UNIVERSITY
OF UDINE

Department of
Mathematics, Computer Science and Physics

PH.D. THESIS
CORSO DI DOTTORATO DI RICERCA IN INFORMATICA E
SCIENZE MATEMATICHE E FISICHE

Efficient Deep Learning: analysis and applications.

CANDIDATE
Giuseppe Lancioni

SUPERVISOR
Prof. Giuseppe Serra

Cycle XXXIV — A.Y. 2020-2021

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
`https://www.dmif.uniud.it/`

Author's Contacts
Via Francesco Denza, 9
34143 Trieste — Italia
+39 347 9448 477
`lancioni.giuseppe@spes.uniud.it`

Alla cara memoria di
Adriano Leonardi e Daniele Nardi.

Acknowledgements

I would like to acknowledge my supervisor, prof. Giuseppe Serra, for his strong support in all the phases of my PhD career. I am also grateful to the students at AILAB, Udine University. They all gave me useful suggestions and intuitions, and greatly helped me in their implementation. Among them, I would like to particularly aknowledge Beatrice Portelli, Alex Falcon, Giovanni D'Agostino, and Lorenzo Cian.
Finally, there is one person without whom all this work would not have been realized: thank you Miriam.

Abstract

During the history of artificial Neural Networks as a methodology in the research of Artificial Intelligence (AI), a continuous path of achievements has been interrupted at least two times due to the frustration of the expectations, leading to the "AI winters". During these periods, funds and research efforts were directed elsewhere, towards topics and fields felt as more stable and promising.

In the last two decades, the rising of Deep Learning (DL), an approach based on neural networks, paved the way to a giant leap forward: historic results have been achieved in many fields such as Computer Vision, Natural Language Processing, Bioinformatics, Home Automation, and so. However, computational cost, amount of training data, model size, training time, inference latency, all together referred to as *footprint metrics*, have all enormously increased. This aspect will probably bring in the near future to a shortage of resources in term of global computing capacity, availability of properly labeled data, and even to a high increase of carbon footprint due to the proliferation of high consuming GPUs, an aspect which raises environmental and ethic concerns. This scenario could prefigure a new AI winter.

Therefore, there is an increasing need of efficient approaches based on Deep Learning, in which the reduction of the footprint metrics is pursued without significant loss in the performance. These approaches are referred to as *Efficient Deep Learning.*

This Ph.D. thesis is divided in two main parts. In the first part, the problem of the computational burden is analyzed by means of a theoretical model of the relation between performance and required computation. It shows a large increase of the footprint to obtain sensible performance improvements. Also is reported how this expectations have been overrun by current systems. Then an analysis is carried out about methodologies and techniques to reduce the footprint metrics, with the aim to build efficient deep learning systems.

In the second part, three research projects are illustrated. To each of them some methodologies and techniques have been applied, which explicitly reduce the footprint metrics and, as a consequence, the required computation. The projects cover three of the main research fields: video person Re-identification (Computer Vision); automatic keyphrase generation (Natural Language Processing); and atomistic Graph Neural Networks for metals (Deep Learning for scientific applications).

Contents

1 Introduction

1.1 Historical outline

Deep Learning (DL) is probably the most promising methodology in the search for Artificial Intelligence [54]. It is a branch of the more general Machine Learning, and it is based on the artificial Neural Networks (NN). So speaking about the history of DL [88, 46] means to trace the development of NN, which started in the first part of the 20th century.

1.1.1 The early days (1943-1969)

The first example of Artificial Neuron was introduced in 1943 by Warren McCulloch and Walter Pitts [113]. They proposed a model directly inspired to the biological neurons, in which a cell receives as inputs binary signals by other connected, identical, cells, then sums these signals and emits 1 if the sum is above some specified threshold, 0 otherwise. The output is sent as input to the other connected cells. With this simple schema authors where able to reproduce the AND/OR/NOT functions. In an era in which Artificial Intelligence was felt as the reproduction of the logical processes of the human brain, these results where considered promising.

Based on the artificial neuron, in 1957 Frank Rosenblatt [133], a psychologist, introduced the Perceptron: the binary inputs are multiplied by real valued coefficients called weights, and a further, constant input is introduced called bias (see fig. 1.1). The bias allows the output to be offset with respect to the inputs, so representing a wider range of relations between inputs and output. These improvements granted the perceptron a fundamental ability: learning.

Rosenblatt proposed an algorithm in which weights of the inputs are adjusted to reflect the importance of the related inputs to the final output. This paved the way to Supervised Learning: learning *by experience* an unknown complex function that relates inputs to known target outputs.

[1]Mayranna, CC BY-SA 3.0 `https://creativecommons.org/licenses/by-sa/3.0`, via Wikimedia Commons; link: `https://commons.wikimedia.org/wiki/File:Perceptron_moj.png`

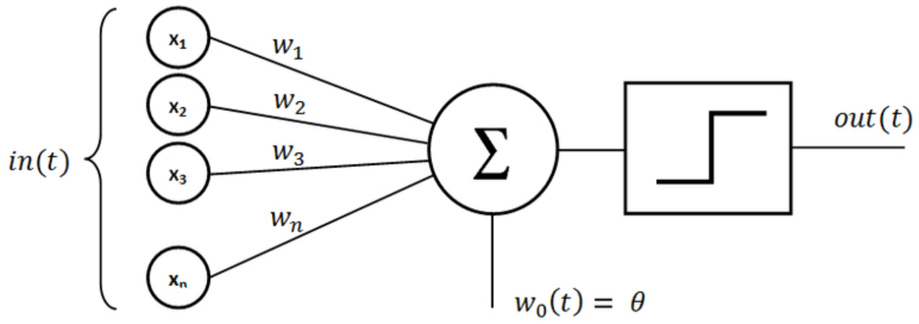


Figure 1.1: Rosenblatt Perceptron. Weights are indicated as $w_1, w_2, ..., w_n$ while w_0 is the bias. (Source:[1]).

Even if a perceptron can only have one output, connecting together multiple perceptrons each receiving the same input and returning an output was the strategy to address more complex problem such as multi class classification.

In 1960 Bernard Widrow and Tedd Hoff [163] proposed the Adaptive Linear Neurons - ADALINE. They eliminated the threshold function with the idea to obtain a differentiable output, which could be derived with respect to the weights of the inputs to find the minimum of the difference between the output and the target value. This was indeed a major step forward and demonstrated that learning a functional connection between input data and the related output was a problem of optimization, in principle similar to the well known linear regression. It is also the very basis of the Gradient Descent strategy, still today used to train neural network based systems.

The first models of neural networks implemented by perceptrons and the ability of such systems to learn by experience following an optimization strategy were in the lines of Connectionism [161], an approach in the study if human cognition based on mathematical models and in particular on neural networks. There was a great hope that systems implemented on the artificial neurons could lead to thinking machines, since in both biological and artificial brains the building elements where conceptually the same: neurons.

1.1.2 The first AI winter (1969-1980)

In 1969 Marvin Minsky and Seymour Papert [118] of the MIT AI Lab presented a book that analysed the achievements and outlook of the perceptrons. They pointed out that the relatively simple systems so far proposed, based on a single *layer* of perceptrons that elaborate the input signal and evaluate an output, could not be applied to real life problems: it could not even learn the XOR logic function, due to the non linear separability of the input domain. They also suggested that this issue could be overcome with more complex networks, in which more layers of perceptrons are arranged in a sequence and where the output of a layer is used as input of the following one. They suggested, in a word, a Multi-layer Neural Network architecture (see fig. 1.2), or Multi-Layer Perceptron (MLP) in which each layer extracts features that represent an abstraction of its input, and that will be further elaborated to a higher degree of abstraction by the next layer. Such architecture is way more “expressive” than the original one layer approach, and could address a wider range of tasks.

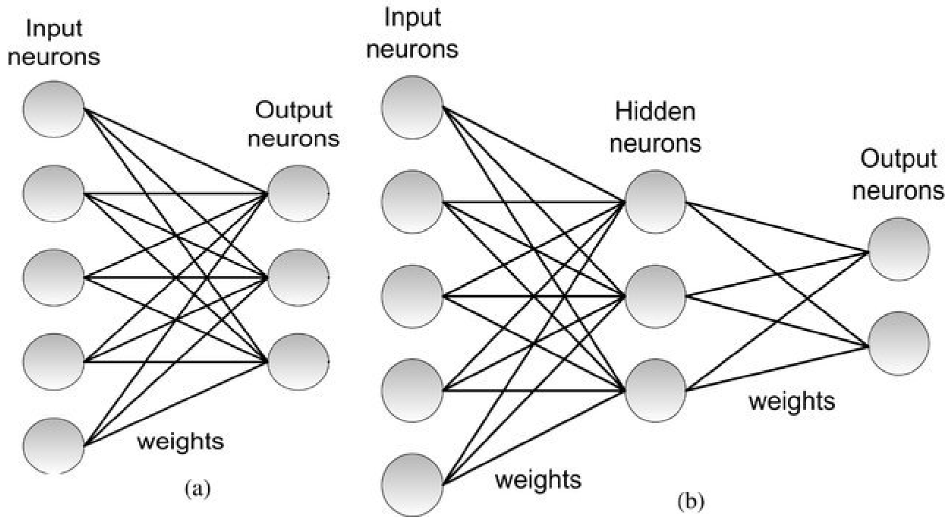


Figure 1.2: Single layer (a) and multilayer (b) feedforward networks. (Source: [24]).

