A Unifying Theory of Learning: DL Meets Kernel Methods

I. de Zarzà

I. de Zarzà.

A Unifying Theory of Learning: DL Meets Kernel Methods.

ETH Zürich.

I. DE ZARZÀ

A UNIFYING THEORY OF LEARNING: DL MEETS KERNEL METHODS

Doctor of Science at

ETH Zürich.
Department of Information Technology and Electrical Engineering.
zarza@vision.ee.ethz.ch

May 2021. Zürich.

DISSERTATION ETHZ

A UNIFYING THEORY OF LEARNING: DL MEETS KERNEL METHODS

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCE of ETH ZÜRICH
(Dr. sc. ETH Zürich)

presented by

I. DE ZARZÀ
Master of Science (CS) at Carnegie Mellon.
Master of Science (EE) at City University of Hong Kong.
Master of Science (Mathematics) at Universitat Autònoma de Barcelona.
Master of Science (Mathematics) at Universitat de Barcelona.

Born on 13 November 1989.
Citizen of Catalunya (Regne d'Espanya).

accepted on the recommendation of

Prof. Dr. Luc van Gool, examiner.
Dr. Radu Timofte, co-examiner.

14th May 2021

I. de Zarzà: *A Unifying Theory of Learning: DL Meets Kernel Methods*, © 2021

DOI: 10.3929/ethz-a-

To De Curtò.

ABSTRACT

We introduce a framework to use kernel approximates in the mini-batch setting with Stochastic Gradient Descent (SGD) as an alternative to Deep Learning. Based on Random Kitchen Sinks [1], we provide a C++ library for Large-scale ML. It contains a CPU optimized implementation of the algorithm in [2], that allows the computation of approximated kernel expansions in log-linear time. The algorithm requires to compute the product of matrices Walsh Hadamard. A cache friendly Fast Walsh Hadamard that achieves compelling speed and outperforms current state-of-the-art methods has been developed.

McKernel establishes the foundation of a new architecture of learning that allows to obtain large-scale non-linear classification combining lightning kernel expansions and a linear classifier. It travails in the mini-batch setting working analogously to Neural Networks. We show the validity of our method through extensive experiments on MNIST and FASHION MNIST [3].

We also propose a new architecture to reduce over-parametrization in Neural Networks. It introduces an operand for rapid computation in the framework of Deep Learning that leverages learned weights. The formalism is described in detail providing both an accurate elucidation of the mechanics and the theoretical implications.

ZUSAMMENFASSUNG

Wir führen ein Framework ein, um Kernel-Approximationen in der Mini-Batch-Einstellung mit Stochastic Gradient Descent (SGD) als Alternative zu Deep Learning zu verwenden. Basierend auf Random Kitchen Sinks [1] bieten wir eine C++ Bibliothek für ML in großem Maßstab. Es enthält eine CPU-optimierte Implementierung des Algorithmus in [2], mit der ungefähre Kernel-Erweiterungen in logarithmisch linearer Zeit berechnet werden können. Der Algorithmus erfordert die Berechnung des Produkts der Matrizen Walsh Hadamard. Es wurde ein cachefreundlicher Fast Walsh Hadamard entwickelt, der eine überzeugende Geschwindigkeit erreicht und die aktuellen Methoden auf dem neuesten Stand der Technik übertrifft.

McKernel legt die Grundlage für eine neue Lernarchitektur, die es ermöglicht, eine nichtlineare Klassifizierung in großem Maßstab zu erhalten, die Blitzkernerweiterungen und einen linearen Klassifizierer kombiniert. Es funktioniert in der Mini-Batch-Einstellung analog zu neuronalen Netzen. Wir zeigen die Gültigkeit unserer Methode durch umfangreiche Experimente mit MNIST und FASHION MNIST [3].

Wir schlagen auch eine neue Architektur vor, um die Überparametrisierung in neuronalen Netzen zu reduzieren. Es wird ein Operand für die schnelle Berechnung im Rahmen von Deep Learning eingeführt, der gelernte Gewichte nutzt. Der Formalismus wird ausführlich beschrieben und bietet sowohl eine genaue Aufklärung der Mechanik als auch der theoretischen Implikationen.

ACKNOWLEDGEMENTS

I would like to thank all the people that have supported me through my studies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

MCKERNEL

1.1 INTRODUCTION

Kernel methods offer state-of-the-art estimation performance. They provide function classes that are flexible and easy to control in terms of regularization. However, the use of kernels in large-scale machine learning has been beset with difficulty. This is because using kernel expansions in large datasets is too expensive in terms of computation and storage. In order to solve this problem, [2] proposed an approximation algorithm based on Random Kitchen Sinks by [1], that speeds up the computation of a large range of kernel functions, allowing us to use them in big data. [4] describes the generalization of Random Features and potential effectiveness. Recent works on the topic build on it to propose state-of-the-art embeddings [5–8].

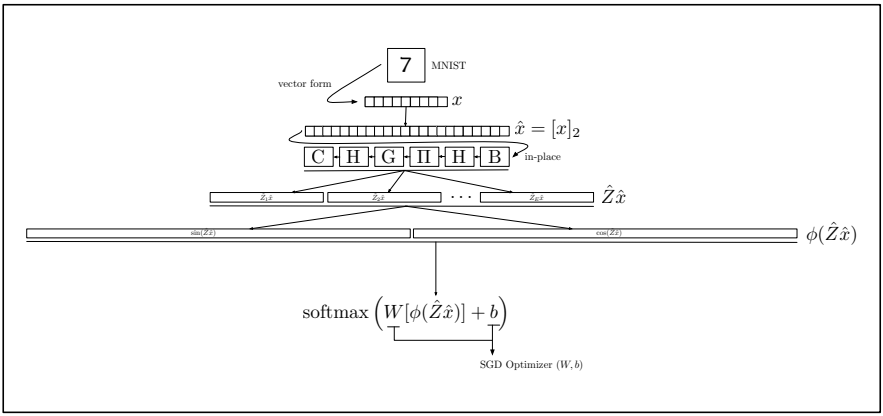

FIGURE 1.1: **Diagram of McKernel**. We visually describe softmax$(W\tilde{x} + b)$ where $\tilde{x} =$ mckernel(x). The original image is padded in form of long vector to the nearest power of 2, mapping $\hat{Z}$ is applied in-place. Calibration C defines the choice of Kernel. The tensor is expanded by the number of Kernel Expansions E building a network with high compositionality. Finally, use real feature map ϕ, Equation 1.9. SGD Optimizer finds appropriate weights W and bias b. Compute $\hat{Z}$ on-the-fly keeping same seed both for training and testing.

In this work, we go beyond former attempts [9–11] and propose a general framework in lieu of Deep Learning [12]. Our goal is to integrate current advances in Neural Networks but at the same time propose a well established theoretically sound background.

[13] claims the unreasonable effectiveness of mathematics in the natural sciences. [14] in the same way, states the unreasonable effectiveness of mathematics in machine learning. Kernel methods originate in rigorous mathematical treatment of the problem, while at the same time are incredibly effective.

At its heart, McKernel requires scalar multiplications, a permutation, access to trigonometric functions, and two Walsh Hadamard for implementation. The key computational bottleneck here is the Walsh Hadamard. We provide a fast, cache friendly SIMD (Single Instruction Multiple Data) oriented implementation that outperforms state-of-the-art codes such as Spiral [15]. To allow for very compact distribution of models, we use hashing and a Pseudo-random Permutation for portability. In this way, for each feature dimension, we only need one floating point number.

In summary, our implementation serves as a drop-in generator of features for linear methods where attributes are generated on-the-fly [16–21], such as for regression, classification, or two-sample tests. This obviates the need for explicit kernel computations, particularly on large amounts of data.

OUTLINE: Learning with Kernels is briefly introduced in Section 1.2. We begin then with a description of the feature construction McKernel in Section 1.3. Fast Walsh Hadamard is enunciated in Section 1.4. This is followed by a discussion of the computational issues for a SIMD implementation in Section 1.5. The concepts governing the API are described in Section 1.6. Large-scale Machine Learning by means of McKernel is discussed in Section 1.7. Concepts regarding TIKHONOV regulation are explained in Section 1.8. Experimental results can be found in Section 1.9. Section 1.10 gives a brief overall discussion.

1.2 LEARNING WITH KERNELS

The problem of learning [22–26] arises from the necessity to adapt a model $f : X \to Y$ to a given training set of data $S_n = (x_c, y_c)_{c=1}^n$, with $X \subset \mathbb{R}^n$

being closed and $Y \subset \mathbb{R}$, having f good properties of generalization.

Let $(x_c, y_c)_{c=1}^n$ be the data. Then, we pick a function $k_x(x') = k(x, x')$ symmetric, positive definite and continuous on $X \times X$. And set $f : X \to Y$ such that

$$f(x) = \sum_{z=0}^{n} t_z k_{x_z}(x), \tag{1.1}$$

where $t = (t_1, \ldots, t_n) \in \mathbb{R}^n$ and

$$(n\gamma I + K)t = y, \tag{1.2}$$

where matrix I is the identity, K is the matrix square positive definite with elements $k_{c,r} = k(x_c, x_r)$ and $\gamma > 0$ in $\mathbb{R}$.

It turns out this linear system of equations in n variables is well-posed as K is positive and $(n\gamma I + K)$ is strictly positive.

The intuition behind this algorithm, for instance given the Gaussian

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right), \tag{1.3}$$

is that we approximate the unknown function by a weighted superposition of Gaussians, each centered at location x_c of one of the n examples. The weight t_c of each Gaussian is chosen to minimize the error on the training set. The σ of the Gaussian, together with γ, controls the degree of smoothing, of noise tolerance and generalization.

[14] proposes a generalization of this framework by the use of invariants. Equation 1.2 becomes

$$(n\gamma I + VK)t = Vy, \tag{1.4}$$

where V takes into account mutual positions of observed vectors and elements $V(c, z)$ can be computed for high-dimensional problems as follows

$$V(c, z) = \sum_{k=1}^{d} \left(t_k - \max(x_c^k, x_z^k)\right), \tag{1.5}$$

with $0 \leq x_k \leq t_k$. Matrix V is a symmetric non-negative matrix. The main idea is to take into account that the desired decision rule is related to the conditional probability function of the observations.

1.3 MCKERNEL

Kernel methods work by defining a kernel function $k(x, x')$ on a domain X. We can write k as inner product between feature maps, as follows

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{1.6}$$

for some suitably chosen ϕ. Random Kitchen Sinks [1] approximate this mapping of features ϕ by a FOURIER expansion in the case of radial basis function (RBF), scilicet whenever $k(x, x') = \kappa(x - x')$. This is possible since the FOURIER transform diagonalizes the corresponding integral operator. This leads to

$$k(x, x') = \int \exp(i\langle w, x \rangle) \exp(-i\langle w, x' \rangle) d\rho(w) \tag{1.7}$$

for some L_2 measurable function $\rho(\omega) \geq 0$ that is given by the FOURIER transform of κ. Random Kitchen Sinks exploit this by replacing the integral by sampling $\omega \sim \rho(\omega)/\|\rho\|_1$. This allows for finite dimensional expansions but it is costly due to the large number of inner products required. [2] resolves this for rotationally invariant κ by providing a fast approximation of the matrix $W := [\omega_1, \dots \omega_n]$.

This is best seen for the RBF kernel, Equation 1.3. Since FOURIER transforms of Gaussians are Gaussians, albeit with inverse covariance, it follows that $\rho(\omega) \propto \exp\left(-\frac{\sigma^2}{2}\|\omega\|^2\right)$ and that W contains random variables independent and identically distributed (i.i.d.) Gaussian. It is this matrix that McKernel approximates via

$$\hat{Z} := \frac{1}{\sigma\sqrt{n}} CHG\Pi HB. \tag{1.8}$$

Here C, G and B are diagonal matrices, Π is a random permutation matrix and H is the Hadamard. Whenever the number of rows in W exceeds the dimensionality of the data, we can simply generate multiple instances of $\hat{Z}$, drawn i.i.d., until the required number of dimensions is obtained.

BINARY B. This is a matrix with entries $B_{kk} \in \{\pm 1\}$, drawn from the uniform distribution. To avoid memory footprint, we simply use Murmurhash as function of hashing and extract bits from $h(k, x)$ with $x \in \{0, \dots N\}$.

HADAMARD H. This matrix is iteratively composed of $H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}$. It is fixed and matrix-vector products are carried out efficiently in $O(n \log n)$ time using the Fast Walsh Hadamard. We will discuss implementation details for a fast variant in Section 1.5.

PERMUTATION Π. We generate a random permutation using the FISHER-YATES shuffle. That is, given a list $L = \{1, \dots n\}$ we generate permutations recursively as follows: pick a random element from L. Use this as the image of n and move n to the position where the element was removed. The algorithm runs in linear time and its coefficients can be stored in $O(n)$ space. Moreover, to obtain a deterministic mapping, replace the generator of random numbers with calls to the function of hashing.

GAUSSIAN G. This is a matrix diagonal with entries i.i.d. Gaussian. We generate the random variates using the BOX-MULLER transform [27] while substituting the generator of random numbers by calls to the function of hashing to allow us to recompute the values at any time without the need to store random numbers.

CALIBRATION C. This is a random scaling operator whose behavior depends on the type of kernel chosen, such as the RBF MATÉRN Kernel, the RBF Kernel or any other radial spectral distribution [28].

Ultimately, compute the feature pairs by assigning

$$[\cos(\hat{Z}x), \sin(\hat{Z}x)]. \tag{1.9}$$

In particular, McKernel computes the features by using the real version of the complex feature map ϕ in [1]. SIMD vectorized instructions and cache locality are used to increase speed performance. These allow a speed improvement of 18x times for a 2^{24} dimension input matrix.

1.4 fast walsh hadamard

Applications of the WALSH HADAMARD transform range across several areas, including Machine Learning [29, 30] and Computer Vision [31, 32].

A naïve implementation results in complexity $O(n^2)$. A divide-and-conquer approach for this task that runs in time $O(n \log n)$ can be derived as follows.

We define the matrix Hadamard H_n

$$H_0 = [1] \tag{1.10}$$

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}, \tag{1.11}$$

with dimension $2^n \times 2^n$.

We want to compute the product of matrix vector $H_n \cdot c$, being $c = (c_o, c_1)$ where $|c_0| = |c_1| = \frac{|c|}{2}$,

$$H_n \cdot c = \begin{bmatrix} H_{n-1}c_0 + H_{n-1}c_1 \\ H_{n-1}c_0 - H_{n-1}c_1 \end{bmatrix}. \tag{1.12}$$

Hence, we only need to compute recursively $H_{n-1}c_0$ and $H_{n-1}c_1$ to obtain $H_n \cdot c$ via additions and subtractions. The running time is

$$T(n) = 2T(n/2) + O(n), \tag{1.13}$$

that gives $T(n) = O(n \log n)$.

