performance computing. In this session, we describe the software and hardware infrastructure used for training the model, the aspects that influence performance and the opportunities for improvement.

The code was written in Python enhanced with the deep learning frameworks Keras [13] and TensorFlow [14]. In addition, the Intel Math Kernel Library (MKL) [15] provides support to both Keras and TensorFlow, allowing optimized executions of mathematical operations on Intel hardware.

The Center for Scientific Computing[1] at the São Paulo State University (Unesp) hosts an heterogeneous cluster composed by three different generations of Intel Xeon processors: Sandy Bridge EP [16], Haswell [17] and Skylake [18]. The configuration of each node is shown in Table III.

The parallelization of a task is performed by TensorFlow in a multithreading level, splitting the execution of each epoch in several threads. In this context, our current implementation explores parallel execution in a single node only. In Table IV, we show the average execution time per epoch and the average execution time per task (10 epochs).

We profiled the application using Intel VTune[2] in order to investigate the vector instruction set, which is the metric for arithmetic floating point computations and memory access operations. In this regard, we highlight the following aspects related to the performance results:

- The vector instruction set available on each node was effectively used by TensorFlow and MKL.
- The Haswell architecture improved the performance of training by at least 15% compared to Sandy Bridge EP. This improvement occurred because the Haswell architecture provides more processing units than Sandy Bridge EP.
- The performance on Skylake was compatible with the Haswell architecture.
- The Skylake has more cores and a better vector instruction set compared to Haswell, but the execution on these nodes presents high thread imbalance.
- Thread imbalance harms performance improvements that could be achieved by using more cores.

[1]CoE in Machine Learning: https://coe-ml.ncc.unesp.br

[2]VTune: https://software.intel.com/en-us/intel-vtune-amplifier-xe

TABLE III
Heterogeneous Cluster Configuration

| Architecture | Cores | RAM |
|---|---|---|
| Sandy Bridge EP | 16 | 62 GB |
| Haswell | 36 | 126 GB |
| Skylake | 48 | 192 GB |

TABLE IV
Time benchmarks for Intel Xeon processors

| # of epochs | Sandy Bridge EP | Haswell | Skylake |
|---|---|---|---|
| 1 | 300 s | 256 s | 257 s |
| 10 | 3027 s | 2566 s | 2576 s |

The following aspects could be improved to explore performance optimization:

- Investigate thread affinity, which is the mapping of thread to each core and the amount of threads to be used in order to minimize thread imbalance.
- Implement multiprocess paralelization in order to use several nodes to execute the inner work of each epoch concurrently, and consequently take advantage of the high performance network deployed in this cluster. These process could be speeded-up by using a low-latency, high-bandwidth interconnection, such as Intel Omni-Path architecture (OPA) [19].

## VI. Conclusions

Variational autoencoders are versatile models able to tackle unsupervised learning problems. This study introduces the diffraction dataset containing simulations of a Young's interference experiments. We demonstrate the adeptness of variational autoencoders to learn diffraction patterns and extract a latent feature that correlates with the physical wavelength.

After training (validating) the model over 2048 (952) images during 10 epochs, the autoencoder started generating realistic diffraction patterns. On top of that, the distribution of the latent feature extracted by the encoder resembles the physical wavelength distribution.

The expensive computational load demanded to train the model was supported by a hardware infrastructure based on the latest Intel Xeon processors. Performance benchmarks were established for three generations of Intel processors and guidelines to improve the configuration of an heterogeneous cluster were discussed.

*Outlooks*

Concerning the future endeavors, we will investigate the impact of systematic uncertainties on the quality of the images generated. We will also perform a study on the size of the latent space and try to assess the trade-off between the quality of the generated samples and the dimension of the latent space. We also intend to perform studies on the quality of the generated data, by performing a more complete statistical comparison with the real data. Moreover, we will investigate alternatives to improve the training performance. We are specially interested in leveraging from the high-bandwidth low-latency Intel omni-path network interconnection between the nodes of our heterogeneous cluster, in order to perform distributed training.