Unfortunately this solution in turn raised an other issue: the straightforward optimization strategy proposed by Rosenblatt and refined with ADALINE was not feasible with multilayers systems. The reason is, it adjusts the weights applied to the input data to predict the final output, and can not deal with the weights of the intermediate, or *hidden*, layers. Even if a new training strategy would be found, the MLP was at that time doomed by the limitation of the resources: computing machines had no enough memory and speed to implement and train relatively complex NN architectures [45], as correctly predicted by the same Rosenblatt [134]: “as the number of connections in the network increases, however, the burden on a conventional digital computer soon becomes excessive”.

All in all, even if the survey of Minsky and Papert was not necessarily a fully negative assessment about perceptron, it was so perceived by the scientific community. A lot of expectations were addressed to NN methodologies and the sentiment was that they were the key tools to access AI. Consequently, after the publication of the survey there was a general sense of frustration about the whole matter. Reasearch and funding were redirected on more traditional and conservative approaches. The years between 1969 and 1980 are referred to as “the first AI winter”.

1.1.3 Neural Networks get maturity (1980-1993)

Research anyway continued in the background. Even if the MLP was an interesting solution for the limitations of the previous approaches, it lacked a suitable training strategy. During the sixties a number of attempts were made to overcome the problem, and in 1970 Seppo Linnainmaa [96] and then Paul Werbos [162] in his PhD thesis in 1974 proposed an algorithm: the Backpropagation. It solves the problem of the minimization of the error through different layers by applying the chain rule of derivation. This was a conceptually simple and elegant solution and made MLP trainable, but given the low interest in NN that followed the survey by Minsky and Papert, it was missed by the many. It was only in 1986 that the idea, again proposed by David Rumelhart, Geoffrey Hinton, and Ronald Williams [135], was accepted and ignited a new start for NN research. In 1989 Hornik *et al.* [68] demonstrated that “Multilayer feedforward networks are universal approximators”, a giant step forward with respect to the original perceptron, unable to reproduce the XOR function. More, in the same year LeCun *et al.* [91] succesfully applied NN and backpropagation for the very first time to a real life problem: the recognition of hand-written digits of US posts ZIP codes. The application introduced the Convolutional Neural Networks (CNN), described as weights sharing networks. They extend the original idea by Minsky of an architecture based on a sequence of layers, in which each layer receives as input the output of the previous, extracts features and then passes its output to the next layer. Here features are extracted by a sub-layer of reduced size, the filter, which shifts covering the whole input. In this way the relatively small amount of weights of the filter are shared for the whole input. CNNs were not a novelty, being for example discussed as the Neocognitron by Fukushima [48] in 1980. Nevertheless there was AI winter at the time, and the work was almost ignored.

As the methodologies improved, a lot of research fields were experimented. Efforts with the aim to address the problem of data compression led to the introduction of autoencoders in 1988/1989 ([22, 12]). Autoencoders are a well known example of Unsupervised Learning (UL): they take input data which are not labeled, and compress their representation. The training consists in the recovery of the initial form of the data starting from the compressed one. In this way the network learn the internal characteristics, or distributions, of the data. Unsupervised Learning can be used to classify data. In the hand written digits recognition problem, a network clusters images which show some similarities, even if it can not say which is the number they represent since data are unlabeled.

Then, after Supervised and Unsupervised Learning the last paradigm of machine learning, the Reinforcement Learning (RL), followed briefly. The aim of RL is to train a system, called the Agent, to execute a task by performing a sequence of Actions. To do so another system, the Environment, judges the so far executed Actions with respect to the completion of the task, and gives back a reward. Agent tries to maximize the total amount of rewards by completing the whole task. The RL paradigm was used in many application such as robotics [95] and control of dynamical systems [123], and in 1995 was responsible of one of the most spectacular achievements of the times: TD-Gammon [148], a system able to play backgammon and to beat human players.

Many attempts were experienced to apply RL to other, more complex, games: Go and Chess. An old problem arose again here, as pointed out by Thrun in the same year, 1995 [150]. RL requires a lot of calculations to evaluate the correct reward, and in complex games where the possible actions are a huge amount, this number tends to explode. At the times the power of calculus required to train such systems was simply beyond the machine possibilities. The problem was not in the model or in some aspect of its implementation, but in the resources availability. This aspect, again, played a major role in the rise of the second AI winter.

1.1.4 The second AI winter (1993-2004)

Research was conducted also in other fields. One demanding problem was the understanding of human speech. The specificity of the topic resides in the fact that the input is sequential, that is is an ordered set of elements. The basic idea to address the problem was already in the famous work about backpropagation by Rumelhart, Hinton and Williams, [135], in the form of Recurrent Neural Networks (RNN).

RNN are the very first NN models to break the feed-forward schema: elements of the input sequence do not pass from a given layer to the next, but are used to feed again the same one together with new elements of the sequence. The new approach required a general overhaul of the Backprogation algorithm to take care of the loops in the information flow. A new problem arose here: the recursive chain rule applied to evaluate the gradients, especially when specific activation functions are present, resulted almost always in a vanishing or exploding gradient. It was indeed unfeasible to train RNN with even sequences of data of ten elements or more. The problem was evident in RNN but was also present, for the same reason, in feedforward NN with many layers. It was a disappointing issue as the increasing amount of layers was the key to make NN systems successful. The vanishing gradient problem was eventually addressed by Hochreiter in 1991 [65], but was only in 1997 that the RNN training problem was substantially alleviated by the introduction of Long Short Term Memory (LSTM) [66].

As in the first AI winter, the sentiment of the scientific community was now distant from NN systems. The metodology was felt as complex and not at all easy to apply, the algorithms were not efficient and often needed substantial changes to work properly. Indeed, another problem was perceived as increasingly troubling: the growing dimensions of the state-of-the-art NN systems, which needed an amount of computation power well beyond the then available resources. This deverted for the second time the reasearch of the AI, and the relative funding, to other more stable technologies: Random Forests [64], or Support Vector Machines [34] for example.

1.1.5 The rise of Deep Learning (2005-present)

There were few groups of researchers that continued to work on NN in a shortage of funding scenario, even if at least one institution, the Canadian Institute for Advanced Research (CIFAR), continued to supply the research. In 2003 Yoshua Bengio [19] published an article in which exploited a previous idea of representing words as numeric vectors of many components (*word embeddings*), and used these vectors as input of NN. The paper resulted one of the most influential ever and paved the way of Natural Language Understanding (NLU) and Natural Language Processing (NLP).

However, the key to overcome the general loss of interest of the researchers was to grow up the networks by adding more *hidden* layers between the input and the output, and to find an efficient training algorithm.

In 2006 Hinton *et al.* [62] recognized that previous attempts to train complex and multilayer NN were frustrated by the inefficiency of the Backpropagation applied to RNN or feed-forward networks with many layers, and that this effect could be fixed by correctly initializing the network, for example with an unsupervised pretraining. The general problems with Backpropagation, such as the vanishing or exploding gradients, were eventually fixed in 2009/2011 by three different reaserach groups [74],[122], [53] who pointed out the activation functions as the origin of the troubles and indicated the best activation: the ReLU. With this choice there was no more need of an unsupervised pre-training. In the same 2007 Bengio [20] demonstrated that Deep Neural Networks (DNN) as they were referred to, were way more efficient with respect to the shallow networks when applied to complex problem. DNNs are able to extract features from input and generate an abstract representation of it, that is passed to the next layer; this layer can work on preprocessed data, in some way, and performs a further abstraction.

The growing of NN architecture and the potentials they embed drove the growing of the related counterpart: the data. There was an increasing need for labeled data to be used to feed DNNs. In 2009 Fei Fei Li *et al.* [38] released the first version of the ImageNet dataset, with the aim of create a general vocabulary of classified images taken from the web. It contained about 3.2 millions of images in 1,000 categories, and needed the employment of crowdsourcing to be elaborated. For these characteristics the publication of ImageNet represents a timestamp for the application of Big Data to NN. It was a giant leap forward with respect to the collections of hand written digits so far used to test Computer Vision systems, and allowed to take the best from the DNN.

The huge amount of weights in DNN make them able to deal with huge amounts of labeled data; in turn the availability of Big Data pushes the creation of new, more complex architectures to take the best from. At this point another aspect required attention: the need of computing power. CPUs were no more suitable for training with Big Data such systems, which easily contain millions of weights. A practical solution was the parallel calculus, in which more CPUs are used simultaneously. However, the very revolution was the introduction of GPUs, leveraged by the world of computer gaming. The seminal work in this new aspect was in 2012 by Mohamed *et al.* [120]: he employed GPUs to train DNNs for the task of Speech Recognition. GPUs are one or two order of magnitude quicker than CPUs and can consequently reduce the training times of complex networks, for example from weeks to days or hours.

1.1.6 Towards a new AI winter?

Undoubtely, thanks to the recent achievements of the first decades of the 21st century, NN research and Deep Learning in particular are now in the Golden Age. More, applications of NN are widely used today in real life: automatic translation (Google Translate [1], DeepL Translate [5]); voice and speech recognition (Google Assistant [4], Voice Dictation [7]); self-driving cars (Tesla Autopilot [3], Waymo [2]); and many other. The context can be easily represented by the famous and in some way historical “ImageNet Large Scale Visual Recognition Competition” (ILSVRC) in 2012. AlexNet [86], a DNN based on CNNs, outperformed humans in classification, obtaining a stunning top-5 error of 15.3%. AlexNet is paradigmatic of the research moment: a complex network with more than 62 millions weights, trained with parallel GPUs on large datasets. Another well known example of successful but huge model is Transformers [156], an encoder-decoder architecture. The encoder part, called BERT [39], was released in 2018. It has been vastly employed in Natural Language Processing and outperformed state-of-the art approaches in a series of tasks, igniting an era of great activity in the topic. BERT consists of roughly 110 millions parameters. Just two years after, in 2020, GPT-3 [23] outclassed BERT in terms of performance and in the amount of parameters, raising the number to 175 billions.