In McKernel we generalize this approach to compute the resulting matrix Hadamard from a hard-coded specific-size routine.

1.5 implementation of fast walsh hadamard

A key part of the library is an efficient implementation of Fast Walsh Hadamard. In particular, McKernel offers considerable improvement over Spiral, due to automatic code generation, the use of SIMD intrinsics (SSE2 using 128 bit registers) and loop unrolling. This decreases the memory

overhead.

McKernel proceeds with vectorized sums and subtractions iteratively for the first $\frac{n}{2^z}$ input vector positions (where n is the length of the input vector and z the iteration starting from 1), computing the intermediate operations of the COOLEY-TUKEY algorithm till a small routine Hadamard that fits in cache. Then the algorithm continues in the same way but starting from the smallest length and doubling on each iteration the input dimension until the whole Fast Walsh Hadamard is done in-place.

For instance, on an intel i5-4200 CPU @ 1.60GHz laptop the performance obtained can be observed in Figure 1.2.

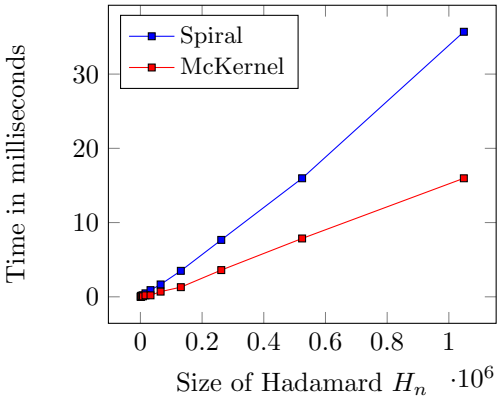

FIGURE 1.2: **Comparison of Fast Walsh Hadamard**. McKernel (red) outperforms Spiral (blue) across the range of arguments.

Our code outperforms Spiral consistently throughout the range of arguments, see Table 1.1. Furthermore, Spiral needs to precompute trees and by default can only perform the computation up to matrix size $n = 2^{20}$. On the other hand, our implementation works for any given size since it computes the high-level partitioning dynamically.

1.6 API DESCRIPTION

The API follows the design pattern in factory. That is, while the object McKernel is fairly generic in terms of computation, we have a factory that

$\lvert H_n \rvert$	McKernel $t(ms)$	Spiral $t(ms)$
1024	0	0.03
2048	0.03	0.07
4096	0.1	0.17
8192	0.07	0.2
16384	0.2	0.47
32768	0.2	0.9
65536	0.7	1.67
131072	1.3	3.5
262144	3.6	7.67
524288	7.86	15.97
1048576	15.97	35.7

Table 1.1: Numeric Comparison of Fast Walsh Hadamard.

acts as a means of instantiating the parameters according to pre-specified sets of parameters, e.g. a RBF Kernel or a RBF MATÉRN Kernel. The so-chosen parameters are deterministic, given by the values of a function of hashing. The advantage of this approach is that there is no need to save the coefficients generated for McKernel when deploying the functions.

McKernel is integrated into a fully-fledged C++ DL framework that lets the user experiment, among other things, with dropout, convolutions, different activation functions, layer normalization, maxpooling, L1 and L2 regularization, gradient clipping, autoencoders, residual blocks, SGD optimization with momentum and dataset loading. That said, it also includes some classical algorithms for learning such as linear and logistic regression.

1.6.1 *Customizing McKernel*

For instance, to generate each C_{kk} entry for RBF MATÉRN Kernel we draw t i.i.d. samples from the n-dimensional unit ball S_n, add them and compute its Euclidean norm.

To draw efficiently samples from S_n we use the algorithm provided below.

Let $X = (X_1, \ldots, X_n)$ be a vector of i.i.d. random variables drawn from $N(0,1)$, and let $||X||$ be the Euclidean norm of X, then $Y = \left(\frac{X_1}{||X||}, \ldots, \frac{X_n}{||X||}\right)$ is uniformly distributed over the n-sphere, where Y is the projection of X onto the surface of the n-dimensional sphere. To draw uniform random variables in the n-ball, we multiply Y by $U^{1/n}$ where $U \sim U(0,1)$. This can be proved as follows: Let $Z = (Z_1, \ldots, Z_n)$ be a random vector uniformly distributed in the unit ball. Then, the radius $R = ||Z||$ satisfies $P(R \leq r) = r^n$. Now, by the inverse transform method we get $R = U^{1/n}$. Therefore to sample uniformly from the n-ball the following algorithm is used:

$$Z = rU^{1/n}\frac{X}{||X||}. \tag{1.14}$$

1.7 LARGE-SCALE MACHINE LEARNING

We introduce McKernel as an alternative to Deep Learning, where we argue that current techniques of Neural Networks are surrogates to very large kernel expansions, where optimization is done in a huge parameter space where the majority of learned weights are trivial to the actual problem statement.

We present here the idea that current developments in very deep neural networks can be achieved while drastically reducing the number of parameters learned. We build on the work in [1] and [2] to expand its scope to mini-batch training with SGD Optimizer. Our concern is to demonstrate that we are able to get the same gains obtained in Deep Learning by the use of McKernel and a linear classifier but with a behemoth kernel expansion.

Current research in Neural Networks is over-optimizing parameters that are indeed not informative for the problem to solve. That is, we pioneer the notion that Deep Learning is learning the inner parameters of a very large kernel expansion and, in the end, is doing a brute-force search of the appropriate kernel k thats fits well the data.

Observe that McKernel generates pseudo-random numbers by the use of hashing which is key for large-scale data. It allows to obtain a deterministic behavior and at the same time to load the weights on both training and testing without the need to actually store the matrices. As a further matter,

it is crucial for distributed computation.

Notice here that the fact that we can increase the number of kernel expansions building highly hierarchical networks, see Equations 2.14 and 1.9, gives the property of compositionality to McKernel. Namely, the theoretical guarantee to avoid the curse of dimensionality [33, 34].

We can behold that McKernel is corresponding to networks of the form

$$G(x) = \sum_{k=1}^{N} a_k \exp\left(-|x - x_k|^2\right), \qquad x \in \mathbb{R}^d. \tag{1.15}$$

In the following section we build on these ideas to propose some very simple examples to illustrate the essence of the problem and the solution proposed.

1.8 REGULARIZATION TIKHONOV

Let H be the hypothesis space of functions, in the problem of Empirical Risk Minimization (ERM) we want to find $f \in H$ that minimizes

$$\frac{1}{n} \sum_{c=1}^{n} (f(x_c) - y_c)^2. \tag{1.16}$$

This problem is in general ill-posed, depending on the choice of H. Following Tikhonov [35, 36] we minimize instead over the hypothesis space H_K, the regularized functional

$$\frac{1}{n} \sum_{c=1}^{n} (f(x_c) - y_c)^2 + \lambda ||f||_K^2, \tag{1.17}$$

where $||f||_K^2$ is the norm in H_K - the REPRODUCING KERNEL HILBERT Space defined by the kernel K.

In general, under the TIKHONOV regularization scheme that follows,

$$\min_{w \in \mathbb{R}^D} \hat{E}(f_w) + \lambda ||w||^2, \tag{1.18}$$

where $||w||^2$ is the regularizer and controls the stability of the solution and λ balances the error term and the regularizer. Different classes of methods

are determined by the appropriate choice of loss function in Equation 1.18. Here we consider

$$\hat{E}(f_w) = \frac{1}{n} \sum_{c=1}^{n} l(y_c, f_w(x_c)) \tag{1.19}$$

with loss function l defined as

$$l(y, f_w(x)) = \log(1 + \exp(-y f_w(x))), \tag{1.20}$$

videlicet Logistic Regression.

Considering the logistic loss is differentiable and that we are in a large-scale setting a reasonable candidate to compute a minimizer is the Stochastic Gradient Descent (SGD),

$$w_{t+1} = w_t - \gamma \Delta g_{c_t}(w_t), \tag{1.21}$$

where c_t denotes a stochastic sequence of indices and γ is the learning rate.

In this line of argument, [37–39] state that local minimization is well posed in Deep Learning using SGD.

Augmenting the number of kernel expansions, and thus the representational power of the model, gives a degree of over-parametrization. That is to say, we increase the size of the network to fit the training data. Given these constraints, BÉZOUT theorem argues the existence of a large number of degenerate global minimizers with zero empirical error, that are very likely to be found by SGD that will in addition select with higher probability the most robust zero-minimizer [40].

1.9 EMPIRICAL ANALYSIS AND EXPERIMENTS

We generalize the use of McKernel in mini-batch with SGD Optimizer, Figure 1.1, drastically reducing the number of parameters that need to be learned to achieve comparable state-of-the-art results to Deep Learning.

What is more, current breakthroughs in Neural Networks can be easily derived from Equation 2.14. Say for instance, Batch Normalization [41] can be obtained from the normalizing factor. Or for example, the use of ensembles [42] and multi-branch architectures [43] to improve performance can

be seen on Figure 1.1, as it follows from the increase on Kernel Expansions. Not only that, but increasing the number of Kernel Expansions has another great property; data augmentation [44, 45], which has recently seen a lot of interest. Its importance follows directly from the construction of McKernel, take the data, apply slightly (randomized) different functions to it to create new high-dimensional samples that will aid the process of learning. It also explains the need for backpropagation [46]: certain types of data will work better for certain kernels, so it may be necessary to learn the appropriate Calibration C and G that fit well the data. Besides, learning B acts as mechanism of attention [47–49]. Further, dropout [50] follows directly from the use of the Subsampled Randomized Hadamard. Additionally, research on finding alternate activation functions [51–54] can be deduced from looking for different mappings in Equation 1.9.

Note here that the number of parameters to be estimated is of the order of thousands, proportional to the number of classes (depending on the size of the input image and the number of kernel expansions),

$$C \cdot (2 \cdot [S]_2 \cdot E + 1), \tag{1.22}$$

where C is the number of classes, $[\cdot]_2$ is an operator that returns the next power of 2, S is the size of the input samples and E is the number of Kernel Expansions. A drastic reduction compared to Neural Networks, while achieving comparable performance. Training time is therefore severely reduced and SVM kernel like learning can be achieved at scale.

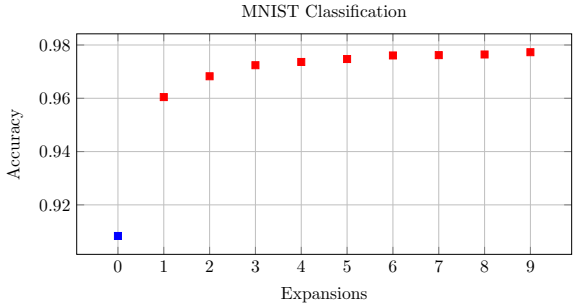

FIGURE 1.3: **MNIST Classification.** Logistic Regression (blue) and RBF MATÉRN (red) with increasing number of Kernel Expansions. 32768 samples of training data and 8192 samples of testing data are used in learning. RBF MATÉRN hyper-parameters, $\sigma = 1.0$, $t = 40$. Seed 1398239763, learning rate $\gamma = 0.001$ and batch size 10. LR learning rate 0.01. Number of epochs 20.

SGD Optimizer finds W and b in

$$\text{softmax}\left(W[\phi(\hat{Z}\hat{x})] + b\right), \tag{1.23}$$

where $\phi = (\sin(\cdot), \cos(\cdot))$, $\hat{x} = [x]_2$. Namely, it minimizes the loss l in Equation 1.20.

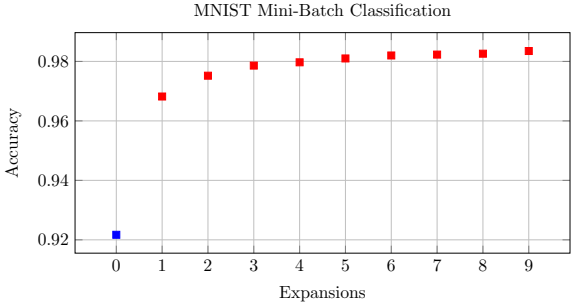

FIGURE 1.4: **MNIST Mini-Batch Classification.** Logistic Regression (LR) (blue) and RBF MATÉRN (red) with increasing number of Kernel Expansions. 60000 samples of training data and 10000 samples of testing data are used in learning. RBF MATÉRN hyper-parameters, $\sigma = 1.0$, $t = 40$. Seed 1398239763, learning rate $\gamma = 0.001$ and batch size 10. LR learning rate 0.01. Number of epochs 20.

Figures 1.3 and 1.4 show RBF MATÉRN, softmax$(W\tilde{x} + b)$ where $\tilde{x} =$ mckernel(x), performance compared to logistic regression, softmax$(Wx + b)$, in full-batch and mini-batch on MNIST, respectively. A fixed seed is used to obtain deterministic reproducible behavior. In full-batch, the number of samples for train and test is rounded to the nearest power of 2 due to algorithm constraint.

The same kind of intuition that applies to Neural Networks, where the deeper the network, the better the results, holds. But this time depending on the number of kernel expansions used.

FASHION MNIST [3] is similar in scope to MNIST but relatively more difficult. Instead of classifying digits, we focus now on the task of fashion. It consists on ten classes of clothing; T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag and Ankle boot.

FIGURE 1.5: **FASHION MNIST Mini-Batch Classification.** Logistic Regression (LR) (blue) and RBF MATÉRN (red) with increasing number of Kernel Expansions. 60000 samples of training data and 10000 samples of testing data are used in learning. RBF MATÉRN hyper-parameters, $\sigma = 1.0$, $t = 40$. Seed 1398239763, learning rate $\gamma = 0.001$ and batch size 10. LR learning rate 0.01. Number of epochs 20.

Figure 1.5 shows RBF MATÉRN performance compared to logistic regression in mini-batch. Comparable state-of-the-art performance to Deep Learning is achieved. The model presents a similar behavior to the one seen in MNIST dataset. McKernel performs analogously to modern techniques in Neural Networks in this highly non-linear problem of estimation.

1.10 DISCUSSION

In this manuscript we provide a new framework of learning and illustrate with two examples that achieves comparable state-of-the-art performance to Neural Networks, proposing a new way to understand Deep Learning, as a huge kernel expansion where optimization is only performed over the parameters that are actually relevant to the problem at-hand. At the same time, a new methodology to build highly compositional networks for Large-scale Machine Learning is introduced.