In addition to the contributions mentioned above, this study intends to motivate researchers to explore different deep neural architectures on top of the diffraction dataset[3], in order to extend the scope of machine learning applications in the domain of physical sciences.

Generative models are a good fit to deal with complex simulation. One domain that might take advantage of these models is High Energy Physics [20]. The experiments at CERN's Large Hadron Collider [21] are particle factories producing enormous amounts of data that rely on complex and computationally costly simulations. Such experiments may find variational autoencoders and generative models very useful alternatives to perform particle physics simulations in a more affordable way.

Another usage of this kind of generative models is in embedded hardware solutions, given the fact that Field Programmable Gate Arrays (FPGAs) have a processor much more simpler than server CPUs. Using solutions that could generate data by just applying matrix multiplications instead of implementing operations that rely on advanced mathematical functions, could save not just execution and development time but also power consumption.

## ACKNOWLEDGMENT

This material is based upon work supported by the São Paulo Research Foundation (FAPESP) under Grant N$^o$ 2013/01907-0 and Grant N$^o$ 2016/15897-4. This work is also supported by Fundação para o Desenvolvimento da Unesp (Fundunesp) under Grant N$^o$ 2597/2017-CCP in cooperation agreement with Intel®.

## REFERENCES

[1] M. Born and E. Wolf, *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Elsevier Science, 2013. [Online]. Available: https://books.google.com.br/books?id=HYGDAAAQBAJ

[2] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *ArXiv e-prints*, Dec. 2013.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *ArXiv e-prints*, Jun. 2014.

[3] Available upon email request: coe-ml@ncc.unesp.br

[4] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016. [Online]. Available: http://arxiv.org/abs/1609.04802

[5] C. Doersch, "Tutorial on Variational Autoencoders," *ArXiv e-prints*, Jun. 2016.

[6] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool, "Pose Guided Person Image Generation," *ArXiv e-prints*, May 2017.

[7] M. Paganini, L. de Oliveira, and B. Nachman, "CaloGAN : Simulating 3D high energy particle showers in multilayer electromagnetic calorimeters with generative adversarial networks," *Phys. Rev.*, vol. D97, no. 1, p. 014021, 2018.

[8] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[9] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[10] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[11] D. Jimenez Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," *ArXiv e-prints*, Jan. 2014.

[12] F. Van Goor and G. Vdovin, "Lightpipes for python: Beam propagation toolbox," https://github.com/opticspy/lightpipes, 2018.

[13] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[15] I. Software, "Intel math kernel library," https://software.intel.com/mkl, 2018.

[16] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, March 2012.

[17] P. Hammarlund, A. J. Martinez, A. A. Bajwa, D. L. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, R. B. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, S. Gunther, T. Piazza, and T. Burton, "Haswell: The fourth-generation intel core processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, Mar 2014.

[18] S. M. Tam, H. Muljono, M. Huang, S. Iyer, K. Royneogi, N. Satti, R. Qureshi, W. Chen, T. Wang, H. Hsieh, S. Vora, and E. Wang, "Skylake-sp: A 14nm 28-core zeon processor," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 34–36.

[19] M. S. Birrittella, M. Debbage, R. Huggahalli, J. Kunz, T. Lovett, T. Rimmer, K. D. Underwood, and R. C. Zak, "Enabling scalable high-performance systems with the intel omni-path architecture," *IEEE Micro*, vol. 36, no. 4, pp. 38–47, July 2016.

[20] D. H. Perkins, *Introduction to high energy physics; 4th ed.* Cambridge: Cambridge Univ. Press, 2000. [Online]. Available: https://cds.cern.ch/record/396126

[21] A. Breskin and R. Voss, *The CERN Large Hadron Collider: Accelerator and Experiments*. Geneva: CERN, 2009. [Online]. Available: https://cds.cern.ch/record/1244506