So, what can go wrong in the future? To match the requirements of better and better performance in the metrics of the specific tasks, the computational cost, amount of training data, model size, training time, inference latency, all together referred to as *footprint* metrics, are all growing in an unprecedented way. Recently, some conferences and journals ask the researches to report also computational information about the submitted articles, see for example the reproducibility checklist of ICML 2020 [6], and not only results based on the performance metrics specific of the task. Analogously, events have been organized with the aim to obtain *sustainable* results, that is a trade-off between performance and footprint; see for example SustaiNLP [8].

There are concerns for the near future about the “gigantism” of Deep Learning. The easy answer to these concerns is to increase the computing power, but it is to be considered that it is a finite if also increasing resource. With the actual pace it is not far the day in which a shortage of the general computing power will happen.

The argument has also ethical consequences: high power consumption means high carbon footprint, a sensible argument today. It could bring to an unfair distribution of resources, favouring a divide in the access of Deep Learning technology and results [167]. This trend could lead to a new general sentiment of frustration about the “false” promises of Deep Learning, and to a new AI winter. It is to underline that, given the importance of Deep Learning in real life today, the impact of this new stop can be dramatic.

1.2 The search for Efficient Deep Learning

This thesis is a contribution in the research of *Efficient Deep Learning*.

We analyze the current and near future scenario in Deep Learning research by the point of view of the computational cost. The analysis reveals that the actual trend, which is driven by the search of better and better performance, will be not sustainable in the future, and that a change in the research priorities is needed to avoid a potential shortage of resources.

This new paradigm in research is known as Efficient Deep Learning. It leverages a series of methodologies and techniques with the aim to lower the footprint metrics of the models, yet with a moderate or null decrease in the performance metrics. A reduction in the footprint means lower size models in terms of number of parameters, or a lower amount of training time as well as a reduced number of needed training samples: as a final consequence, a reduction of the overall computational cost.

To demonstrate the effectiveness of the proposed approach, we detailed three research projects which have taken advantage from Efficient Deep Learning design. They cover different topics in AI research, so showing the feasibility of the approach.

The first project is related to Computer Vision and consider the problem of video-based person re-identification, which is the task of associating videos of the same person captured by different and non-overlapping cameras. The approach proposes a Siamese framework in which video frames of the person to re-identify and of the candidate one are processed by two identical networks which share the same trainable weights and produce a similarity score. An attention mechanisms is introduced to capture the relevant information both at frame level (spatial information) and at video level (temporal information given by the importance of a specific frame within the sequence). One of the novelties of the proposed architecture is given by a joint concurrent processing of both frame and video levels, providing in such a way a very simple and compact architecture. Further, training data samples are augmented to improve generalization ability. Despite these aspects which contribute to lower the footprint metrics, the approach achieves better performance than the state-of-the-art on the challenging iLIDS-VID dataset.

The second project describes a model for Automatic Keyphrase Generation, a leading task in the topic of Natural Language Processing. Keyphrases are short phrases that summarize the semantic meaning of a given document. Several past studies provided diverse approaches to generate Keyphrases for an input document. However, all of these approaches still need to be trained on very large datasets. The proposed approach introduces a GAN architecture to address the problem of Keyphrase Generation in a low-resource scenario. The main contribution relies in the Discriminator's architecture: a new BERT-based module which is able to distinguish between the generated and human-curated KPs reliably. Its characteristics make it suitable in a low-resource scenario, where only a small amount of training data are available, and able to train an efficient Generator. The resulting architecture achieves, on five public datasets, competitive results with respect to the state-of-the-art approaches, using less than 1% of the training data.

The third project is about the prediction of the interatomic potentials in crystal materials, a relatively novel topic in Deep Learning applications. The prediction of the atomistic structure and properties of crystals including defects is essential for unraveling the nano-scale mechanisms that control the micromechanical and macroscopic behaviour of metals. Density functional theory is the traditional approach to the problem, and can enable the quantum-accurate prediction of some of these properties, however at high computational costs and thus limited to systems of $\sim 1,000$ atoms. In order to predict with quantum-accuracy the mechanical behaviour of nanoscale structures involving from several thousands to millions of atoms, machine learning interatomic potentials have been recently developed. The project explores the performance of interatomic potentials based on Graph Neural Networks (GNNs). Two state-of-the-art GNN models are considered. They have been modified to be compliant with a periodic crystal environment and trained on an extensive database of ferromagnetic bcc iron, a relevant material both in the research activity and in real life applications. GNN potentials proved to be effective in reproducing the volume and tetragonal distortion, as well as defected configurations (vacancy and surfaces), so demonstrating their capability of reproducing the energetics of defects in bcc iron. They also prove to be especially efficient approaches among the machine learning ones, as they leverage the message passing paradigm that introduces convolution and weight sharing in graph domains. Furthermore, the message passing guarantees scalability of the system at a relatively low computational cost.

All three research projects show that is possible to reduce the footprint metrics of a model while keeping performance metrics comparable with state-of-the-art approaches.

2 Computational limits and Efficient Deep Learning

DL architectures are *universal function approximators* [68] being able to learn the unknown and complex relation between input data and target values. The key aspect of this ability is *overparameterization*: there are much more learnable parameters, or weights, in a DL model then data points in the input dataset. Overparameterization makes the networks flexible and able to extract otherwise hidden features of the data. The well known downside is *overfitting*: the model learns precisely the characteristics of the training data but can not extend its knowledge to data never seen before. There are however a lot of regularization methodologies developed to counteract and mitigate overfitting: dropout [63], early stopping [175], weight decay in optimization algorithms [103], and so.

A great number of parameters, needed to deal with a great number of data points, means a great computational effort. In this chapter the past and actual trend of the required computation is analyzed, and a projection is also given in terms of computational cost *vs.* increase in the performance. Then a short survey of the methodologies developed to implement efficient DL models is presented.

2.1 The computational limits

2.1.1 Methodology and definitions

The measure of the computation required for training a Deep Learning model is not straightforward since there is traditionally a shortage of such info in publications. As the AI community is mainly performance-oriented, often details of the architectures are not explicitly given.

In what follow the computational costs are evaluated in terms of the number of floating point operations executed, *FLOP*, which is related to *FLOPS* (or *flops*, or *flop/s*), namely FLoating Operations Per Second, by the relation: $FLOP = FLOPS \times$

time, where *time* is expressed in seconds. The estimate of the number of FLOPs needed to train a model is carried out with two methodologies:

- if details of the implementation are known, then:

  $$FLOP = \#operations \times \#samples \times \#epochs \tag{2.1}$$

  where *#operations* is the total number of adds and multiplies for each training pass, *#samples* is the number of training samples, and *#epochs* is the number of training epochs;

- otherwise if details of the hardware are available, then:

  $$FLOP = \#processors \times FLOPS \times time \tag{2.2}$$

  where $FLOPS$ represents the computing rate of the processor, and *time* is the total amount of training time (in seconds).

2.1.2 The trend of the required computation

Amodei and Hernandez [9] conducted an interesting analysis on the past and actual trend of the computation effort required to train state-of-the-art systems. They distinguish two main eras, the border represented by the introduction of GPUs in the 2010s years. In the first era, the trend follows approximatively the famous Moore's law, with a 2-years doubling period. After 2012, starting from AlexNet and ImageNet contest, the doubling period is reduced to 3.4 months. Fig. 2.1 shows the results of the analysis.

Even if authors declare themselves confident in future progresses in the technology of microprocessors and in a better management of great hardware systems, it is clear that the trend is dramatically changed. As a figure, in the period 2012-2019 the number of Petaflop/s-day[1] has increased by more than 300.000 times.

2.1.3 Required computation vs. performance

Thompson *et al.* [149] introduces a simple theoretical model which connects the training computational cost with the increase of the performance. They use it as a benchmark to analyze present and future scenarios, showing that the trend is worse than expected by theory.

A rough estimate of the number of parameters in an overparameterized system is given by: $(\#params) \gg d \times (\#data)$ where d is the fixed dimension of the data point, and the number of the parameters scales with the number of data points. Note that a data point is a single sample in the input dataset, and generally is a multidimensional entity being composed by a set of *features*: e.g. a grid of pixels in Computer Vision, an embedding vector in Natural Language Processing, and so. Since the computational cost of training a model scales with the product $(\#params) \times (\#data)$, then: $computation = \mathcal{O}(\#data^2)$.

[1]A Petaflop/s-day consists of performing 10^{15} FLOPS for one day, or a total of about 10^{20} FLOPs. It is an estimate of the gross amount of the executed operations.