We account for both the theoretical underpinnings and the practical implications to establish the building blocks of a unifying theory of learning.

2

DR. OF CROSSWISE

2.1 INTRODUCTION

Re-thinking how Deep Networks operate at large scale using current techniques is difficult. We build on the work in [12] to propose a fast variant [55] named Dr. of Crosswise [1]. Our approach does not lose the ability to generalize while reduces vastly the number of parameters and improves training and testing speed.

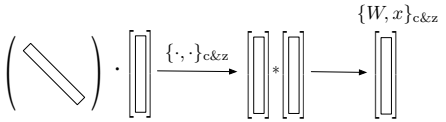

FIGURE 2.1: Dr. of Crosswise. Visual description of c&z.

Dr. of Crosswise substitutes common products matrix-vector in the architectures of Neural Networks by the use of simplified one-dimensional multiplication of tensors. Namely, it establishes a framework of learning where learned weight matrices are diagonal and optimized weights are considerably reduced. We introduce a new operation between vectors to allow for a fast computation.

The structure of learning using matrices diagonal follows directly from the construction McKernel in [12] based on [1, 2, 56], which can be understood as a GAUSSIAN network [33, 34] of the form

$$G(x) = \sum_{k=1}^{N} a_k \exp\left(-|x - x_k|^2\right), \qquad x \in \mathbb{R}^d. \tag{2.1}$$

We extend this idea to a more general framework, unconstrained in the sense that we no longer consider a form Gaussian, Equation 2.1, but any possible non-linearity (for instance ReLU). That is to say, Deep Learning.

1 Code is available at https://www.github.com/curto2/dr_of_crosswise/

2.2 DEEP LEARNING

Successes in Neural Networks range across all domains, from Natural Language Processing [57–59] to Computer Vision [60–71], going through Automatic Structural Learning [72] or Data Augmentation [45]. Techniques to improve aspects of current architectures have been widely explored [73–75]. Significant progress has also been made by combining Deep Learning for extraction of features with Reinforcement Learning [76, 77].

2.3 DOCTOR OF CROSSWISE

The way to go on Deep Learning entangles the following formalism

$$y = \max(Wx + b, 0) \tag{2.2}$$

where matrix W and vector b are the weights and biases learned by assignment of credit [46, 78], videlicet backpropagation.

Dr. of Crosswise substitutes products matrix-vector by

$$y = \max(\hat{W}x + b, 0) \tag{2.3}$$

where $\hat{W}$ is a matrix with form diagonal

$$\hat{W} = \begin{bmatrix} W_1 & 0 & \dots & 0 \\ 0 & W_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & W_n \end{bmatrix}. \tag{2.4}$$

We can now factorize Equation 2.3 by the use of a new operand between vectors $\{\cdot, \cdot\}_{\text{c\&z}}$

$$y = \max(\{c, x\}_{\text{c\&z}} + b, 0) \tag{2.5}$$

where z is a vector that holds the diagonal of $\hat{W}$, $c = [W_1 \dots W_N]$.

The operand defined by $\{\cdot,\cdot\}_{\text{c\&z}}$ is similar to a HADAMARD product and does the following operation. Given vectors c and x it computes

$$\{c, x\}_{\text{c\&z}} = \begin{bmatrix} c_1 x_1 \\ c_2 x_2 \\ \vdots \\ c_n x_n \end{bmatrix} \tag{2.6}$$

where c holds the non-zero elements of a matrix diagonal.

Note that Equation 2.5 is equivalent to 2.3. In other words, the new operator c&z helps factorize products between matrix diagonal and vector in form vector-vector. All the notation is preserved but for a change in the definition of the product.

We can understand this new operation between vectors as a product component-wise that scales each component by a given factor, Figure 2.1.

Furthermore, the factors that need to be learned reduce from $n \times n$ to n for each given layer, a momentous reduction of learned parameters and time of computation. Considering the compositionality of the problem, where architectures are built as a stack of many of these formalisms, Equation 2.2, the improvements are considerably remarkable.

2.3.1 *Extension to Higher-order*

If we now consider the setting where we have multiple input data and the need to expand it to higher-dimensional spaces from layer to layer, as it is normally done in Deep Learning. We have to consider several facts. Given a matrix W of dimension $M \times N$ and input vector x of dimension N we will generate an output vector with size proportional to N. W will be substituted by a set of matrices diagonal whose cardinal will be the closest multiplicity of N to M, see Figure 2.2. In this way, we will need a product that operates element-wise between the elements of each matrix diagonal and the input vector. Considering a varying number of input vectors, this mathematical operation is very close to a KHATRI RAO product, which is the matching columnwise KRONECKER product [79].

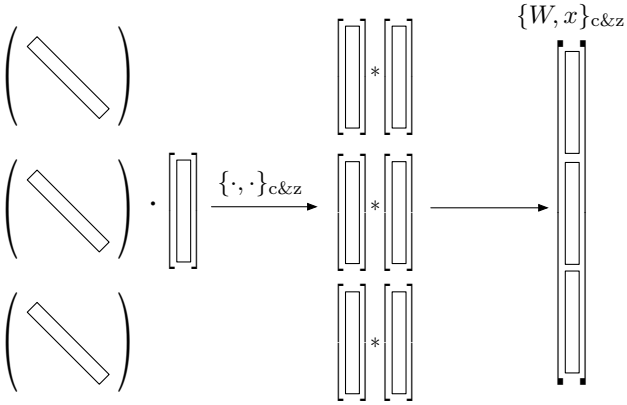

FIGURE 2.2: Dr. of Crosswise. Visual description of c&z.

These products between matrices have useful properties:

$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{2.7}$$

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger \tag{2.8}$$

$$A \odot B \odot C = (A \odot B) \odot C = A \odot (B \odot C) \tag{2.9}$$

$$(A \odot B)^T (A \odot B) = A^T A * B^T B \tag{2.10}$$

$$(A \odot B)^\dagger = ((A^T A) * (B^T B))^\dagger (A \odot B)^T \tag{2.11}$$

where $\odot$ denotes KHATRI RAO product, $\otimes$ specifies KRONECKER product and $*$ means HADAMARD product. $A\dagger$ is the MOORE PENROSE pseudo-inverse of A.

Take special note on the fact that for example, to transform an input vector of size 4, to an output vector of size 8, in the standard formulation of Neural Networks we have to learn 32 weights. If we consider Dr. of Crosswise, the number of learned parameters to do exactly the same operation reduces to 8. More importantly, the number of multiplications and additions also drastically lowers.

2.4 RATIONALE

We start with an exordium on Kernel Methods [14, 22]. Let X be a measure space with $k : L^2(X \times X) \to \mathbb{R}$. We name k a kernel if, and only if, there is some feature map $\phi : X \to \mathbb{H}$ into a separable HILBERT space $\mathbb{H}$ such that

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathbb{H}}. \tag{2.12}$$

In other words, k is a kernel if, and only if, for some space $\mathbb{H}$ and map ϕ the following diagram commutes:

$$\begin{array}{ccc} X \times X & & \\ \phi \downarrow & \searrow^{k} & \\ \mathbb{H} \times \mathbb{H} & \xrightarrow[\langle \cdot, \cdot \rangle_{\mathbb{H}}]{} & \mathbb{R}. \end{array}$$

Random Kitchen Sinks [1] approximate this mapping of features ϕ by a FOURIER expansion in the case of RBF kernels

$$k(x, x') = \int \exp\left(i\langle w, x \rangle\right) \exp\left(-i\langle w, x' \rangle\right) d\rho(w). \tag{2.13}$$

[2, 28] present a fast approximation of the matrix W. Recent works on the matter [5–7, 80] use this technique to extract meaningful features.

Dr. of Crosswise is motivated by the construction McKernel in [12], where Deep Learning and Kernel Methods are unified by extending the use of the approximation of W, $\hat{Z}$, to a SGD optimization setting

$$\hat{Z} := \frac{1}{\sigma\sqrt{n}} CHG\Pi HB. \tag{2.14}$$

Here C, G and B are matrices diagonal, Π is a random matrix of permutation and H is the Hadamard. Whenever the number of rows in W exceeds the dimensionality of the data, simply generate multiple instances of $\hat{Z}$, drawn i.i.d., until the required number of dimensions is obtained.

The key idea behind it is that the FOURIER transform diagonalizes the integral operator.

This can be best seen as follows. Considering the operator of convolution working on the complex exponential $f(z) = \exp(ikz)$

$$g(z) = (f * r)(z) = \int_{-\infty}^{\infty} f(\tau) r(z - \tau) d\tau. \tag{2.15}$$

Then

$$g(z) = \lambda f(z) \tag{2.16}$$

where $\lambda = R(k)$ and R is the FOURIER transform of r.

We build on these ideas to pioneer a framework of Deep Learning where the formalism used entangles matrices diagonal.

3

GENERATIVE ADVERSARIAL NETWORKS

3.1 INTRODUCTION

Developing a Generative Adversarial Network (GAN) [60] able to produce good quality high-resolution samples from images has important applications [80–95] including image inpainting, 3D data, domain translation, video synthesis, image edition, semantic segmentation and semi-supervised learning.

FIGURE 3.1: **HDCGAN Synthetic Images**. A set of random samples. Our system generates high-resolution synthetic faces with an extremely high level of detail. HDCGAN goes from random noise to realistic synthetic pictures that can even fool humans. To demonstrate this effect, we create the Dataset of Curtò & Zarzà, the first GAN augmented dataset of faces.

In this paper, we focus on the task of face generation, as it gives GANs a huge space of learning attributes. In this context, we introduce the Dataset of Curtò & Zarzà [71], a well-balanced collection of images containing 14,248 human faces from different ethnical groups and rich in a wide range of learnable attributes, such as gender and age diversity, hair-style and pose variation or presence of smile, glasses, hats and fashion items. We also ensure the presence of changes in illumination and image resolution. We propose to use Curtò as de facto approach to empirically test the distribution learned by a GAN, as it offers a challenging problem to solve, while keeping the number of samples, and therefore training time, bounded. It can also be used as a drop-in substitute of MNIST for simple tasks of classification, say for instance using labels of ethnicity, gender, age, hair

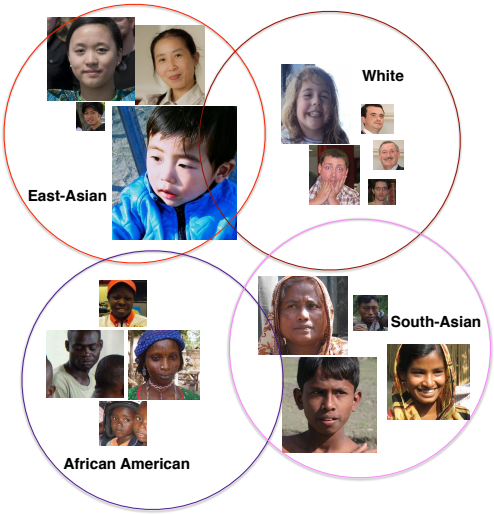

FIGURE 3.2: **Samples of Curtò**. A set of random instances for each class of ethnicity: African American, White, East-asian and South-asian. See Table 3.1 for numerics.

style or smile. It ships with scripts in TensorFlow and Python that allow benchmarks of classification. A set of random samples can be seen in Figure 3.2.

Despite improvements in GANs training stability [67, 96, 97] and specific-task design during the last years, it is still challenging to train GANs to generate high-resolution images due to the disjunction in the high dimensional pixel space between supports of the real image and implied model distributions [98, 99].

Our goal is to be able to generate indistinguishable sample instances using face data to push the boundaries of GAN image generation that scale well to high-resolution images (such as 512×512) and where context information is maintained.

In this sense, Deep Learning has a tremendous appetite for data. The question that arises instantly is, what if we were able to generate additional realistic data to aid learning using the same techniques that are later used to train the system. The first step would then be to have an image

generation tool able to sample from a very precise distribution (e. g. faces from celebrities) which instances resemble or highly correlate with real sample images of the underlying true distribution. Once achieved, what is desirable and comes next is that these generated image points not only fit well into the original distribution set of images but also add additional useful information such as redundancy, different poses or even generate highly-probable scenarios that would be possible to see in the original dataset but are actually not present.

Current research trends link Deep Learning and Kernel Methods to establish a unifying theory of learning [12, 55]. The next frontier in GANs would be to achieve learning at scale with very few examples. To achieve the former goal this work contributes in the following:

- Network that achieves compelling results and scales well to the high-resolution setting where to the best of our knowledge the majority of other variants are unable to continue learning or fall into mode collapse.

- New dataset targeted for GAN training, Curtò, that introduces a wide space of learning attributes. It aims to provide a well-posed difficult task while keeping training time and resources tightly bounded to spearhead research in the area.

3.2 PRIOR WORK

Generative image generation is a key problem in Computer Vision and Computer Graphics. Remarkable advances have been made with the renaissance of Deep Learning. Variational Autoencoders (VAE) [88, 100] formulate the problem with an approach that builds on probabilistic graphical models, where the lower bound of data likelihood is maximized. Autoregressive models (scilicet PIXELRNN [101]), based on modeling the conditional distribution of the pixel space, have also presented relative success generating synthetic images. Lately, Generative Adversarial Networks (GANs) [60, 65, 87, 102–105] have shown strong performance in image generation. However, training instability makes it very hard to scale to high-resolution (256×256 or 512×512) samples. Some current works on the topic pinpoint this specific problem [106], where conditional image generation is also tackled while other recent techniques [67, 70, 107–111] try to stabilize training.

3.3 DATASET OF CURTÒ & ZARZÀ

Curtò contains 14,248 faces balanced in terms of ethnicity: African American, East-Asian, South-Asian and White. Mirror images are included to enhance pose variation and there is roughly 25% per image class. Attribute information, see Table 3.1, is composed of thorough labels of gender, age, ethnicity, hair color, hair style, eyes color, facial hair, glasses, visible forehead, hair covered and smile. There is also an extra set with 3,384 cropped labeled images of faces, ethnicity white, no mirror samples included, see Column 4 in Table 3.1 for statistics. We crawled Flickr to download images of faces from several countries that contain different hair-style variations and style attributes. These images were then processed to extract 49 facial landmark points using [112]. We ensure using Mechanical Turk that the detected faces are correct in terms of ethnicity and face detection. Cropped faces are then extracted to generate multiple resolution sources. Mirror augmentation is performed to further enhance pose variation.

Curtò introduces a difficult paradigm of learning, where different ethnical groups are present, with very varied fashion and hair styles. The fact that the photos are taken using non-professional cameras in a non-controlled environment, gives us multiple poses, illumination conditions and camera quality.