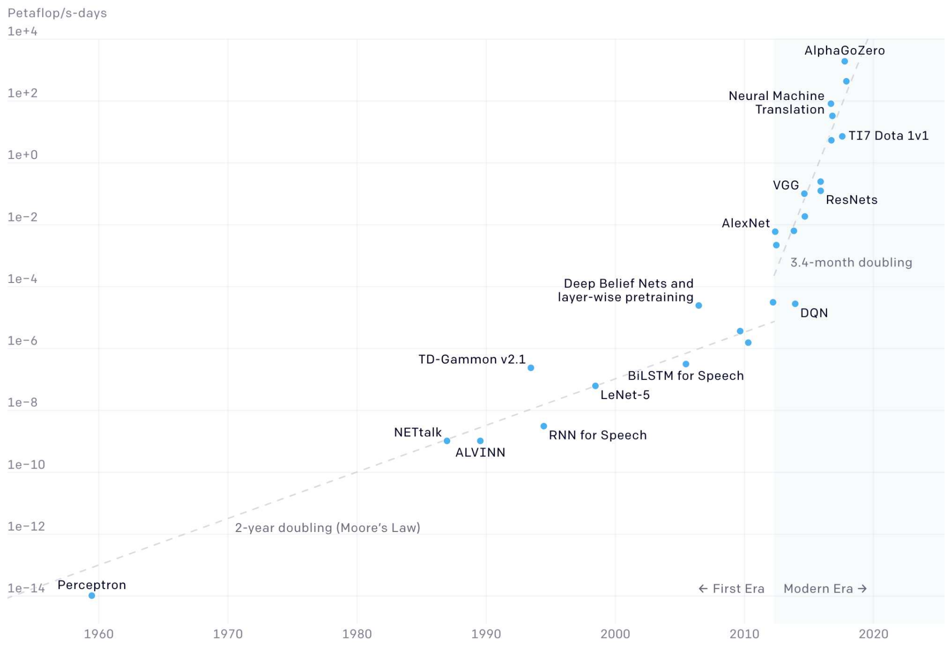


Figure 2.1: Computational cost for training state-of-the-art models, in number of Petaflop/s-day. The dramatic change in the slope of the trend is clearly shown. (Source: [9])

The performance P is related to the drop in the Root Mean Square Error (RMSE) [2] of the output of the given model, which in turn is related to the dispersion of its predictions around the target value, and scales as $1/\sqrt{(\#data)}$. So P scales as $\sqrt{(\#data)}$ and to improve linearly the performance of the model a quadratic increase in the number of data is required. Putting together the two trends, the required computation scales as the fourth power of the performance:

$$computation = \mathcal{O}(P^4) \tag{2.3}$$

with the care that the evaluations have been performed with the aim to find a lower limit.

2.1.4 Projections of the required computation

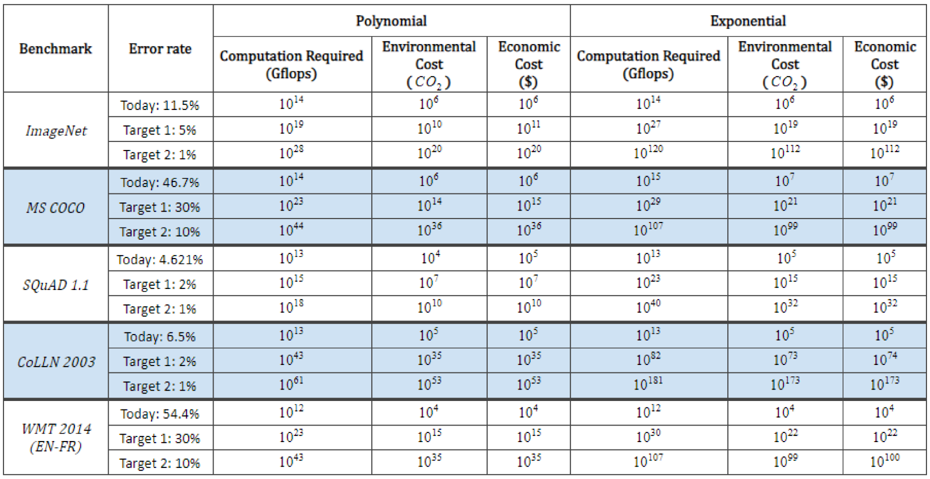
Authors [149] then tested the theoretical model of eq. 2.3 with real observations. Improvement of the performance in terms of required computation for training a model has been measured for 1,058 papers covering the main areas of Deep Learning Research:

[2]This is correct in a regression problem; note that a classification problem can be reduced to a regression by means of 1-hot encoding of d classes into a d-dimensional binary vector.

image classification, object detection, question answering, named-entity recognition, and machine translation. Results have been analyzed using a polynomial function: $computation = \mathcal{O}(P^{\alpha})$; an exponential one has also been tested. Fitting of the model with respect to sampled data are evaluated with the coefficient of determination R^2. Overall, it seems that the polynomial function fits better the data, but with a value of α that is much higher than the theoretical value $\alpha = 4$.

In particular, exponent α is about 9 for ImageNet if required computation is evaluated with the number of parameters of the model, or about 14 if estimated from hardware details (see sect. 2.1.1). This second method is used to evaluate the other tasks. They all have α of about 50, with the notable exception of $\alpha = 7.7$ in machine translation.

Figure 2.2 shows the projections of the required computation (GigaFLOPs) per performance increase (decrease of the error rate). Also projections for the Environmental and the Economic costs are represented, see also [143] for reference. All the predictions indicate that the exclusive search of the best performance could not be the best strategy in the future, as computational cost will probably explode.



Benchmark	Error rate	Polynomial			Exponential		
		Computation Required (Gflops)	Environmental Cost (CO_2)	Economic Cost ($)	Computation Required (Gflops)	Environmental Cost (CO_2)	Economic Cost ($)
ImageNet	Today: 11.5%	10^{14}	10^{6}	10^{6}	10^{14}	10^{6}	10^{6}
	Target 1: 5%	10^{19}	10^{10}	10^{11}	10^{27}	10^{19}	10^{19}
	Target 2: 1%	10^{28}	10^{20}	10^{20}	10^{120}	10^{112}	10^{112}
MS COCO	Today: 46.7%	10^{14}	10^{6}	10^{6}	10^{15}	10^{7}	10^{7}
	Target 1: 30%	10^{23}	10^{14}	10^{15}	10^{29}	10^{21}	10^{21}
	Target 2: 10%	10^{44}	10^{36}	10^{36}	10^{107}	10^{99}	10^{99}
SQuAD 1.1	Today: 4.621%	10^{13}	10^{4}	10^{5}	10^{13}	10^{5}	10^{5}
	Target 1: 2%	10^{15}	10^{7}	10^{7}	10^{23}	10^{15}	10^{15}
	Target 2: 1%	10^{18}	10^{10}	10^{10}	10^{40}	10^{32}	10^{32}
CoLLN 2003	Today: 6.5%	10^{13}	10^{5}	10^{5}	10^{13}	10^{5}	10^{5}
	Target 1: 2%	10^{43}	10^{35}	10^{35}	10^{82}	10^{73}	10^{74}
	Target 2: 1%	10^{61}	10^{53}	10^{53}	10^{181}	10^{173}	10^{173}
WMT 2014 (EN-FR)	Today: 54.4%	10^{12}	10^{4}	10^{4}	10^{12}	10^{4}	10^{4}
	Target 1: 30%	10^{23}	10^{15}	10^{15}	10^{30}	10^{22}	10^{22}
	Target 2: 10%	10^{43}	10^{35}	10^{35}	10^{107}	10^{99}	10^{100}

Figure 2.2: Computational (in units of GigaFLOPs), Economic ($USD) and Environmental (lbs of CO_2) costs projections per requested Error rate, for five benchmark datasets. Note that the smaller the Error rate, the higher the Performance. Both polynomial and exponential fitting functions are considered. (Source: [149]).

The positive consideration here is that the wide difference in order of magnitude between the theoretical model and the real life implementations implies a proportional margin of improvement in the efficiency of the latter.

2.2 Methodologies for efficient DL: a survey

Given the burden of the analyzed computational effort, it is more and more important to apply techniques and methodologies with the aim to improve efficiency in DL models. The problem can be described as the search of the Pareto Optimality [119] in which the competing aspects are the performance and the footprint, each represented by convenient metrics, and is illustrated in figure 2.3.

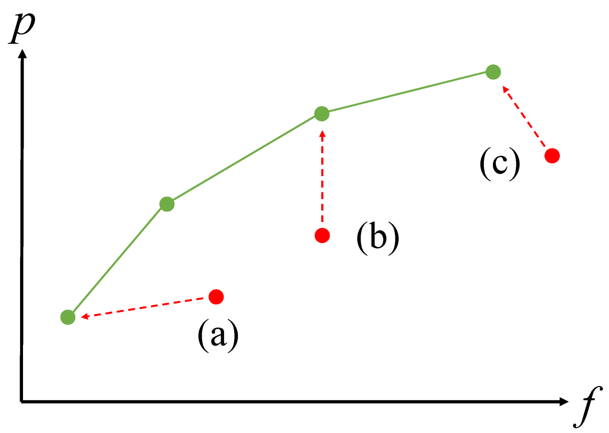


Figure 2.3: Pareto Optimality. Axes p and f represent performance and footprint in convenient metrics. Green dots and line are the optimal models and the Pareto front; red dots are sub-optimal models. Paths toward the Pareto front are represented for: (a), compression techniques; (b), training techniques; (c), network implementation techniques.

Optimal solutions represent the Pareto front (green dots and line in figure) and are by definition the best compromise between performance and footprint: any improvement in one of the aspects for these points is necessarily obtained at the expense of the other. For all the other solutions under the Pareto front (red dots in figure), improvements are possible in one or both the variables.

Following the practical survey by Menghani [116], some convenient approaches are illustrated in the following. They are all aimed at shifting the given model towards the Pareto front, and are grouped with respect to the path they describe.

2.2.1 Compression techniques

The aim of this set of techniques is to gain efficiency by means of a direct cut in the footprint metrics such as the number of parameters, memory size, training time, with a cut in the performance which is kept as little as possible. Their path toward the Pareto front is illustrated in fig. 2.3 (*a*).

Pruning. In Pruning [35, 59] the compression is realized by directly cutting a subset of the parameters of a given layer. Once pruned the network passes from a fully connected to a sparsely connected one, where the sparsity can be computed as $s = 1 - \frac{|W'|}{|W|}$ with W, W' the numbers of parameters before and after the cut, respectively. Among the strategies used to determine the parameters to be cut, *saliency* [92] evaluates a score for each parameter which is related to its effect in the final output: the lower the score, the higher the probability to prune. More in detail, saliency score of the parameter w_i is generally evaluated as the second derivatives of the loss function with respect to the parameter, $\frac{\partial L}{\partial w_i^2}$: pruning the lowest scoring parameters implies a scarce if any impact on the loss function and on the final output.