TABLE 3.1: Dataset of Curtò & Zarzà. Attribute Information. Descending order of class instances by number of samples, Column 3.

Attribute	Class	# Samples	# Extra
Age	Early Adulthood	3606	966
	Middle Aged	2954	875
	Teenager	2202	178
	Adult	1806	565
	Kid	1706	85
	Senior	1102	402
	Retirement	436	218
	Baby	232	14
Ethnicity	African American	4348	0
	White	3442	3384

	East Asian	3244	0
	South Asian	3214	0
Eyes Color	Brown	9116	2119
	Other	4136	875
	Blue	580	262
	Green	416	128
Facial Hair	No	12592	2821
	Light Mustache	466	156
	Light Goatee	444	96
	Light Beard	258	142
	Thick Goatee	168	39
	Thick Beard	166	68
	Thick Mustache	154	62
Gender	Male	7554	1998
	Female	6694	1386
Glasses	No	12576	2756
	Eyeglasses	1464	539
	Sunglasses	208	89
Hair Color	Black	8402	964
	Brown	3038	1241
	Other	1554	253
	Blonde	616	543
	White	590	347
	Red	48	36
Hair Covered	No	12292	3060
	Turban	1206	76
	Cap	722	237
	Helmet	28	11
Hair Style	Short Straight	5038	1642
	Long Straight	2858	857
	Short Curly	2524	287
	Other	2016	249

	Bald	1298	187
	Long Curly	514	162
Smile	Yes	8428	2118
	No	5820	1266
Visible Forehead	Yes	11890	3033
	No	2358	351

3.4 APPROACH

Generative Adversarial Networks (GANs) proposed by [60] are based on two dueling networks, Figure 3.3; Generator G and Discriminator D. In essence, the process of learning consists of a two-player game where D tries to distinguish between the prediction of G and the ground truth, while at the same time G tries to fool D by producing fake instance samples as closer to the real ones as possible. The solution to a game is called NASH equilibrium.

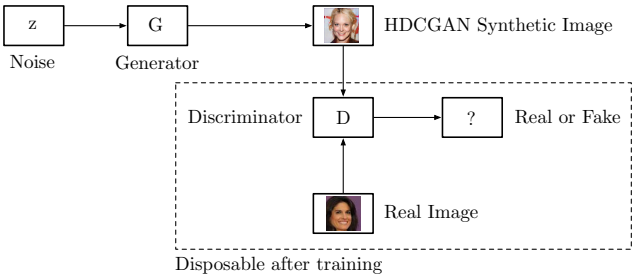

FIGURE 3.3: **Generative Adversarial Networks**. A two-player game between the Generator G and the Discriminator D. The dotted line denotes elements that will not be further used after the game stops, namely, end of training.

The min-max game entails the following objective function

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}}\left[\log D(x)\right] + \tag{3.1}$$

$$+\mathbb{E}_{z\sim p_z}\left[\log(1-D(G(z)))\right], \tag{3.2}$$

where x is a ground truth image sampled from the true distribution p_{data}, and z is a noise vector sampled from p_z (that is, uniform or normal distribution). G and D are parametric functions where $G : p_z \rightarrow p_{data}$ maps samples from noise distribution p_z to data distribution p_{data}.

The goal of the Discriminator is to minimize

$$L^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}}\left[\log D(x)\right] - \tag{3.3}$$

$$-\frac{1}{2}\mathbb{E}_{z\sim p_z}\left[\log(1 - D(G(z)))\right]. \tag{3.4}$$

If we differentiate it w.r.t $D(x)$ and set the derivative equal to zero, we can obtain the optimal strategy

$$D(x) = \frac{p_{data}(x)}{p_z(x) + p_{data}(x)}. \tag{3.5}$$

Which can be understood intuitively as follows. Accept an input, evaluate its probability under the distribution of the data, p_{data}, and then evaluate its probability under the generator's distribution of the data, p_z. Under the condition in D of enough capacity, it can achieve its optimum. Note the discriminator does not have access to the distribution of the data but it is learned through training. The same applies for the generator's distribution of the data. Under the condition in G of enough capacity, then it will set $p_z = p_{data}$. This results in $D(x) = \frac{1}{2}$, that is actually the NASH equilibrium. In this situation, the generator is a perfect generative model, sampling from $p(x)$.

As an extension to this framework, DCGAN [65] proposes an architectural topology based on Convolutional Neural Networks (CNNs) to stabilize training and re-use state-of-the-art networks from tasks of classification. This direction has recently received lots of attention due to its compelling results in supervised and unsupervised learning. We build on this to propose a novel DCGAN architecture to address the problem of high-resolution image generation. We name this approach HDCGAN.

3.4.1 *HDCGAN*

Despite the undoubtable success, GANs are still arduous to train, particularly when we use big images (e. g. 512×512). It is very common to see D beating G in the process of learning, or the reverse, ending in unrecognizable imagery, also known as mode collapse. Only when stable learning is achieved, the GAN structure is able to succeed in getting better and better results with time.

This issue is what drives us to carefully derive a simple yet powerful structure that leverages common problems and gets a stable and steady training mechanism.

Self-normalizing Neural Networks (SNNs) were introduced in [54]. We consider a neural network with activation function f, connected to the next layer by a weight matrix $\mathbf{W}$, and whose inputs are the activations from the preceding layer x, $y = f(\mathbf{W}x)$.

We can define a mapping g that maps mean and variance from one layer to mean and variance of the following layer

$$\begin{pmatrix} \mu \\ \nu \end{pmatrix} \longmapsto \begin{pmatrix} \tilde{\mu} \\ \tilde{\nu} \end{pmatrix} : \begin{pmatrix} \tilde{\mu} \\ \tilde{\nu} \end{pmatrix} = g \begin{pmatrix} \mu \\ \nu \end{pmatrix}. \tag{3.6}$$

Common normalization tactics such as batch normalization ensure a mapping g that keeps (μ, ν) and $(\tilde{\mu}, \tilde{\nu})$ close to a desired value, normally (0, 1).

SNNs go beyond this assumption and require the existence of a mapping $g : \Omega \longmapsto \Omega$ that for each activation y maps mean and variance from one layer to the next layer and at the same time have a stable and attracting fixed point depending on (ω, τ) in Ω. Moreover, the mean and variance remain in the domain Ω and when iteratively applying the mapping g, each point within Ω converges to this fixed point. Therefore, SNNs keep activations normalized when propagating them through the layers of the network.

Here (ω, τ) are defined as follows. For n units with activation x_c, $1 \leq c \leq n$ in the lower layer, we set n times the mean of the weight vector $w \in \mathbb{R}^n$ as $\omega := \sum_{c=1}^{n} w_c$ and n times the second moment as $\tau := \sum_{c=1}^{n} w_c^2$.

Scaled Exponential Linear Units (SELU) [54] is introduced as the choice of activation function in Feed-forward Neural Networks (FNNs) to construct a mapping g with properties that lead to SNNs.

$$selu(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha \exp^x - \alpha & \text{if } x \leq 0. \end{cases} \tag{3.7}$$

Empirical observation leads us to say that the use of SELU greatly improves the convergence speed on the DCGAN structure, however, after some iterations mode collapse and gradient explosion completely destroy training when using high-resolution images. We conclude that although SELU gives theoretical guarantees as the optimal activation function in FNNs, numerical errors in the GPU computation degrade its performance in the overall min-max game of DCGAN. To alleviate this problem, we propose to use SELU and BatchNorm [41] together. The motivation is that when numerical errors move $(\tilde{\mu}, \tilde{v})$ away from the attracting point that depends on $(\omega, \tau) \in \Omega$, BatchNorm will ensure it is close to a desired value and therefore maintain the convergence rate.

Experiments show that this technique stabilizes training and allows us to use fewer GPU resources, having steady diminishing errors in G and D. It also accelerates convergence speed by a great factor, as can be seen after some few epochs of training on CelebA [113] in Figure 3.8.

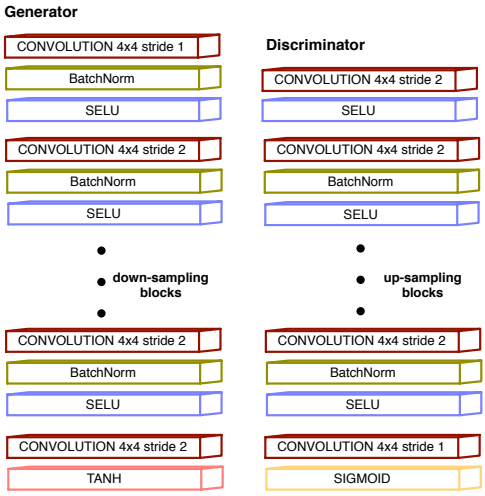

FIGURE 3.4: **HDCGAN Architecture**. Generator and Discriminator.

As SELU + BatchNorm (BS) layers keep mean and variance close to $(0,1)$ we get an unbiased estimator of p_{data} with contractive finite variance. These are very desirable properties from the point of view of an estimator as we are iteratively looking for a MVU (Minimum Variance Unbiased) criterion and thus solving MSE (Minimum Square Error) among unbiased estimators. Hence, if the MVU estimator exists and the network has enough capacity to actually find the solution, given a sufficiently large sample size by the Central Limit Theorem, we can attain NASH equilibrium.

HDCGAN Architecture is described in Figure 3.4. It differs from traditional DCGAN in the use of BS layers instead of ReLUs.

We observe that when having difficulty in training DCGAN, it is always better to use a fixed learning rate and instead increase the batch size. This is because having more diversity in training, gives a steady diminishing loss and better generalization. To aid learning, noise following a Normal $N(0,1)$ is added to both the inputs of D and G. We see that this helps overcome mode saturation and collapse whereas it does not change the distribution of the original data.

We empirically show that the use of BS induces SNNs properties in the GAN structure, and thus makes learning highly robust, even in the stark presence of noise and perturbations. This behavior can be observed when the zero-sum game problem stabilizes and errors in D and G jointly diminish, Figure 3.9. Comparison to traditional DCGAN, WASSERSTEIN GAN [114] and WGAN-GP [115] is not possible, as to date, the majority of former methods, such as [116], cannot generate recognizable results in image size 512×512, 24GB GPU memory setting.

Thus, HDCGAN pushes up state-of-the-art results beating all former DCGAN-based architectures and shows that, under the right circumstances, BS can solve the min-max game efficiently.

3.4.2 *Glasses*

We introduce here a key technique behind the success of HDCGAN. Once we have a good convergence mechanism for large input samples, that is a concatenation of BS layers, we observe that we can arbitrarily improve the final results of the GAN structure by the use of a Magnifying Glass

approach. Assuming our input length is $N \times M$, we can enlarge it by a constant factor, $\zeta_1 N \times \zeta_2 M$, which we call telescope, and then feed it into the network, maintaining the size of the convolutional filters untouched. This simple procedure works similar to how contact lenses correct or assist defective eyesight on humans and empowers the GAN structure to appreciate the inner properties of samples.

Note that as the input gets bigger so does the neural network. That is, the number of layers is implicitly set by the image size, see up-sampling and down-sampling blocks in Figure 3.4. For example, for an input size of 32 we have 4 layers while for an input size of 256 we have 7 layers.

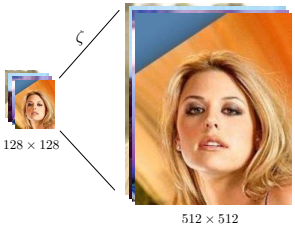

FIGURE 3.5: **Glasses on a set of samples from CelebA**. HDCGAN introduces the use of a Magnifying Glass approach, enlarging the input size by a telescope ζ.

We can empirically observe that BS layers together with Glasses induce high capacity into the GAN structure so that a NASH equilibrium can be reached. That is to say, the generator draws samples from p_{data}, which is the distribution of the data, and the discriminator is not able to distinguish between them, $D(x) = \frac{1}{2} \forall x$.

3.5 EMPIRICAL ANALYSIS

We build on DCGAN and extend the framework to train with high-resolution images using Pytorch. Our experiments are conducted using a fixed learning rate of 0.0002 and ADAM solver [117] with batch size 32 and 512×512 samples with the number of filters of G and D equal to 64.

In order to test generalization capability, we train HDCGAN in the newly introduced Curtò, CelebA and CelebA-hq.

Technical Specifications: 2 × NVIDIA Titan X, Intel Core i7-5820k@3.30GHz.

3.5.1 *Curtò*

The results after 150 epochs are shown in Figure 3.6. We can see that HDCGAN captures the underlying features that represent faces and not only memorizes training examples. We retrieve nearest neighbors to the generated images in Figure 3.7 to illustrate this effect.

FIGURE 3.6: **HDCGAN Example Results. Dataset of Curtò & Zarzà**. 150 epochs of training. Image size 512×512.

FIGURE 3.7: **Nearest Neighbors. Dataset of Curtò & Zarzà**. Generated samples in the first row and their five nearest neighbors in training (rows 2-6).

3.5.2 *CelebA*

CelebA is a large-scale dataset with 202,599 celebrity faces. It mainly contains frontal portraits and is particularly biased towards groups of ethnicity white. The fact that it presents very controlled illumination settings and good photo resolution, makes it a considerably easier problem than Curtò. The results after 19 epochs of training are shown in Figure 3.8.

FIGURE 3.8: **HDCGAN Example Results. CelebA**. 19 epochs of training. Image size 512×512. The network learns swiftly a clear pattern of the face.

In Figure 3.9 we can observe that BS stabilizes the zero-sum game, where errors in D and G concomitantly diminish. To show the validity of our method, we enclose Figure 3.10, presenting a large number of samples for epoch 39. We also attach a zoomed-in example to appreciate the quality and size of the generated samples, Figure 3.11. Failure cases can be observed in Figure 3.12.

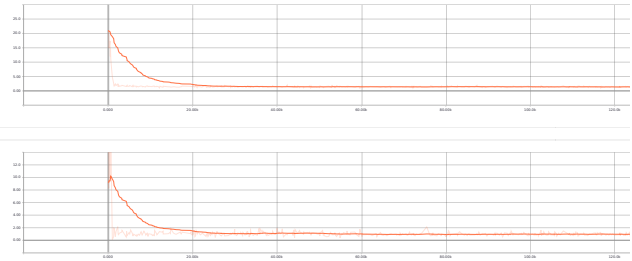

FIGURE 3.9: **HDCGAN on CelebA**. Error in Discriminator (top) and Error in Generator (bottom). 19 epochs of training.

FIGURE 3.10: **HDCGAN Example Results. CelebA**. 39 epochs of training. Image size 512×512. The network generates distinctly accurate and assorted faces, including exhaustive details.

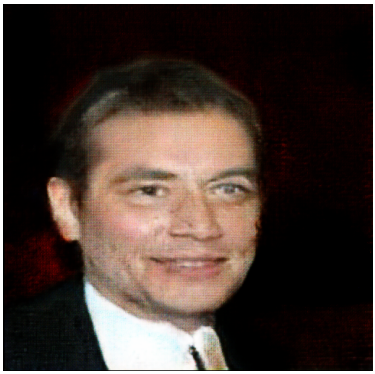

FIGURE 3.11: **HDCGAN Example Result. CelebA**. 39 epochs of training. Image size 512×512. 27% of full-scale image.

FIGURE 3.12: **HDCGAN Example Results. CelebA**. 39 epochs of training. Image size 512×512. Failure cases. The number of failure cases declines over time, and when present, they are of more meticulous nature.

Besides, to illustrate how fundamental our approach is, we enlarge Curtò with 4,239 unlabeled synthetic images generated by HDCGAN on CelebA, a random set can be seen in Figure 3.13.

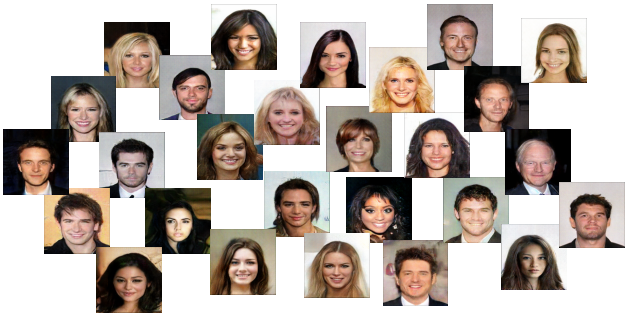

FIGURE 3.13: **HDCGAN Synthetic Images**. A set of random samples.

3.5.3 *CelebA-hq*

CelebA-hq in introduced in [70], a set of 30.000 high-definition images to improve training on CelebA. A set of samples generated by HDCGAN on CelebA-hq can be seen in Figures 3.1, 3.14 and 3.15.

FIGURE 3.14: **HDCGAN Example Results. CelebA-hq**. 229 epochs of training. Image size 512×512. The network generates superior faces, with great attention to detail and quality.

FIGURE 3.15: **HDCGAN Example Result. CelebA-hq**. 229 epochs of training. Image size 512×512. 27% of full-scale image.

To exemplify that the model is generating new bona fide instances instead of memorizing samples from the training set, we retrieve nearest neighbors to the generated images in Figure 3.16.

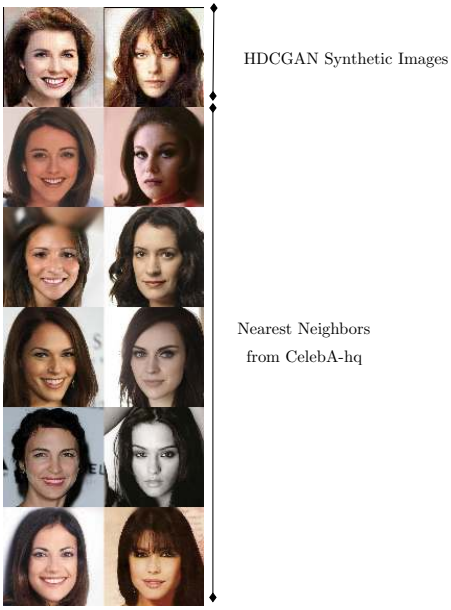

FIGURE 3.16: **Nearest Neighbors. CelebA-hq**. Generated samples in the first row and their five nearest neighbors in training (rows 2-6).

3.6 ASSESSING THE DISCRIMINABILITY AND QUALITY OF GENERATED SAMPLES

We build on previous image similarity metrics to qualitatively evaluate generated samples of generative models. The most effective of these is multi-scale structural similarity (MS-SSIM) [102]. We make comparison at resized image size 128×128 on CelebA. MS-SSIM results are averaged from 10,000 pairs of generated samples. Table 3.2 shows HDCGAN significantly improves state-of-the-art results.

	MS-SSIM
Gulrajani *et al.* [115]	0.2854
Karras *et al.* [70]	0.2838
HDCGAN	**0.1978**

TABLE 3.2: Multi-scale structural similarity (MS-SSIM) results on CelebA at resized image size 128×128. Lower is better.

We monitor MS-SSIM scores across several epochs averaging from 10,000 pairs of generated images to see the temporal performance, Figure 3.17. HDCGAN improves the quality of the samples while increases the diversity of the generated distribution.

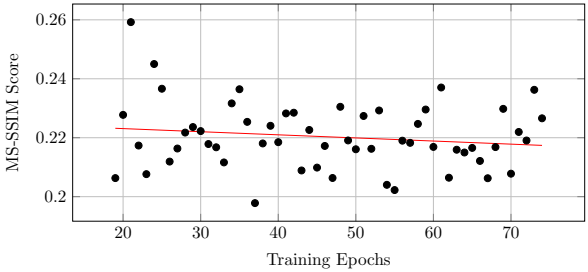

FIGURE 3.17: **MS-SSIM Scores on CelebA across several epochs**. Results are averaged from 10,000 pairs of generated images from epoch 19 to 74. Comparison is made at resized image size 128×128. Affine interpolation is shown in red.

In [118] they propose to evaluate GANs using the FRÉCHET Inception Distance, which assesses the similarity between two distributions by the difference of two Gaussians. We make comparison at resized image size 64×64 on CelebA. Results are computed from 10,000 512×512 generated samples from epochs 36 to 52, resized at image size 64×64 yielding a value of 8.44, Table 3.3, clearly outperforming current reported scores in DCGAN architectures [119].

	Fréchet
Karras *et al.* [70]	16.3
Wu *et al.* [119]	16.0
HDCGAN	**8.44**

TABLE 3.3: FRÉCHET Inception Distance on CelebA at resized image size 64×64. Lower is better.

3.7 DISCUSSION

In this chapter, we propose High-resolution Deep Convolutional Generative Adversarial Networks (HDCGAN) by stacking SELU + BatchNorm (BS) layers. The proposed method generates high-resolution images (e. g. 512×512) in circumstances where the majority of former methods fail. It exhibits a steady and smooth mechanism of training. It also introduces Glasses, the notion that enlarging the input image by a telescope ζ while keeping all convolutional filters unchanged, can arbitrarily improve the final generated results. HDCGAN is the current state-of-the-art in synthetic image generation on CelebA (MS-SSIM 0.1978 and FRÉCHET Inception Distance 8.44).

Further, we present a bias-free dataset of faces containing well-balanced ethnical groups, Curtò & Zarzà, that poses a very difficult challenge and is rich on learning attributes to sample from. Moreover, we enhance Curtò with 4,239 unlabeled synthetic images generated by HDCGAN, being therefore the first GAN augmented dataset of faces.

4

AN EFFICIENT SEGMENTATION TECHNIQUE

4.1 INTRODUCTION

Detection and Segmentation are key components in any toolbox of Computer Vision. In this paper we present a technique of hashing to segment an object given its bounding box and therefore attain simultaneously both Detection and Segmentation. At its heart lies a novel way to retrieve and generate a high-quality segmentation, which is crucial for a wide variety of CV applications. Simply put, we use a state-of-the-art convolutional network to detect the objects, but hashing on top of a high-quality hierarchy of regions to generate the segmentations [120].

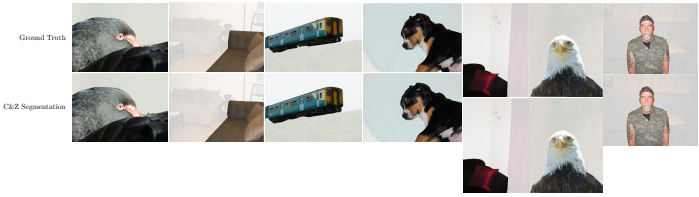

FIGURE 4.1: **Top Detections**. Top: VOC 2012 Ground Truth. Bottom: C&Z Segmentation.

Detection and Segmentation of Objects are two popular problems in Computer Vision and Machine Learning, historically treated as separated tasks. We consider these strongly related vision tasks as a unique one: detecting each object in an image and assigning to each pixel a binary label inside the corresponding bounding box.

C&Z Segmentation addresses the problem with a surprising different technique that deviates from the current norm of using proposal object candidates [121]. In semantic segmentation the need for rich information models that entangle some kind of notion from the different parts that constitute an object is exacerbated. To alleviate this issue we build on the use of the hierarchical model in [122] and explore the rich space of information of the Ultrametric Contour Map (UCM) in order to find the best

possible semantic segmentation of the given object. For this task, we exploit bounding boxes to facilitate the search. Hence, we simply hash the patches enclosed by the bounding boxes and retrieve closest nearest neighbors to the given objects, obtaining superior segmentations. Using this simple but effective technique we get the segmentation mask which is then refined using Hierarchical Section Pruning.

We start from a detector of bounding boxes and refine the object support, as in Hypercolumns [123]. We propose here a train-free similarity hashing alternative to their approach.

We present a simple yet effective module that leverages the need for a training step and can provide segmentations after any given detector. Our approach is to use a state-of-the-art region-based CNN detector [62] as prior step to guide the process of segmentation.

Outline: We begin next with a high-level description of the proposed method and develop further the idea to propose Hierarchical Section Hashing and Hierarchical Section Pruning in Section 5 and Section 4.3. Prior work follows in Section 4.2. We conclude with the evaluation metrics in Section 4.4 and a brief discussion in Section 4.5.

We start with a primer. C&Z Segmentation consists on the following main blocks:

- **Detection of Objects using Bounding Boxes.** We use a convolutional neural network [62] to detect all the objects in an image and generate the corresponding bounding boxes. We consider a detected object in an image as each output candidate thresholded by the class level score (benchmarks specifications in Section 4.4).

- **Hierarchy.** The image is presented as a tree of hierarchical regions based on the UCM [122].

- **Similarity Hashing.** We develop Hierarchical Section Hashing based on the LSH technique of [124].

- **Refinement of Regions.** The segmentations are refined by the use of Hierarchical Section Pruning.

- **Evaluation.** We evaluate the results on the PASCAL VOC 2012 Segmentation dataset [125] using the JACCARD index metric, which measures the average best overlap achieved by a segmentation mask for a ground truth object.

This work is inspired on how humans segment images: they first localize the objects they want to segment, they carefully inspect the object on the image by the use of their visual system and finally they choose the region that belongs to the body of that particular object. We believe that although the problem of detection has to be solved by the use of deep learning based on convolutional neural networks, in the same way that current breakthroughs have been attained in Generative Adversarial Networks (GAN) [60, 65, 67, 70, 71, 95], the problem of segmenting those objects is of a different nature and can be best understood by the use of hashing. Furthermore, current research trends link concepts of Deep Learning to Kernel Methods, proposing a unifying theory of learning in [12, 55].

Our main contributions are presented as follows:

- Novel approach to solve the task of segmentation by similarity hashing exploiting the detection of objects using bounding boxes.

- Use of hierarchical structures which are rich on semantic meaning instead of other current state-of-the-art techniques such as generation of proposal object candidates.

- No need of training data for the task of segmentation under the framework of detection using bounding boxes, that is train-free accurate segmentations.

- State-of-the-art results.

To our knowledge, we are the first to provide a segmentation based on hashing. This approach leverages the need to optimize over a high-dimensional space.

Despite the success of region proposal methods in detection, they have in turn arisen as the main computational bottleneck of these approaches. Yet unlike the latter, hierarchical structures derived from the UCM are in comparison inexpensive to compute and store. While we continue to use a very fast region-based convolutional neural network (R-CNN) to solve the task of detection, we propose to solve the problem of segmentation by exploring efficiently the space generated by a hierarchical image.

4.2 PRIOR WORK

Recent works [69, 126] present Object Detection and Segmentation as a single problem. The task requires to detect and segment every instance of a category in the image. Our work is however more closely related to the approach in Hypercolumns [123], where they go from bounding boxes to segmented masks. Our course of action is related in the sense that we propose an alternative that does not require a training step and can be used as an off-the-shelf high-quality segmenter.

For semantic segmentation [66, 107, 121, 122, 127–129], there has been several approaches where they guide the process of segmentation by the use of a prior detector [62, 64, 130–133]. Recently, this strategy has also presented state-of-the-art results in person detection and pose estimation [134]. Alternate procedures count on a human-on-the-loop [74]. Ongoing research on the matter uses Neural Architecture Search [135–140] to design efficient architectures of neural networks for dense image prediction [141]. With the advent of present-day autonomous vehicles, the need to generate segmentations directly from the point cloud given by LIDAR [142–144], as well as detect 3D objects [37, 145–150], is also recently attracting lots of research efforts. Our segmenter starts rather than from raw pixels, [151] and [152], or bounding box proposals as in [153] and [126], from a set of hierarchical regions given by the UCM structure. Other techniques rely on superpixels e. g. [154]. This is a distinct tactic that works directly on a different representation.

4.3 SEGMENTATION C&Z

We delve into the details of the C&Z Segmentation construction, Figure 4.2.

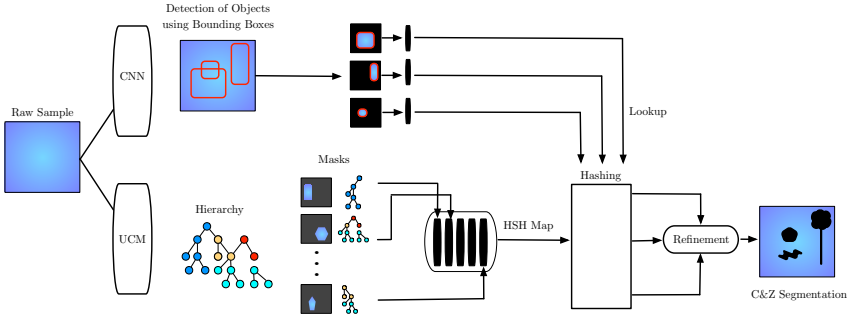

FIGURE 4.2: **C&Z Segmentation**. We construct a hierarchical image based on the UCM and 'train' the HSH map by hashing each region of the parent partition nodes. To retrieve a segmentation mask, we 'test' the HSH map by doing a lookup of the detected area enclosed by the bounding box, videlicet a fast search of approximate nearest neighbors on the hierarchical structure, and finally refine the result through HSP.