Pruning shows some resemblance with dropout which is used during training as a strategy to counteract overfitting, but has a permanent character. Nevertheless, it can help to improve flexibility and generalization ability of the network.

Quantization. Weights and activations inside a network are in general 32-bit floating-point values. Passing to a data type with lower precision reduces the footprint of the model in terms of size (less memory needed to store data) and inference latency (computation cost). Typically the target data type is the widely supported 8-bit fixed-point integer. In the quantization scheme [85], any given floating point value in the range $[x_{min}, x_{max}]$ is mapped in the interval $[0, 2^b - 1]$ by a step function, where b is the number of digits of the fixed point type, usually $b = 8$. Inference need a *dequantization* pass, which maps back the values in the interval $[x_{min}, x_{max}]$, with an acceptable loss of precision. Reduction in the memory required to store each value is of the order $32/8 = 4\times$.

Quantization can also be applied to reduce inference latency. In this case all the layers *and* the activations work with quantized values, and there is no need to dequantize for inference. The difference with the previous case is that here all the operations are performed in fixed-point integer; passing to a 8-bit configuration can improve up to a $3\times$ the latency time [154].

Early Stopping. Introduced by [175], it is a straightforward methodology to optimize the training process. Training is run as long as it improves the performance metrics of the trainee model. If there is not any improvement for a fixed number of iterations (*patience*), training is stopped. To avoid bad decisions in the first epochs, which generally show unstable metrics, the control is activated from a starting epoch on. Since the performance is evaluated on test data, early stopping also reduces overfitting of the model.

The gain of early stopping is mainly in the reduction of training times, as well as a reduction of human activity. Both factors have a relevant economic impact. Also to be noted is that early stopping does not reduce performance.

2.2.2 Training techniques

These methodologies are used in training phase and are aimed to obtain better performance metrics by the introduction of different training strategies, in some way supplementing the usual supervised learning. Their path to the Pareto front is illustrated in fig. 2.3 (*b*). In some cases the gain in performance can be then traded off with a cut in some footprint metrics, with a moderate impact on the improved performance. For example, if the model to deploy meets the required performance but exceeds the memory limits of the release device, its performance can be boosted with training techniques, and then its dimensions can be shrunk using some of the compression techniques, reducing the performance close to the original (see [116]).

Data Augmentation. First introduced and widely employed in Computer Vision [86], this technique transforms available training data to generate new, synthetic labeled data. Transformations like small rotations and scaling, or translations in some directions, generate effectively new images with the same label of the starting one. Training a network with augmented data can improve the flexibility of the model, as it can better generalize inherent features of the images. Other implementations of data augmentation imply for example mixing of images, with a consequent mixed label given by a combination of the two. The aim in this case is to learn hidden features that are common to both merged images.

Data augmentation has been also introduced in Natural Language Processing [178]: transformation in this case is performed by translating a text in a given language and then *back-translated* to the former language. Texts contain about the same information, but the aspect is different.

Data augmentation improves the performance metrics in terms of classification accuracy by making trained models more able to generalize, with no need of new labeled data. Moreover, it can also be employed to reduce the needed amount of labeled data.

Self-Supervised Learning. Given the great shortage of labeled data with respect to the great abundance of unlabeled ones, in Self-Supervised Learning (SSL) the models are trained with unlabeled data and then fine-tuned on labeled ones. More in detail, the process is performed in two phases. First, the model is trained on a pretext task with a large collection of unlabeled data, and tries to learn a general representation of the internal structure of the data. Then, the pretrained model is fine-tuned with a limited amount of labeled data to address the specific task. A well known example of this approach comes from Natural Language Processing: BERT [39], the Bidirectional Encoder Representations from Transformers [156]. It is trained on two pretext tasks: the first is to predict some words (15% of the total) that are masked in a text, the other is to predict whether a given sentence A is followed, or not, by a given sentence B. The unlabeled dataset consists of the BooksCorpus (800M words) [191] and English Wikipedia (2,500M words). This makes BERT able to generate a strong representation of the internal structure of the language. Then the model is put in an otherwise thin architecture with only few layers fitted for the task output, and the resulting system is trained with the labeled data. BERT achieved outstanding results, outperforming other dedicate models in eleven NLP tasks.

The great advantage of architectures which rely on pretrained SSL models is that they need a reduced amount of labeled data to obtain state-of-the-art performance. Labeled data are costly since are to be elaborated by humans, so SSL reduces substantially the general costs of training. Together with the aforementioned data augmentation, it is a well known data-efficient approach.

2.2.3 Network implementation techniques

A great efficiency in an architecture can be obtained starting from the very basic building blocks: an accurate choice of the type and of the sequence of the layers, for example. The advantage of this set of techniques is that they can be applied to a wide range of architectures; moreover, their path to the Pareto front marks improvements in both performance and footprint, see fig. 2.3 (*c*). For these reasons they are largely employed.

Convolutional Layers. In a fully connected architecture each neuron of a layer is connected with each neuron of the next layer. To elaborate an image of, say, 100×100 pixels with 3 channels (RGB), a fully connected layer needs 3×10^4 trainable parameters. In a Convolutional Layer (CL) architecture the parameters are indeed shared, and are in a much smaller number: a low size matrix of trainable parameters, the filter, shifts over the input aggregating the pixels values it covers, and generates a new, reduced, features map. If the filter has, say, a 5×5 size, and considering a different filter for each channel, then the number of the trainable parameters is $5 \times 5 \times 3 = 75$. In general CLs are followed by pooling layers (PL) which further downsizes the features maps.

More, convolutional layers have the ability to extract features from the image which are independent with respect to translations, as the action of the filter is to compress the features regardless of their position. Fully connected layers miss this ability because they address each pixel with a set of unique parameters, so lacking the relations with other neighbouring pixels.

In a typical Convolutional architecture a set of CLs+PLs blocks are put in sequence, and the features extracted by one block are passed to the next block. At each block, features of higher abstraction are extracted: simple lines or areas with different luminosity in the firsts, more complex shapes and aggregations of shapes in the lasts. All state-of-the-art image classifiers are based on the CL architecture: AlexNet [86], Inception [145], ResNet [60].

Attention Mechanism. Attention mechanism was introduced by Bahdanau [11] to fix the well known bottleneck issue of the encoder-decoder, an architecture mostly used to address Natural Language Processing tasks. The encoder, implemented by means of Recurrent Neural Networks, elaborates a text by accessing each word, or *token*, sequentially. At step i it takes token i and updates consequently its hidden state h_i, then at step $i+1$ the input is given by token $i+1$ and previous hidden state h_i and a new hidden state h_{i+1} is generated. At last step T the hidden state $c = h_T$, called the *context*, contains a compressed (encoded) representation of the whole input text. Then the decoder, also implemented with RNNs, takes the context and at each step generates an output token. Here the problem: the decoder can access only the context h_T, not all the hidden vectors, and has to decode an output token at each step starting from the very same input. Further, the context h_T will be more "similar" to the last hidden states of the encoder, that is the last input tokens, as the first will be easily "forgotten" during the encoding sequence. The attention mechanism fix the problem by generating a context vector for each decode step: $c_i = \sum_j^T \alpha_{ij} h_i$ where α_{ij} is a matrix of trainable weights, the *attention scores*. Decoder now can access all the encoded representations of the input, and the attention scores gives a measure of the similarity or *alignment* between the i-*th* input token and the j-*th* output token. Considering that in the past the bottleneck issue was addressed by increasing the dimension of the hidden state, the attention mechanism has the effect to lower the number of trainable parameters.

Transformers [156] is a large encoder-decoder architecture fully implemented leveraging linear layers and attention mechanism. This design choice allows Transformers to execute highly efficient parallel calculations and to improve the training process. Indeed, processing all the input tokens at a single step reduces the burden of learning long range dependencies in the input text from $\mathcal{O}(n)$, with n number of input tokens, to $\mathcal{O}(1)$. More, parallel calculations make Transformers able to be trained with multi-GPU or TPU hardware, reducing the FLOPs of $300\times$ with respect to similar CNN-based or RNN-based architectures.

3 Video-Based Convolutional Attention for Person Re-Identification

3.1 Introduction

Given an image or video of a person taken from one camera, the Re-Identification task (ReID) is the process of re-associating the person by analyzing images or videos taken from a different camera with non-overlapping field of view. Although humans can easily re-identify others by leveraging descriptors based on the person's face, height, clothing, and walking pattern, ReID is a difficult problem for a machine to solve, since it should deal with features between cameras like different lighting conditions, different point of views or person occluded by objects or other people.

Traditionally many attempts to explore the problem has been proposed for still images (e.g., [112, 110, 97, 111, 109, 98]), while recently some research groups have experimented approaches based on video images (e.g., [187]). Using videos for Re-Identification provides several advantages over still images. The video setting is a more natural way to perform Re-Identification, as a person will normally be captured by a video camera producing a sequence of images rather than a single still image. Given the availability of sequences of images, temporal information related to a person motion may help to disambiguate difficult cases that arise when trying to recognize a person in a different camera. Furthermore, sequences of images provide a larger number of samples of a person appearance, thus allowing a better appearance model to be built. On the other hand, this large set of information needs to be treated properly.