4.3.1 *Detection of Objects Using Bounding Boxes*

We begin by using the R-CNN object detector proposed by [62], which is in turn based on [64]. It introduces a Region Proposal Network (RPN) for the task of generating detection proposals and then solves the task of detection by the use of a FAST R-CNN detector. They train a CNN on ImageNet Classification and fine-tune the network on the VOC detection set. For our experiments, we use the network trained on VOC 2007 and 2012, and evaluate the results on the VOC 2012 evaluation set. We use the very deep VGG-16 model [63], Figure 4.3.

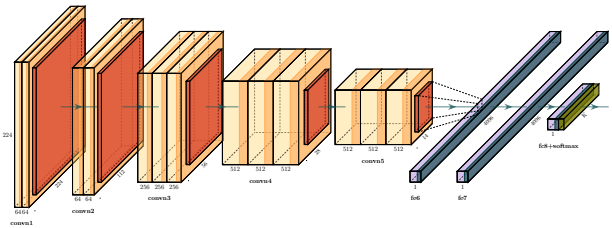

FIGURE 4.3: **VGG-16 Architecture**. VGG-16 model consists on an arrangement of convolutions, layers fully connected and softmax.

4.3.2 *Hierarchical Image*

We consider the representation of a hierarchical image described in [121]. Considering a segmentation of an image into regions that partition its domain $S = \{S_c\}_c$. A segmentation hierarchy is a family of partitions $\{S^*, S^1, \ldots, S^L\}$ such that: (1) S^* is the finest set of superpixels, (2) S^L is the complete domain, and (3) regions from coarse levels are unions of regions from fine levels.

4.3.3 *Hierarchical Section Hashing*

In this paper we introduce a novel segmentation algorithm that exploits bounding boxes to automatically select the best hierarchical region that segments the image. We introduce Hierarchical Section Hashing (HSH) which is in turn based on Locality-Sensitive Hashing (LSH). This algorithm helps us surpass the problem of computational complexity of the k-nearest neighbor rule and allows us to do a fast approximate neighbor search in the hierarchical structure of [122].

HSH can be summarized as follows:

- Detect bounding boxes on an image using a state-of-the-art convolutional neural network [62].
- Construct a hierarchical image by using the UCM and convey the result as a hierarchical region tree.
- Each hierarchical region is indexed by a number of tables of hashing using LSH and then constructing a HSH map.
- Each bounding box is hashed into the HSH map to retrieve the approximate nearest neighbor in sublinear time.

C&Z Segmentation has two main steps: first 'train' the HSH map with all the hierarchical regions of the image. Then 'test' the HSH map with all the detected bounding boxes to retrieve the approximate nearest neighbors that segment each of the objects in the image. The novelty of this approach is that it provides the best hierarchical region provided by the UCM structure

that segments the object image. C&Z exploits the detection of objects using bounding boxes because it relies on the correct detection of the object detector.

4.3.4 *Hierarchical Section Pruning*

The final piece is to refine the segmentations given by the HSH map by using what we call Hierarchical Section Pruning (HSP).

HSP procedure can be summarized as follows:

- Once a segmentation mask has been selected for all the objects in the given image, and their bounding boxes recomputed, the bounding box overlap ratio for all box pair combinations, according to the intersection over union criterium, is performed.

- Those masks that present overlap with other object masks on the same image are hierarchically unselected. We always proceed to unselect the low-level hierarchical regions, which by construction enclose a smaller region area and thus a single segmented object, from the high-level hierarchical region, which encloses more than one object and a bigger image area.

- Finally, isolated pixels on the mask are erased to preserve a single connected segmentation.

HSP is based on the fact that each segmentation mask represents a node on the hierarchical region tree constructed from the UCM. Therefore, hierarchical sections containing more than one object represent higher level nodes on the hierarchy. When HSP is applied, low-level hierarchical regions are unselected from the high-level hierarchical sections and therefore replaced by mid-low level sections on the same region tree structure that represents a single object or a smaller area of the image.

HSH and HSP Visual Examples can be seen in Figure 4.4.

	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	MBike	Person	Plant	Sheep	Sofa	Train	TV	Global
C&Z Segmentation (Instance Level)	45.4	27.5	55.9	44.2	42.0	43.2	41.3	66.3	31.4	57.2	42.3	63.3	43.8	43.6	40.9	40.6	57.2	51.2	48.0	54.1	45.2
C&Z Segmentation (Class Level)	33.3	18.5	48.1	37.5	40.7	45.1	39.4	59.9	23.3	51.0	43.3	60.4	39.8	43.1	34.6	37.2	51.0	47.0	53.6	54.2	43.1

TABLE 4.1: **VOC 2012 Validation Set**. Per-class and global JACCARD Index Metric at instance level.

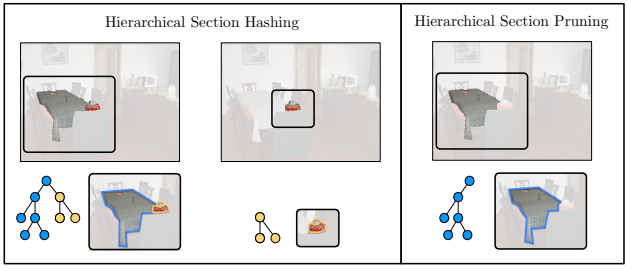

FIGURE 4.4: Left: HSH Visual Example. Right: HSP Visual Example.

C&Z Segmentation relies on the prior detection and therefore availability of bounding boxes for all the objects in a given image. The latter can be very useful as C&Z can be understood as a simple and effective technique to provide high-quality segmentations of still images after any available detector of bounding boxes. Likewise, you get train-free off-the-shelf accurate segmentations for any given method that detects bounding boxes.

4.3.5 *Locality Sensitive Hashing*

Our goal is to retrieve the k-nearest neighbors of a given hierarchical vector, which we call *image code*. In this setup we are limited by the curse of dimensionality and therefore using an exact search is inefficient. Our approach uses a technique of approximate nearest neighbors: Locality Sensitive Hashing (LSH).

A LSH function maps $x \rightarrow h(x)$ such that the similarity between $(\mathbf{x}, \mathbf{y})$ is preserved as

$$\left| \frac{d(h(\mathbf{x}), h(\mathbf{y}))}{D(\mathbf{x}, \mathbf{y})} - 1 \right| \leq \epsilon \tag{4.1}$$

which is not possible for all $D(\mathbf{x}, \mathbf{y})$ but available for instance for euclidean metrics.

We build on the LSH work of [30, 124, 155–158]. LSH is a randomized hashing scheme, investigated with the primary goal of $\epsilon - R$ neighbor search. Its main constitutional block is a family of locality sensitive functions. We can define that a family $\mathcal{H}$ of functions $h : \mathcal{X} \to \{0, 1\}$ is (p_1, p_2, r, R)-sensitive if, for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$,

$$Pr_{h \sim U[\mathcal{H}]}(h(\mathbf{x}) = h(\mathbf{y}) \mid ||\mathbf{x} - \mathbf{y}|| \leq r) \geq p_1, \tag{4.2}$$

$$Pr_{h \sim U[\mathcal{H}]}(h(\mathbf{x}) = h(\mathbf{y}) \mid ||\mathbf{x} - \mathbf{y}|| \geq R) \leq p_2, \tag{4.3}$$

where these probabilities are chosen from a random choice of $h \in \mathcal{H}$.

Algorithm 1 gives a simple description of the LSH algorithm for the given case when the distance of interest is L_1, which is the one in use in C&Z Segmentation. The family $\mathcal{H}$ in this case contains axis-parallel stumps, which means the value of $h \in \mathcal{H}$ is generated by taking a simple dimension $d \in \{1, \ldots, \dim(\mathcal{X})\}$ and thresholding it with some T:

$$h^{LSH} = \begin{cases} 1 & \text{if } x_d \leq T, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

A LSH function $g : \mathcal{X} \to \{0, 1\}^k$ is formed by independently k functions $h_1, \ldots, h_k \in \mathcal{H}$.

That is, we can understand that an example in our hierarchical partition $S_c \in \mathcal{S}$ provides a k-bit key

$$g(S_c) = [h_1(S_c), \ldots, h_k(S_c)]. \tag{4.5}$$

This process is repeated l times and produces l independently constructed functions of hashing $g_1, \ldots, g_l$. The available reference ('training') data S are indexed by each one of the l functions of hashing, producing l tables of hashing, namely each of the S_c hierarchical partitions generated by all the corresponding parents of the hierarchical tree.

Algorithm 1 LSH Algorithm [156]

Given: Dataset $X = [\mathbf{x}_1, \mathbf{x}_N], \mathbf{x}_c \in \mathbb{R}^{\dim(X)}$.
Given: Number of bits k, number of tables l.
Output: A set of

1: **for** $z = 1, \ldots, l$ **do**
2: **for** $c = 1, \ldots, k$ **do**
3: Randomly (uniformly) draw

$$d \in \{1, \ldots, \dim(\mathcal{X})\}.$$

4: Randomly (uniformly) draw

$$\min\{\mathbf{x}_{(d)}\} \leq v \leq \max\{\mathbf{x}_{(d)}\}.$$

5: Let h_c^z be the function $\mathcal{X} \to \{0, 1\}$ defined by

$$h_c^z(\mathbf{x}) = \begin{cases} 1 & \text{if } x_{(d)} \leq v, \\ 0 & \text{otherwise.} \end{cases}$$

6: The z-th LSH function is $g_z = [h_1^z, \ldots, h_k^z]$.

Once the LSH data structure has been built it can be used to perform a very efficient search for approximate neighbors in the following way. When a query S_0 arrives, we compute its key for each table of hashing z, and record the examples $C = \{S_1^l, \ldots, S_{n_l}^l\}$ resulting from the lookup with that key. In other words, we find the 'training' examples that fell in the same bucket of the l-th table of hashing to which S_0 would fall. These l lookup operations produce a set of candidate matches, $C = \cup_{z=1}^{l} C_z$. If this set is empty, the algorithm reports it and stops. Otherwise, the distances between candidate matches and S_0 are explicitly evaluated, and the examples that match the search criteria, which means that are closer to S_0 than $(1 + \epsilon)R$, are returned.

4.4 EVALUATION AND RESULTS

We extensively evaluate C&Z Segmentation on VOC 2012 validation set. Top detections from our algorithm can be seen in Figure 5.1.

4.4.1 *JACCARD Index Metric*

In Table 4.1 we show the results of the JACCARD Index Metric. This measure represents the average best overlap achieved by a candidate for a ground truth object.

C&Z Segmentation with Jaccard at instance level 45.24% and Jaccard at class level 43.05%. Recall at overlap 0.5 is 43.36%.

4.5 DISCUSSION

In this paper we introduce C&Z Segmentation, an algorithm to segment objects based on hashing that exploits the detection using bounding boxes. We show C&Z achieves compelling results and generates off-the-shelf accurate segmentations.

5

A NOTE ON CLOSED TIMELIKE CURVES

INTRODUCTION

Our aim is to present a novel interpretation [159] of Closed Timelike Curves as solutions of the EINSTEIN field equation.

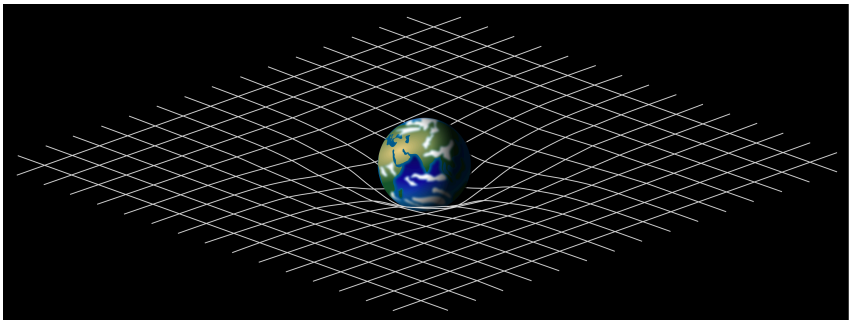

FIGURE 5.1: Space-time curvature.

TIME-TRAVELING

As stated in reference [160], the existence of closed timelike lines makes theoretically possible in these worlds to travel into the past, or otherwise influence the past. Our contribution lies in the interpretation of these solutions.

We claim that these CTC solutions are indeed the ones that govern space time for humans and animated objects in general. What this means is that our actions in the future, can actually affect our past actions. In some sense, all our human existence is based on these interactions between present and past. Not only these solutions are not pathological, but instead describe our day-to-day.

BIBLIOGRAPHY

1. Rahimi, A. & Recht, B. Random Features for Large-scale Kernel Machines. *NIPS* **20** (2007).
2. Le, Q., Sarlós, T. & Smola, A. Approximating Kernel Expansions in Loglinear Time. *ICML* (2013).
3. Xiao, H., Rasul, K. & Vollgraf, R. FASHION-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747* (2017).
4. Rudi, A. & Rosasco, L. Generalization Properties of Learning with Random Features. *NIPS* **30** (2017).
5. Yang, Z., Moczulski, M., Denil, M., Freitas, N., Smola, A., Song, L. & Wang, Z. Deep Fried Convnets. *IEEE International Conference on Computer Vision* (2015).
6. Moczulski, M., Denil, M., Appleyard, J. & Freitas, N. ACDC: a Structured Efficient Linear Layer. *ICLR* (2016).
7. Hong, W., Yuan, J. & Bhattacharjee, S. Fried Binary Embedding for High-dimensional Visual Features. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).
8. Kawaguchi, K., Xie, B., Verma, V. & Song, L. Deep Semi-random Features for Nonlinear Function Approximation. *AAAI Conference on Artificial Intelligence* (2018).
9. Cho, Y. & Saul, L. Kernel Methods for Deep Learning. *NIPS* **22** (2009).
10. Wilson, A. G., Hu, Z., Salakhutdinov, R. & Xing, E. P. Stochastic Variational Deep Kernel Learning. *NIPS* (2016).
11. Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z. & Xing, E. P. Learning Scalable Deep Kernels with Recurrent Structure. *JMLR* **18** (2017).
12. Curtó, J. D., Zarza, I. C., Yang, F., Smola, A., Torre, F., Ngo, C. & Gool, L. McKernel: a Library for Approximate Kernel Expansions in Log-linear Time. *arXiv:1702.08159* (2017).
13. Wigner, E. The Unreasonable Effectiveness of Mathematics in the Natural Sciences. *Communications in Pure and Applied Mathematics* **13** (1960).

14. Vapnik, V. & Izmailov, R. Rethinking Statistical Learning Theory: Learning Using Statistical Invariants. *Machine Learning* (2018).

15. Johnson, J. & Püschel, M. In Search of the Optimal WALSH-HADAMARD Transform. *IEEE International Conference on Acoustics, Speech, and Signal Processing* (2000).