To address this challenge, we propose an approach to the problem of video-based person re-identification that is characterized by two main aspects. First, we propose a deep neural network architecture based on a Siamese framework [114] which evaluates the similarity of the query video to a candidate one. Second, we introduce a novel spatio-temporal attention mechanism with the aim to select relevant information from different areas of the frames of the input video, and from their evolution over time. Attention mechanisms have been largely exploited in a variety of different implementations and in many different domains of Deep Learning such as Natural Language Processing [125] and Computer Vision [142]. The intuition behind Attention in Computer Vision is to mimic the human visual process. Humans give different importance to different areas in an image as they are able to focus on *hot* areas and neglect others [33]. This improves greatly the ability to recognize structures and patterns in otherwise flat data. Nevertheless there are relatively few attempts to use Attention in the field of Automatic Re-Identification. [99] proposes integrating a soft attention based model in a Siamese network to focus adaptively on the important local regions of an input image pair. [172] uses a spatial pyramid layer as the component attentive spatial pooling to select important regions in spatial dimension. [142] proposes a spatial attention module focused on recognizing the skeleton to identify the poses, and then a temporal module to recognize the actions.

Unlike other approaches, which use at least two separate modules to identify spatial and temporal features, we use a joint module to identify both at the same time. This allows us to define a simpler architecture which provides state-of-the-art performance on the well-known iLIDS-VID dataset.

This work [181] has been presented at the "International Conference on Image Analysis and Processing (ICIAP) 2019"[1].

3.2 Related work

The interest for video-based Person Re-Identification has increased significantly in recent years [157]. The aim of the first works was to manually extract feature representations invariant to changes in poses, lighting conditions, and viewpoints. Using these features, they proposed distance metrics to measure the similarity between two images. In particular, one of the first studies computes the spatio-temporal stable region with foreground segmentation [49]; while [32] employs more compact spatial descriptors and color features, constructed by using the manifold geometry structure in video sequences.

With the advent of Deep Learning approaches, Convolutional Neural Networks (CNNs) have been introduced in visual recognition tasks yielding to considerable improvements in the performance [87] with respect to more classical solutions [130]. In fact, CNNs are able to extract different features from a given image, representing them as a set of output maps avoiding manual effort in feature engineering. Image-based Automatic Person Re-Identification is one of the fields in which CNNs achieved remarkable results [127, 153, 155, 168, 182, 57].

However, considering that Person Re-Identification is usually done in settings that involve, for example, surveillance cameras, it is easy to argue that image-based person re-identification is no more an adequate schema to address current needs.

This led to most recent works that began exploring video-based person re-identification [94, 100, 114, 159, 172, 173, 189, 190], a setting closer to real-world applications. Videos have the advantage to contain temporal information that is potentially helpful in differentiating between persons. For example, in [114], the proposed CNN model extracts

[1] `https://event.unitn.it/iciap2019/`

features from subsequent video frames that are fed through a recurrent final layer in order to combine frame-level features and video-level features.

Not all the parts of an image or of a video are equally important and humans place more focus only on some of them, assigning little to no importance to the rest. This attention mechanism has been adopted in a variety of applications, such as machine translation [11], action recognition [140], image recognition [10] and caption generation [171]. Recently, Attention models [140, 142] have been proposed for video and image understanding. These models assign weights to different parts of each frame, making some of them more important than others. In particular, [99] proposes integrating a spatial attention based model in a siamese network to adaptively focus on the important local parts of an input image pair.

With respect to the existing literature, [189] and [131] are the most similar to our approach. [189] uses a Recurrent Neural Network (RNN) to generate temporal attentions over frames so that the model can focus on the most discriminative ones in a video. [131] instead directly calculates the attention scores on frame-based features, using a simple architecture with two separate temporal and spatial modules. Our approach exploits a single attentive module to extract both temporal and spatial features from frames at the same time, resulting in an even simpler architecture that provides state-of-the-art performance.

3.3 The proposed approach

The proposed approach (see Fig. 3.1) is based on a Siamese network [114]. This schema is composed by two identical networks, or branches, in which the first is fed with the query video and the second with the candidate video to be compared. Each branch includes a sequence of modules that will be described in details in next sub-sections. The parameters of the two branches are shared. The output of the Siamese network is a value that represents the similarity of the two input video sequences in terms of the distance between their respective features vectors, which should be close to zero if they belong to the same person, close to one otherwise.

3.3.1 Spatio-Temporal Attentive Module

The Spatio-Temporal Attentive Module is the core module of the proposed architecture. It aims to identify the portions of a frame which an human eye would normally focus on. Those areas should contain relevant spatial information, and we want to exploit them to improve the re-identification performance. Since the input frames are enhanced with the temporal information of the optical flow, both spatial and temporal features will be exploited by this network.

Inspired by [33], we propose to use a particular combination of convolutional network and LSTM, called Attentive ConvLSTM, capable of working on spatial features, in which the internal state of the network is given by the standard LSTM state equations where the matrix products between weights and inputs are replaced by convolutional operators. The ability to work with sequences is exploited to process input spatial features iteratively. The general idea of how this module works is shown in the bottom part of Fig.3.2.

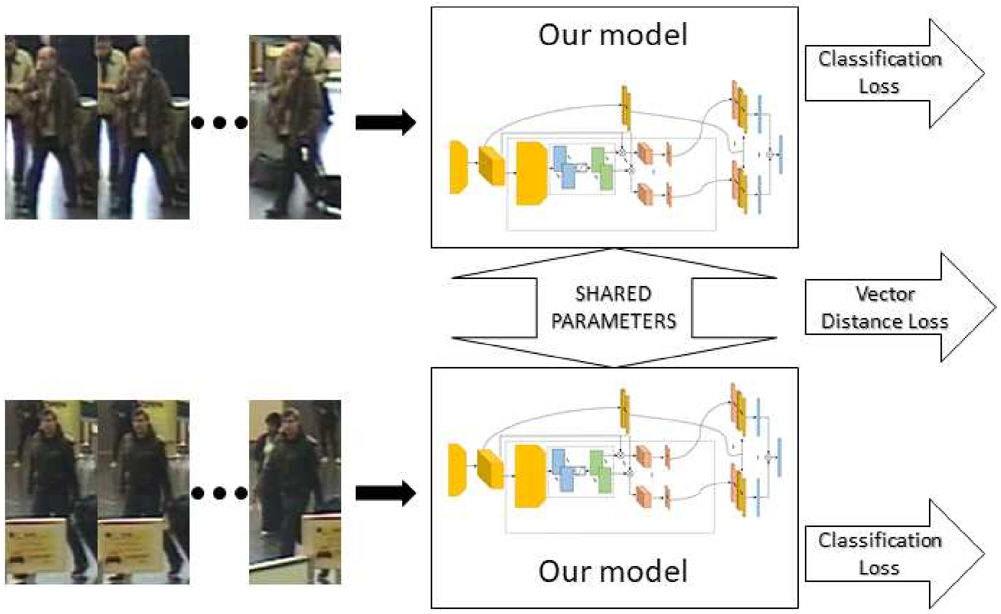


Figure 3.1: Siamese network scheme. Each network receives as input a person image sequence to classify. The loss is calculated as the sum of the classification error of each network, plus the Euclidean distance between the two descriptive vectors, which should be close to zero if the two sequences belong to the same person, or close to one if they belong to different people.

Our aim is to exploit attentive maps to better identify relevant features of frames and provide state-of-the-art performance while using a simple network. The architecture of each branch (see Fig. 3.2) is based on an initial convolutional network to reduce the image size, an attentive model to generate attentive maps, a fully connected layer to extract significant features from the original frames, and a final part where the features are combined.

More in details, the architecture consists of a ConvLSTM to recurrently processes attentive features at different locations of the frame, focusing on different regions of the tensor. A stack of features $\mathbf{X}$ is repeatedly given in input to the LSTM, which sequentially updates an internal state based on three sigmoid activators. Update is performed by two blocks: the Attentive Model, and the ConvLSTM. The Attentive Model generates an attention map using a convolutional layer that takes as input the original $\mathbf{X}$ and the previous hidden state, followed by a `tanh` activation function and another convolutional layer, and finally normalized with a `softmax` operator. The resulting output represents a normalized spatial attention map, which is then applied to the original $\mathbf{X}$ with an element-wise product, resulting in the filtered $\mathbf{X}'$. In ConvLSTM, each of the three sigmoid activators is given in input the sum of two different convolutive layers, the first taking as input $\mathbf{X}'$ and the second taking as input the previous hidden state, and a bias. The output of the first sigmoid is then multiplied element-wise with the

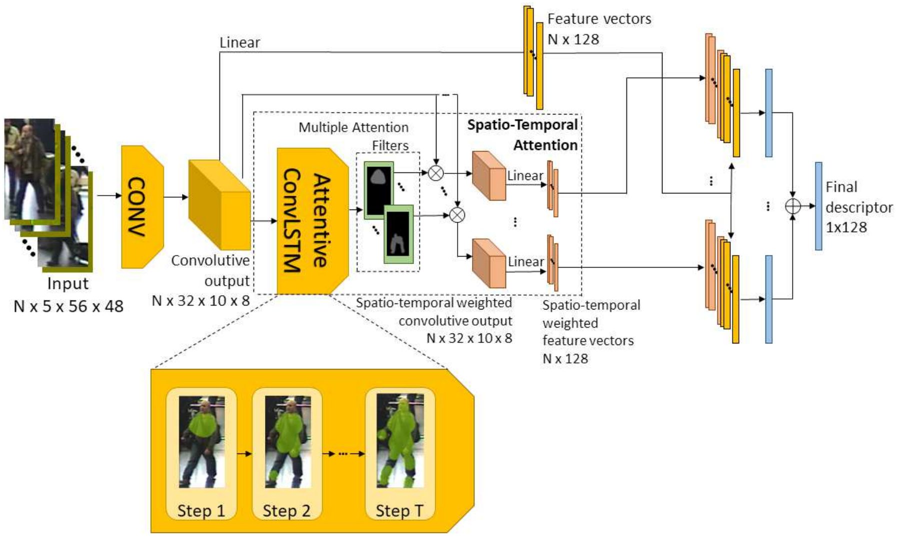


Figure 3.2: Detailed Network scheme. The main blocks are the initial convolutional network, the Spatio-Temporal Attention Module, and the final part which performs averaging and normalization. The bottom part gives an idea of the multiple refining steps.

previous $\mathbf{X}'$, the output of the second sigmoid is multiplied element-wise with the state of the LSTM memory cell, and the two resulting outputs are summed together and fed to a `tanh` activator. The result is multiplied element-wise with the output of the third sigmoid, and the resulting tensor is the new hidden state.