16. Sharmanska, V., Quadrianto, N. & Lampert, C. H. Learning to Rank Using Privileged Information. *IEEE International Conference on Computer Vision* (2013).

17. Chwialkowski, K., Ramdas, A., Sejdinovic, D. & Gretton, A. Fast Two-sample Testing with Analytic Representations of Probability Measures. *NIPS* **2** (2015).

18. Reddi, S., Ramdas, A., Póczos, B., Singh, A. & Wasserman, L. On the High-dimensional Power of a Linear-time Two Sample Test under Mean-shift Alternatives. *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2015).

19. Wang, Z. & Ji, Q. Classifier Learning with Hidden Information. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

20. Li, W., Dai, D., Tan, M., Xu, D. & Gool, L. Fast Algorithms for Linear and KERNEL SVM+. *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

21. Wang, M., Liu, Y. & Huang, Z. Large Margin Object Tracking with Circulant Feature Maps. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

22. Cortes, C. & Vapnik, V. Support-vector Networks. *Machine Language* **20** (1995).

23. Cucker, F. & Smale, S. On the Mathematical Foundations of Learning. *Bulletin of the AMERICAN Mathematical Society* **39** (2001).

24. Poggio, T. & Smale, S. The Mathematics of Learning: Dealing with Data. *Notices of the AMERICAN Mathematical Society* (2003).

25. Vapnik, V. & Vashist, A. A New Learning Paradigm: Learning Using Privileged Information. *Neural Networks* **22** (2009).

26. Vapnik, V. & Izmailov, R. Learning Using Privileged Information: Similarity Control and Knowledge Transfer. *JMLR* **16** (2015).

27. Box, G. E. P. & Muller, M. E. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics* **29** (1958).

28. Yang, Z., Smola, A., Song, L. & Wilson, A. G. Learning Fast Kernels. *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2014).

29. Lu, Y., Dhillon, P. S., Foster, D. & Ungar, L. Faster Ridge Regression via the Subsampled Randomized HADAMARD Transform. *NIPS* **1** (2013).

30. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I. & Schmidt, L. Practical and Optimal LSH for Angular Distance. *NIPS* **1** (2015).

31. Ben-Artzi, G., Hel-Or, H. & Hel-Or, Y. The Gray-code Filter Kernels. *IEEE Transactions (TPAMI)* **29** (2007).

32. Ouyang, W. & Cham, W. K. Fast Algorithm for WALSH HADAMARD Transform on Sliding Windows. *IEEE Transactions (TPAMI)* **32** (2010).

33. Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B. & Liao, Q. Why and When Can Deep – but Not Shallow – Networks Avoid the Curse of Dimensionality: a Review. *International Journal of Automation and Computing* **14** (2017).

34. Mhaskar, H., Liao, Q. & Poggio, T. When and Why Are Deep Networks Better than Shallow Ones? *AAAI* (2017).

35. Girosi, F., Jones, M. & Poggio, T. Regularization theory and neural networks architectures. *Neural Computation* (1995).

36. Girosi, F. An equivalence between sparse approximation and Support Vector Machines. *Neural Computation* (1998).

37. Liang, M., Yang, B., Wang, S. & Urtasun, R. Deep Continuous Fusion for Multi-sensor 3D Object Detection. *EUROPEAN Conference on Computer Vision* (2018).

38. Kawaguchi, K. & Kaelbling, L. P. Elimination of All Bad Local Minima in Deep Learning. *arXiv:1901.00279* (2019).

39. Sohl-Dickstein, J. & Kawaguchi, K. Eliminating All Bad Local Minima from Loss Landscapes without Even Adding an Extra Unit. *arXiv:1901.03909* (2019).

40. Poggio, T. & Liao, Q. Theory II: Landscape of the Empirical Risk in Deep Learning. *arXiv:1703.09833* (2017).

41. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML* **37** (2015).

42. Lakshminarayanan, B., Pritzel, A. & Blundell, C. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. *NIPS* (2017).

43. Zhang, H., Shao, J. & Salakhutdinov, R. Deep Neural Networks with Multi-branch Architectures Are Less Non-convex. *arXiv:1806.01845* (2018).

44. Tran, T., Pham, T., Carneiro, G., Palmer, L. & Reid, I. A Bayesian Data Augmentation Approach for Learning Deep Models. *NIPS* (2017).

45. Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V. & Le, Q. V. AutoAugment: Learning Augmentation Strategies from Data. *IEEE Conference on Computer Vision and Pattern Recognition* (2019).

46. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proceedings of the Institute of Electrical and Electronics Engineers* **86** (1998).

47. Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR* (2015).

48. Luong, M., Pham, H. & Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2015).

49. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. Attention Is All You Need. *NIPS* (2017).

50. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *JMLR* **15** (2014).

51. Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. *ICML* **30** (2013).

52. He, K., Zhang, X., Ren, S. & Sun, J. Delving Deep into Rectifiers: Surpassing Human-level Performance on ImageNet Classification. *IEEE International Conference on Computer Vision* (2015).

53. Clevert, D., Unterthiner, T. & Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ICLR* (2016).

54. Klambauer, G., Unterthiner, T., Mayr, A. & Hochreiter, S. Self-normalizing Neural Networks. *NIPS* (2017).

55. Curtó, J. D., Zarza, I. C., Kitani, K., King, I. & Lyu, M. R. Doctor of Crosswise: Reducing Over-parametrization in Neural Networks. *arXiv:1905.10324* (2019).

56. Rahimi, A. & Recht, B. Weighted Sums of Random Kitchen Sinks: Replacing Minimization with Randomization in Learning. *NIPS* **21** (2008).

57. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *NIPS* **2** (2013).

58. Pennington, J., Socher, R. & Manning, C. Global Vectors for Word Representation. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014).

59. Devlin, J., Chang, M., Lee, K. & Toutanova, K. Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805* (2018).

60. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. Generative Adversarial Networks. *NIPS* **27** (2014).

61. Long, J., Shelhamer, E. & Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

62. Ren, S., He, K., Girshick, R. & Sun, J. FASTER R-CNN: Towards Real-time Object Detection with Region Proposal Networks. *NIPS* **28** (2015).

63. Simonyan, K. & Zisserman, A. Very Deep Convolutional Networks For Large-scale Image Recognition. *ICLR* (2015).

64. Girshick, R. FAST R-CNN. *IEEE International Conference on Computer Vision* (2015).

65. Radford, A., Metz, L. & Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ICLR* (2016).

66. Yu, F. & Koltun, V. Multi-scale Context Aggregation by Dilated Convolutions. *ICLR* (2016).

67. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. & Chen, X. Improved Techniques for Training GANs. *NIPS* **29** (2016).

68. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

69. He, K., Gkioxari, G., Dollár, P. & Girshick, R. MASK R-CNN. *IEEE International Conference on Computer Vision* (2017).

70. Karras, T., Aila, T., Laine, S. & Lehtinen, J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *ICLR* (2018).

71. Curtó, J. D., Zarza, I. C., Torre, F., King, I. & Lyu, M. R. High-resolution Deep Convolutional Generative Adversarial Networks. *arXiv:1711.06491* (2017).

72. Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M. & Yang, S. Adaptive Structural Learning of Artificial Neural Networks. *ICML* (2017).

73. Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S. & Birchfield, S. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

74. Acuna, D., Ling, H., Kar, A. & Fidler, S. Efficient Interactive Annotation of Segmentation Datasets with POLYGON-RNN++. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

75. Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Ré, C. & Zaharia, M. DAWNBench: an End-to-end Deep Learning Benchmark and Competition. *NIPS* (2017).

76. Duan, Y., Chen, X., Houthooft, R., Schulman, J. & Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. *ICML* **48** (2016).

77. Levine, S., Finn, C., Darrell, T. & Abbeel, P. End-to-end Training of Deep Visuomotor Policies. *JMLR* (2016).

78. Bengio, Y. & Frasconi, P. Credit Assignment through Time: Alternatives to Backpropagation. *NIPS* (1993).

79. Kolda, T. G. & Bader, B. W. Tensor Decompositions and Applications. *SIAM* (2009).

80. Wu, J., Zhang, C., Xue, T., Freeman, W. T. & Tenenbaum, J. B. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-adversarial Modeling. *NIPS* **29** (2016).

81. Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O. & Li, H. High-resolution Image Inpainting Using Multi-scale Neural Patch Synthesis. *IEEE International Conference on Computer Vision* (2017).

82. Zhu, J., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation Using Cycle-consistent Adversarial Networks. *IEEE International Conference on Computer Vision* (2017).

83. Bousmalis, K., Silberman, N., Dohan, D., Erhan, D. & Krishnan, D. Unsupervised Pixel-level Domain Adaptation with Generative Adversarial Networks. *IEEE International Conference on Computer Vision* (2017).

84. Li, C., Xu, K., Zhu, J. & Zhang, B. Triple Generative Adversarial Nets. *NIPS* **30** (2017).

85. Wang, T., Liu, M., Zhu, J., Tao, A., Kautz, J. & Catanzaro, B. High-resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *IEEE International Conference on Computer Vision* (2018).

86. Wang, T., Liu, M., Zhu, J., Liu, G., Tao, A., Kautz, J. & Catanzaro, B. Video-to-video Synthesis. *NIPS* (2018).

87. Portenier, T., Hu, Q., Szabó, A., Bigdeli, S. A., Favaro, P. & Zwicker, M. FaceShop: Deep Sketch-based Face Image Editing. *ACM Transactions on Graphics (SIGGRAPH)* **37** (2018).

88. Lombardi, S., Saragih, J., Simon, T. & Sheikh, Y. Deep Appearance Models for Face Rendering. *ACM Transactions on Graphics (SIGGRAPH)* **37** (2018).

89. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X. & Huang, T. S. Generative Image Inpainting with Contextual Attention. *IEEE International Conference on Computer Vision* (2018).

90. Sankaranarayanan, S., Balaji, Y., Jain, A., Lim, S. N. & Chellappa, R. Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation. *IEEE International Conference on Computer Vision* (2018).

91. Romero, A., Arbeláez, P., Gool, L. & Timofte, R. SMIT: Stochastic Multi-label Image-to-image Translation. *arXiv:1812.03704* (2018).

92. Chen, Y., Li, W., Chen, X. & Gool, L. Learning Semantic Segmentation from Synthetic Data: a Geometrically Guided Input-output Adaptation Approach. *IEEE International Conference on Computer Vision* (2019).

93. de Curtó, J. & Duvall, R. Cycle-consistent Generative Adversarial Networks for Neural Style Transfer Using Data from Chang'E-4. *arXiv:2011.11627* (2020).

94. de Curtó, J. & Duvall, R. Vulcan Centaur: towards End-to-end Real-time Perception in Lunar Rovers. *arXiv:2011.15104* (2020).

95. de Curtò, J. *Vision and Learning in the Context of Exploratory Rovers* PhD thesis (ETH Zürich, 2021).

96. Mescheder, L., Nowozin, S. & Geiger, A. The Numerics of GANs. *NIPS* (2017).

97. Mescheder, L., Nowozin, S. & Geiger, A. Which Training Methods for GANs Do Actually Converge? *ICML* (2018).

98. Arjovsky, M. & Bottou, L. Towards Principled Methods for Training Generative Adversarial Networks. *ICLR* (2017).

99. Sønderby, C., Caballero, J., Theis, L., Shi, W. & Huszár, F. Amortised MAP Inference for Image Super-resolution. *ICLR* (2017).

100. Kingma, D. P. & Welling, M. Auto-encoding Variational Bayes. *ICLR* (2014).

101. van den Oord, A., Kalchbrenner, N. & Kavukcuoglu, K. Pixel Recurrent Neural Networks. *ICML* (2016).

102. Odena, A., Olah, C. & Shlens, J. Conditional Image Synthesis with Auxiliary Classifier GANs. *ICML* (2017).

103. Antoniou, A., Storkey, A. & Edwards, H. Data Augmentation Generative Adversarial Networks. *ICLR* (2018).

104. Wang, X. & Gupta, A. Generative Image Modeling Using Style and Structure Adversarial Networks. *EUROPEAN Conference on Computer Vision* (2016).

105. Zhu, J., Krähenbühl, P., Shechtman, E. & Efros, A. A. Generative Visual Manipulation on the Natural Image Manifold. *EUROPEAN Conference on Computer Vision* (2016).

106. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. & Metaxas, D. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. *IEEE International Conference on Computer Vision* (2017).

107. Chen, Q. & Koltun, V. Photographic Image Synthesis with Cascaded Refinement Networks. *IEEE International Conference on Computer Vision* (2017).

108. Dosovitskiy, A. & Brox, T. Generating Images with Perceptual Similarity Metrics Based on Deep Networks. *NIPS* (2016).

109. Zhao, J., Mathieu, M. & LeCun, Y. Energy-based Generative Adversarial Networks. *ICLR* (2017).

110. Wei, X., Gong, B., Liu, Z., Lu, W. & Wang, L. Improving the Improved Training of WASSERSTEIN GANs: a Consistency Term and its Dual Effect. *ICLR* (2018).

111. Brock, A., Donahue, J. & Simonyan, K. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *ICLR* (2019).

112. Xiong, X. & Torre, F. Supervised Descent Method and its Application to Face Alignment. *CVPR* (2013).

113. Liu, Z., Luo, P., Wang, X. & Tang, X. Deep Learning Face Attributes in the Wild. *ICCV* (2015).

114. Arjovsky, M., Chintala, S. & Bottou, L. WASSERSTEIN Generative Adversarial Networks. *ICML* (2017).

115. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. Improved Training of WASSERSTEIN GANs. *NIPS* (2017).

116. Denton, E., Chintala, S., Szlam, A. & Fergus, R. Deep Generative Image Models Using a LAPLACIAN Pyramid of Adversarial Networks. *NIPS* **28** (2015).

117. Kingma, D. & Ba, J. A Method for Stochastic Optimization. *ICLR* (2015).

118. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S. GANs Trained by a Two Time-scale Update Rule Converge to a Local NASH Equilibrium. *NIPS* **30** (2017).

119. Wu, J., Huang, Z., Thoma, J., Acharya, D. & Gool, L. WASSERSTEIN Divergence for GANs. *EUROPEAN Conference on Computer Vision* (2018).

120. Curtó, J. D., Zarza, I. C., Smola, A. & Gool, L. Segmentation of Objects by Hashing. *arXiv:1702.08160* (2017).

121. Arbeláez, P., Pont-Tuset, J., Barron, J. T., Marques, F. & Malik, J. Multiscale Combinatorial Grouping. *IEEE Conference on Computer Vision and Pattern Recognition* (2014).

122. Arbeláez, P., Maire, M., Fowlkes, C. & Malik, J. Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions (TPAMI)* **33** (2011).

123. Hariharan, B., Arbeláez, P., Girshick, R. & Malik, J. Hypercolumns for Object Segmentation and Fine-grained Localization. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

124. Charikar, M. Similarity Estimation Techniques from Rounding Algorithms. *STOC* (2002).

125. Everingham, M., Gool, L., Williams, C., Winn, J. & Zisserman, A. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV* **88** (2010).

126. Hariharan, B., Arbeláez, P., Girshick, R. & Malik, J. Simultaneous Detection and Segmentation. *EUROPEAN Conference on Computer Vision* (2014).

127. Chen, L., Papandreou, G., Schroff, F. & Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv:1706.05587* (2017).

128. Chen, L., Zhu, Y., Papandreou, G., Schroff, F. & Adam, H. Encoder-decoder with Atrous Separable Convolution for Semantic Image Segmentation. *EUROPEAN Conference on Computer Vision* (2018).

129. Chen, L., Hermans, A., Papandreou, G., Schroff, F., Wang, P. & Adam, H. MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

130. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You Only Look Once: Unified, Real-time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

131. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. & Berg, A. C. SSD: Single Shot Multibox Detector. *EUROPEAN Conference on Computer Vision* (2016).

132. Redmon, J. & Farhadi, A. YOLO9000: Better, Faster, Stronger. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

133. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. & Murphy, K. Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

134. Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C. & Murphy, K. Towards Accurate Multi-person Pose Estimation in the Wild. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

135. Zoph, B. & Le, Q. Neural Architecture Search with Reinforcement Learning. *ICLR* (2017).

136. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning Transferable Architectures for Scalable Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

137. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A., Huang, J. & Murphy, K. Progressive Neural Architecture Search. *EUROPEAN Conference on Computer Vision* (2018).

138. Pham, H., Guan, M., Zoph, B., Le, Q. & Dean, J. Efficient Neural Architecture Search via Parameters Sharing. *ICML* (2018).

139. Zhang, Z. B., Zuo, W., Zeng, X. G., Gao, X. Y. & Ren, X. The Scientific Data and its Archiving from Chang'E 4 Mission. *4th Planetary Data Workshop* (2019).

140. Li, L. & Talwalkar, A. Random Search and Reproducibility for Neural Architecture Search. *ICML* (2019).

141. Chen, L., Collins, M. D., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H. & Shlens, J. Searching for Efficient Multi-scale Architectures for Dense Image Prediction. *NIPS* (2018).

142. Qi, C. R., Su, H., Mo, K. & Guibas, L. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

143. Qi, C. R., Yi, L., Su, H. & Guibas, L. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NIPS* (2017).

144. Qi, C. R., Liu, W., Wu, C., Su, H. & Guibas, L. Frustum PointNets for 3D Object Detection from RGB-D Data. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

145. Chen, X., Ma, H., Wan, J., Li, B. & Xia, T. Multi-view 3D Object Detection Network for Autonomous Driving. *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

146. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A. & Torralba, A. Semantic Understanding of Scenes through the ADE20K Dataset. *IJCV* (2018).

147. Yang, B., Luo, W. & Urtasun, R. PIXOR: Real-time 3D Object Detection from Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

148. Luo, W., Yang, B. & Urtasun, R. Fast and Furious: Real Time End-to-end 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. *IEEE Conference on Computer Vision and Pattern Recognition* (2018).

149. Ku, J., Mozifian, M., Lee, J., Harakeh, A. & Waslander, S. Joint 3D Proposal Generation and Object Detection from View Aggregation. *IEEE International Conference on Intelligent Robots and Systems (IROS)* (2018).

150. Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J. & Beijbom, O. PointPillars: Fast Encoders for Object Detection from Point Clouds. *IEEE Conference on Computer Vision and Pattern Recognition* (2019).

151. Long, J., Shelhamer, E. & Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

152. Badrinarayanan, V., Kendall, A. & Cipolla, R. SegNet: a Deep Convolutional Encoder-decoder Architecture for Image Segmentation. *IEEE Transactions (TPAMI)* **39** (2017).

153. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition* (2014).

154. Mostajabi, M., Yadollahpour, P. & Shakhnarovich, G. Feedforward Semantic Segmentation with Zoom-out Features. *IEEE Conference on Computer Vision and Pattern Recognition* (2015).

155. Indyk, P. & Motwani, R. Approximate Nearest Neighbors: towards Removing the Curse of Dimensionality. *STOC* (1998).

156. Gionis, A., Indyk, P. & Motwani, R. Similarity Search in High Dimensions via Hashing. *VLDB* (1999).

157. Shakhnarovich, G., Viola, P. & Darrell, T. Fast Pose Estimation with Parameter-sensitive Hashing. *IEEE International Conference on Computer Vision* (2003).

158. Shakhnarovich, G. *Learning Task-specific Similarity* PhD thesis (MIT, 2005).

159. de Curtò, J. & de Zarzà, I. A Note on Closed Timelike Curves (2021).

160. Gödel, K. An Example of a New Type of Cosmological Solutions of Einstein's Field Equations of Gravitation. *Reviews of Modern Physics. 21, 447.* (1949).

6

PUBLICATIONS

Articles:

1. de Curtò, J. & de Zarzà, I. A Note on Closed Timelike Curves (2021).
2. Curtò, J. D., Zarzà, I. C., Kitani, K., King, I. & Lyu, M. R. Doctor of Crosswise: Reducing Over-parametrization in Neural Networks. *arXiv:1905.10324* (2019).
3. Curtò, J. D., Zarzà, I. C., Torre, F., King, I. & Lyu, M. R. High-resolution Deep Convolutional Generative Adversarial Networks. *arXiv:1711.06491* (2017).
4. Curtò, J. D., Zarzà, I. C., Smola, A. & Gool, L. Segmentation of Objects by Hashing. *arXiv:1702.08160* (2017).
5. Curtò, J. D., Zarzà, I. C., Yang, F., Smola, A., Torre, F., Ngo, C. & Gool, L. McKernel: a Library for Approximate Kernel Expansions in Log-linear Time. *arXiv:1702.08159* (2017).

7

CURRICULUM VITAE

GENERAL INFORMATION

I. de Zarzà i Cubero.

E-mail: z@dezarza.tw
Webpage: www.dezarza.tw
Phone: +1 (412) 407-6791.

Nationality: Catalunya (Regne d'Espanya).

CAREER (SELECTED)

Iris Lunar Rover.
ETH Zürich. **Carnegie Mellon**. Pittsburgh.
BRAIN Team. Sub-lead.
December 2020 - till now.

A small research unit that works at the forefront of research in space exploration in the context of planetary rovers.

Universitat Rovira i Virgili. Tarragona.
Research Scholar. September 2018 - September 2020.
Group of Intelligent Robotics and Computer Vision.

The Chinese University of Hong Kong (CUHK). Hong Kong.
Research Assistant. October 2017 - February 2018.
Department of Computer Science and Engineering.

Carnegie Mellon. Pittsburgh.
Research Assistant. June 2017 - July 2017.
School of Computer Science. Robotics.

City University of Hong Kong. Hong Kong.
Senior Research Assistant. January 2017 - May 2017.
Department of Electrical Engineering.

Challenger Deep, Ltd. Hong Kong.
Technical Co-Lead. Co-founder. May 2016 - November 2017.
Top state-funded incubator program. Cyberport.

ETH Zürich (ETHZ). Zürich.
Graduate Research Assistant. April 2015 - September 2016.
Laboratory of Computer Vision.

City University of Hong Kong. Hong Kong.
Research Associate. July 2015 - August 2015.
Department of Computer Science.

Carnegie Mellon. Pittsburgh.
Research Associate I. June 2014 - August 2014.
School of Computer Science. Robotics.

EDUCATION (SELECTED)

ETH Zürich (ETHZ). Zürich.

Doctor of Science. April 2015 -
Laboratory of Computer Vision.

Carnegie Mellon. Pittsburgh.

Master of Science. May 2014 - February 2015.
School of Computer Science. ML Department and Robotics.

Thesis: A Library for Fast Kernel Expansions with Applications to Computer Vision and Deep Learning.

City University of Hong Kong. Hong Kong.

Master of Science. September 2013 - February 2015.
Department of Electrical Engineering.
GPA: 3.86 (0-4 scale).
Classification of Award: Distinction (4/+100).

Academic Distinctions:

MS Internship Sponsorship 2014.
Award for top performing students. Robotics. Carnegie Mellon. Pittsburgh.

Universitat Autònoma de Barcelona. Cerdanyola del Vallès (Barcelona).

Degree in Mathematics. 2011 - 2013.
Specialization in Pure Mathematics.
Department of Mathematics. Faculty of Sciences.

Thesis: Physical-layer Network Coding: Design of Constellations over Rings.
Grade: Excellent. First Class with Distinction.

Universitat de Barcelona. Barcelona.

Mathematics, Licentiate. First Cycle. 2007 - 2011.
Department of Mathematics and Computer Science.

University Entrance Examination.
Average Grade: 8.93/10. First Class with Distinction.

Academic Distinctions:

- First year scholarship for university studies. Ministry of Education. This award is given to the top nationwide first year university students.
- First year scholarship for university studies. Caixa Manresa. This award is given to the top university entrance examination average grades in the region of Catalunya.

Technological Baccalaureate. 2005 - 2007.
Average Grade: 10/10. First Class Degree and Honorary Scholarship.

Academic Distinctions:

- Outstanding Thesis of Research: Squaring the Circle. Study of the different mathematical approaches to solve the ancient problem of the quadrature of the circle.

- Outstanding Curriculum.

PUBLICATIONS

De Curtò and **De Zarzà**.
A Note on Closed Timelike Curves.

Curtò, **Zarzà**, Kitani, King and Lyu.
Doctor of Crosswise: Reducing Over-parametrization in Neural Networks.
https://www.dezarza.tw/c/doctor_of_crosswise.pdf

Curtò, **Zarzà**, Torre, King and Lyu.
High-resolution Deep Convolutional Generative Adversarial Networks.
https://www.dezarza.tw/c/hdcgan.pdf

Curtò, **Zarzà**, Smola and Gool.
Segmentation of Objects by Hashing.
https://www.dezarza.tw/c/c_and_z.pdf

Curtò, **Zarzà**, Yang, Smola, Torre, Ngo and Gool.
McKernel: A Library for Approximate Kernel Expansions in Log-linear Time.
https://www.dezarza.tw/c/mckernel.pdf

De Curtò i DíAz, **De Zarzà i Cubero** and Vázquez.
Secure Network Coding: Overview and State-of-the-art.
Universitat Autònoma de Barcelona. Cerdanyola del Vallès (Barcelona). 2012.
https://hal.archives-ouvertes.fr/hal-03168605/document

DISSERTATIONS

Master of Science.
A Library for Fast Kernel Expansions with Applications to Computer Vision and Deep Learning.
Supervisors: Smola, De la Torre and Ngo.
Carnegie Mellon. Pittsburgh. 2014.
https://www.dezarza.tw/z/dezarza.pdf
https://www.dezarza.tw/z/slides_dezarza.pdf

Degree in Mathematics.
Physical-layer Network Coding: Design of Constellations over Rings.
Supervisors: Vázquez and Mondelo.
Universitat Autònoma de Barcelona. Cerdanyola del Vallès (Barcelona).
2013.
https://www.dezarza.tw/z/pfc_dezarza.pdf
https://www.dezarza.tw/z/slides_pfc_dezarza.pdf

WORK EXPERIENCE

Institut d'Educació Secundària Joan Coromines. Barcelona.
Teaching Assistant. 2005 - 2006.

LANGUAGES

English -

TOEFL Internet Based test. 11-12-2016. **Score 102/120**.

SERVICES

ICML 2020. Virtual Conference. Formerly in Vienna. 12/07 - 18/07. 2020.

Attendee.

Robotics: Science and Systems. Virtual Conference. Formerly in Corvallis. 12/07 - 16/07. 2020.

Attendee.

SIAM Conference on Imaging Science. Virtual Conference. Formerly in Toronto. 06/07 - 17/07. 2020.

Attendee.

IEEE International Symposium on Information Theory. Virtual Symposium. Formerly in Los Angeles. 21/06 - 26/06. 2020.

Attendee.

IEEE Conference on Computer Vision and Pattern Recognition. Virtual Conference. Formerly in Seattle. 14/06 - 19/06. 2020.

Attendee.

IEEE International Conference on Communications. Virtual Conference. Formerly in Dublin. 07/06 - 11/06. 2020.

Attendee.

IEEE International Conference on Robotics and Automation. Virtual Conference. Formerly in Paris. 31/05 - 04/06. 2020.

Attendee.

SIAM Conference on Mathematics of Data Science. Virtual Conference. Formerly in Cincinnati. 04/05 - 30/06. 2020.

Attendee.

IEEE International Conference on Acoustics, Speech, and Signal Processing. Virtual Conference. Formerly in Barcelona. 04/05 - 08/05. 2020.

Attendee.

ICLR 2020. Virtual Conference. Formerly in Addis Ababa. 26/04 - 01/05. 2020.

Attendee.

Written presentation at the Social Virtual Event on Open source tools and practices in state-of-the-art DL research.

Slides with Q&A annotations: www.decurto.tw/c/iclr2020_DeCurto.pdf

First European Training School in Network Coding: Random Network Coding and Designs over $GF(q)$. IEEE Information Theory Society. Universitat Autònoma de Barcelona. Cerdanyola del Vallès (Barcelona). 04/02 - 08/02. 2013.

From designs over $GF(q)$ to applications of networking: a crossroad for mathematics, computer science and engineering.

Attendee and Volunteer.

EXTRACURRICULAR ACTIVITIES

Program of Open Mentoring. Department of Computer Science. The University of Hong Kong. 2014 - 2018.

Symposium of the Royal Society of Mathematics. The Millennium Problems. Awarded with an assistance grant by Institut de Matemàtica de la Universitat de Barcelona. Barcelona. 1 - 3 June 2011.

Course in Investment and Financial Markets. Technical Analysis and Risk Management. Barcelona. 23 - 26 May 2011.

Course in Investment and Financial Markets. Barcelona. 18 - 21 April 2011.

Competition of Entrepreneurship. EMPRÈN UPC. 1st Edition. Universitat Politècnica de Catalunya (UPC). Finalist project awarded with honorable mention and 1000 euros. Barcelona. 14 March 2011 - 14 June 2011.

PROGRAMMING

C, C++, Java, Python, MATLAB and Prolog.

SOFTWARE

LaTeX, R, Maple, Mathematica and EViews.