The Spatio-Temporal Attentive Module takes in input an image and produces in output multiple attentive maps, using an iterative refinement in T steps (based on our preliminary experiments, we set $T = 10$). We then apply those maps to the original input and obtain multiple different filtered outputs. Ideally, each filter should focus on a different spatio-temporal feature of the frame.

3.3.2 Architecture details

The starting input (see Fig. 3.2) consists of video sequences composed by a batch of N frames, each frame has size 56×48, with 3 channels for the YUV, plus 2 for the vertical and horizontal components of the optical flow, for a total of 5 channels.

The input is first processed through a convolutional network which consists of 3 stages, each composed by convolution, max-pooling, and nonlinear activations. Each convolution filter uses 5×5 kernels with 1×1 stride and 4×4 zero padding. This outputs a batch of size $N \times 32 \times 10 \times 8$.

At this point, the model branches in two lines: the same input is passed to the Spatio-Temporal Attentive Module previously described, and to a fully connected layer preceded by a dropout applied with $p = 0.6$ probability. The first aims to output multiple spatio-temporal-filtered feature vectors for each frame, and the second a general feature vector for each frame.

Spatio-Temporal Attention generates multiple attentive filters. Each of these filters has size 10×8, is first normalized with a sigmoid between 0 and 1, and then applied with an element-wise multiplication to the original output of the first convolutional network, obtaining multiple blocks weighted with a different filter with the same dimension of the input, $N \times 32 \times 10 \times 8$; each of these blocks focus on a specific zone of the frames. A final fully connected layer generates, for each block, a batch of spatio-temporal-weighted feature vectors of size $N \times 128$. This final layer is also preceded by a dropout with p=0.6. In our model, since we generate 3 filters, we obtain 3 spatio-temporal-weighted feature vectors.

The two branches of the network are then merged together, and the general feature vectors are concatenated with each of the spatio-temporal weighted feature vectors, resulting in 3 combined-feature vectors of size $2N \times 128$. Finally, each of these batches is averaged, normalized using L2 normalization, and lastly summed together, obtaining a final feature descriptor of size 1×128.

3.4 Experimental results

Our approach has been tested and evaluated on the public iLIDS-VID benchmark [159], since it is a challenging dataset that contains many occlusions, severe illumination changes and background clutters. It is also widely used in literature and it is then easier to fair compare our results. The iLIDS-VID dataset consists of videos of 300 distinct people. For each person there are two different video sequences, captured by two non-overlapping cameras. The video sequences have a varying number of frames, with the shortest sequence having 23 frames long and the longest having 192 frames, averaging at 73 frames.

3.4.1 Experimental setup

To be comparable with literature, we follow the experimental setup proposed by [114]. The dataset is randomly split in two: 50% of the people form the training set and 50% the test set. During the execution of the experiments, a different train/test split is computed for every repetition and the final results are then averaged. The network is trained for 1500 epochs using Stochastic Gradient Descent algorithm. One epoch consists in showing the Siamese network an equal number of positive sequence pairs and negative pairs, sampled randomly from all the persons in the training set, alternatively.

A positive sequence pair consists of two full sequences of arbitrary length, recorded by two different cameras, showing the same person. Analogously, a negative sequence pair shows two different persons. During the training phase, the length of the sequences is set to 16, that is, 16 consecutive frames belonging to a person are randomly sampled and used during this phase. As in [159], the first camera is the probe and the second the gallery.

All the images in the dataset go through a preprocessing step where they are converted from the RGB to the YUV color space and each color channel is normalized in order to have zero mean and unit variance. The three color channels are expanded with two more channels corresponding to the horizontal and vertical component of the optical flow computed between each pair of consecutive frames using Lucas-Kanade algorithm [106]. The two optical flow channels are normalized to bring them within the $[-1, 1]$ range.

Data augmentation is applied to each sequence during the training phase in order to increase the diversity of the training sequences. Each frame in the sequence undergoes cropping and mirroring, the same transformation is applied in the same way to all the frames belonging to the same sequence.

The testing phase is performed considering a video sequence belonging to the first camera as probe and a video sequence belonging to the second camera as gallery. In this phase, we use up to to 128 frames to form a sequence. The frames are always the starting frames for the probe, and the ending frames for the gallery. If this is not possible, because a person's sequence does not have enough frames, we consider all the available frames.

All tests are performed 10 times with different seeds, each time presenting the model different people for training and testing.

3.4.2 Results

First we compared the results of our model when using different numbers of filters for the Spatio-Temporal Attention Module, as shown in Table 3.1. We found that performance increases when generating more filters, but with four or more the model saturates and the performance starts decreasing.

# filters	Rank-1	Rank-5	Rank-10	Rank-20
0	60.5	84.8	93.0	96.9
1	59.4	85.7	93.2	97.4
2	63.0	**87.7**	93.9	97.3
3	**63.3**	87.4	**94.0**	**97.8**
4	59.6	87.2	93.9	97.7

Table 3.1: Average results obtained using an increasing number of filters.

Second, we present experimental results with 3 filters on sequences of varying lengths between 2 and up to 128 frames, and the results are shown in Table 3.2. Note that if a person's sequence does not have enough frames, we still consider all the available frames and that in all cases the training has been performed using a fixed length sequence of 16. As one could expect, it is confirmed that the performance increases as the number of frames in sequence of frames grows, as also noted by [114]. Since the average sequence length in the dataset is 73, the performance does not increase much between 64 and 128, because most sequences are not long enough to benefit from the additional length.

Finally, we present the comparison of our model with the state-of-the-art in Table 3.3. Despite beeing a simple architecture, our solution outperforms other methods proposed in the literature on 2 metrics out of 4. Note that [131] claim better results on their paper, but, in order to provide a fair comparison, we re-ran their provided code on our dataset splits. In addition, for the sake of completeness we report the performance of

Length	**Rank-1**	**Rank-5**	**Rank-10**	**Rank-20**
2	16.7	37.7	50.9	64.6
4	22.7	46.9	60.3	72.6
8	31.7	59.3	71.3	84.2
16	43.8	72.6	83.9	91.4
32	53.9	80.7	89.0	95.3
64	61.0	85.6	92.5	96.7
128	63.3	87.4	94.0	97.8

Table 3.2: Average results with different sequence lengths (expressed in frames).

[101] as well, even if their testing protocol is not directly comparable with the others, as they always use all the available frames.

The simplicity of our architecture comes from the choice of making the spatial and temporal module work jointly. In fact their output is merged in order to, hopefully, get the best of the two and select only the most relevant information obtained by their combination.

Methods	**Rank-1**	**Rank-5**	**Rank-10**	**Rank-20**
Proposed Approach	**63.3**	**87.4**	94.0	97.8
Rao et al.[131]	62.2[1]	86.8	**94.8**	97.8
Xu et al.[177]	62.0	86.0	94.0	**98.0**
Zhang et al.[184]	60.2	85.1	-	94.2
McLaughlin et al.[114]	58.0	84.0	91.0	96.0
Zhengl et al.[186]	53.0	81.4	-	95.1
Yan et al.[173]	49.3	76.8	85.3	90.1
Liu et al.[101]	68.0[2]	86.8	95.4	97.4

Table 3.3: Comparison with state-of-the-art methods (iLIDS-VID dataset).

3.5 Considerations about efficiency

Video based person Re-Identification, as a part of Computer Vision general topic, relies greatly on Convolutional Neural Networks. Based on weights sharing, convolutional layers allow a great reduction in the number of trainable parameters, with a consequential cut in the required computation. Furthermore, the proposed architecture exhibits other solutions aimed at reducing the overall complexity, yet obtaining results comparable with state-of-the-art approaches.

- The choice of a siamese network architecture is a key element of the proposed approach. It allows to process the query video sequence and the candidates one at the same time, and to evaluate their difference. Weights of the two branches

[1]These results were obtained in our tests on the code provided by the authors, and are substantially lower than claimed in the paper

[2]Results are shown for completeness, but are not directly comparable

are shared: this characteristic is aimed to minimize the possibility to extract very different features from very similar, or the same, videos. It also means the half of trainable parameters to elaborate the same amount of input data, with a reduction of required memory.

- The model introduces a spatio-temporal attention based on a single module, whereas previous approaches manage the two aspects in specific modules. The core of the system is the ConvLSTM layer, which elaborates in a temporal sequence the attention maps generated by a spatial Attention model. Operations inside the ConvLSTM are executed via convolutions and so also this core component enables weight sharing.
- Data augmentation has been used to improve the generalization ability of the system. This technique extends the training dataset by generating new samples that are modified version of the original ones, so reducing the amount of original data needed to train the model with the same accuracy.

The solution proposed successfully addresses the demanding task of video based person Re-Identification by means of a light and efficient model based on a siamese architecture which leverages spatio-temporal attention module, and of an implementation that relies on ConvLSTM. Trained with an adequately augmented benchmark dataset, it obtains results comparable with state-of-the-art approaches.

4 Keyphrase Generation with GANs in Low-Resources Scenarios

4.1 Introduction

A Keyphrase (KP) is a piece of text that conveys the main semantic meaning of a document. KPs can be either present (or extractive) or absent (or abstractive): present KPs are exact substrings of the document while absent KPs are not. Their automatic prediction is an important challenge for the community research as KPs are a key component for a wide range of applications such as text summarization [185], opinion mining [21], document clustering [58], information retrieval [75] and text categorization [73].

Historically, the first approaches focused on simply extracting substrings of the text to be used as keyphrases candidates [176, 105, 183].

Recently, the research community has focused on the broader task of Keyphrase Generation [115, 27, 28]. Keyphrase Generation aims to *produce* a set of phrases that summarize the essential information in a given text, as opposed to simply *look for them* in the text. This allows for greater flexibility.

Several approaches introduced generative models based on the Encoder-Decoder architecture [115, 27]. This architecture works by compressing the contents of the input (e.g. the text document) into a hidden representation using an Encoder module. The same representation is then decompressed using the Decoder module, which returns the desired output (e.g. a sequence of KPs). The modules are trained jointly to learn the best intermediate representation to perform this mapping.

More recently, an approach based on GAN (Generative Adversarial Networks [55]) architecture has been proposed to address the task [144]. Although all these solutions achieved interesting results, they require a very large amount of data in order to be trained.

Our aim is to improve training efficiency, so that a model can be trained using only small subsets of the data. We focus our research in the generation of present KPs and we propose a new conditional GAN architecture for Keyphrase Generation that can be trained with a relatively small set of samples. The key component of our solution is the Discriminator: a model based on BERT that is able to distinguish between human and machine-generated Keyphrases leveraging on the language modelling information obtained from finetuning in a low-resource scenario. A Reinforcement Learning (RL) strategy is then used to train the Generator, with rewards evaluated by the Discriminator. This encourages the model to generate more accurate and relevant KPs.

Thanks to the characteristics of our architecture, we are able to use only a small subset of the available data, using less than 1% of them to train our system. Compared to all the previous approaches that needed to be fully trained on large set of training samples, our architecture greatly reduces required resources, while still providing competitive results in the generation of present KPs.

This work [89] has been presented at "SustaiNLP 2020, the First Workshop on Simple and Efficient Natural Language Processing"[1], part of the "2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)"[2].

4.2 Related work

4.2.1 Keyphrase Extraction

Extractive methods aim at identifying Keyphrases in the span of the source text. Most of the algorithms in this field adopt a two steps pipeline to extract KPs. First, given a document, a list of candidates phrases is selected using heuristic methods [158, 90]. Secondly, all candidates are scored against the document. The first step has a considerable impact on the ability of the whole model to correctly identify all KPs, so selecting a sufficiently high number of candidates is of utmost importance. The second step can be done either in a supervised or unsupervised manner [117, 165, 124]. The top-scoring candidates are returned as KPs. Two interesting strategies that differ from the common pipeline approach have been proposed by [151] and [183]. The first method employs two statistical language-based models to extract Keyphrases. The latter introduces a model based on joint layer recurrent neural network to extract Keyphrases from tweets.

4.2.2 Keyphrase Generation

Recently, research has focused on the introduction of methods of text generation to predict Keyphrases. Most of these approaches rely on Encoder-Decoder framework in which the source text is first mapped to an encoded representation, and then decoded to the target text, that is the Keyphrases to predict.

[115] proposed CopyRNN, a RNN-based generative model for KP Generation, which is an Encoder-Decoder model with copy mechanism. [27] proposed CorrRNN model which is a sequence-to-sequence architecture for Keyphrase Generation that captures the correlations among Keyphrases. TG-Net model was introduced by [29] for improving

[1]`https://sites.google.com/view/sustainlp2020`

[2]`https://2020.emnlp.org/`

automatic Keyphrase Generation using the information contained in the title of the document. [28] proposed an integrated approach for Keyphrase Generation which is a multitask learning framework that jointly learns an extractive model and a generative model.

Two recurrent generative based models, CatSeq and CatSeqD, were proposed by [180]. One of their main characteristics is the ability to determine the appropriate number of Keyphrases for each input document. CatSeq is based on an Encoder-Decoder mechanism, which is used to identify relevant components of the source text (abstracts) and generate KPs (sequence-to-concatenated sequences) [180, 25]. It employed the sequence-to-sequence framework combined with an attention mechanism and pointer softmax mechanisms in the Decoder. CatSeqD introduces the following techniques: orthogonal regularization, which prevents the model from predicting the same word after generating the constant KP separator; semantic coverage, which encodes again the decoded sequences and uses it as a representation of the target phrases. These representations are employed as further input during a self-supervised training phase with the aim of improve the semantic content of the predictions. [25] subsequently proposed a Reinforcement Learning approach with adaptive rewards to improve catSeq, CatSeqD, CorrRNN and TG-Net generative models, leading to a new version for each of them. These versions are called, respectively, catSeq-2RF1, catSeqD-2RF1, catSeqCorr-2RF1 and catSeqTG-2RF1.

Recently, [144] proposed a GAN model conditioned on scientific articles for KP Generation. The author uses a catSeq model to implement the Generator, conditioning it on abstracts of scientific articles. The Discriminator is based on a hierarchical attention mechanism consisting of two GRU layers. The two layers model the relationship between the document and each generated KP to assess whether the KP is synthetic or human in origin.

To the best of our knowledge, no attempts have been made of either extracting or generating KPs in a low-resources scenario, in which only a small amount of the available data samples is used during training. Our proposed architecture, based on a Discriminator that relies on a language model, requires less than 1% of the available training data to achieve good results.

4.3 The proposed approach

To generate present KPs in a low-resource scenario we propose an approach based on the GAN Framework that we call BeGan-KP. It mainly consists of three components: (1) a conditioned Generator model that produces a set of KPs, (2) a novel BERT Discriminator model that checks if the KPs are fake (generated) or real (human-curated), and (3) the Reinforcement Learning (RL) module that is involved in the training process of the system as a whole (see Figure 4.1).

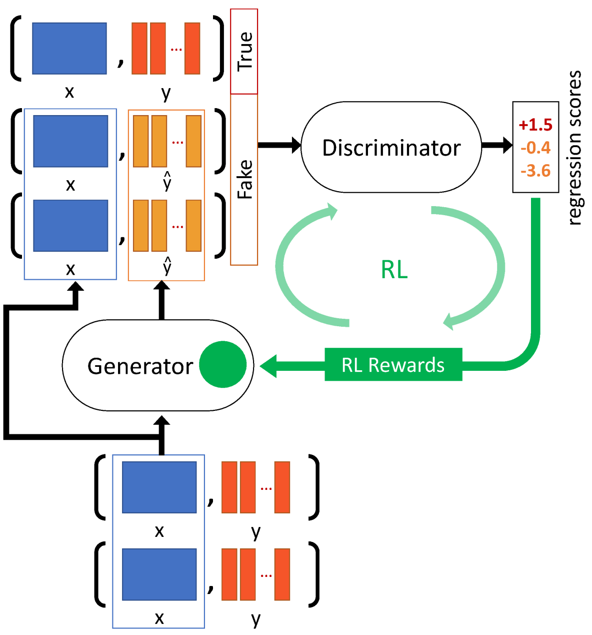


Figure 4.1: Schema of BeGan-KP.

4.3.1 Notations and problem definition

The samples available to train the system are pairs $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is a document and $\mathbf{y} = (\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^M)$ is the set of M Keyphrases (True KPs) associated to $\mathbf{x}$. Note that both $\mathbf{x}$ and $\mathbf{y}^i$ are sequences of words:

$$\mathbf{x} = x_1, x_2, \ldots, x_L$$
$$\mathbf{y}^i = y_1^i, y_2^i, \ldots, y_{K_i}^i$$

where L and K_i are the number of words of $\mathbf{x}$ and of its i-th KP respectively.

The Generator takes as input $\mathbf{x}$ and outputs $\hat{\mathbf{y}} = (\hat{\mathbf{y}}^1, \hat{\mathbf{y}}^2, \ldots, \hat{\mathbf{y}}^J)$, that is the set of the J predicted KPs for $\mathbf{x}$ (Fake KPs).

The objective is to generate Fake KPs that match exactly the True KPs: $\hat{\mathbf{y}} \equiv \mathbf{y}$.

4.3.2 Generator

The Generator G takes as input the document $\mathbf{x}$ and generates as output the sequence of $\hat{\mathbf{y}}$ (Fake KPs).

Following the work of [144] we use the catSeq model as Generator. It consists in an Encoder-Decoder model in which the Encoder is a bidirectional Gated Recurrent Unit (GRU) and the Decoder is a forward GRU. It is based on CopyRNN by [115].

We choose this component because it embeds some interesting features. It exploits the copying mechanism [56] to deal with long-tail words. These are words which are removed from the vocabulary due to their low frequency but are often topic-specific and therefore good candidates to be KPs. It also introduces the capability of predicting a variable number of Keyphrases for different documents. Furthermore it employs a beam-search strategy during the decoding step, meaning that at each time step the model decodes not just one word (greedy-search) but the top k most probable words. This allows generating more consistent sequences of words.

4.3.3 Discriminator

The Discriminator D receives as input the document $\mathbf{x}$ and a set of Keyphrases. These might be either the True KPs $\mathbf{y}$ or the Fake KPs $\hat{\mathbf{y}}$. Its task is to judge whether the KPs are True or Fake.

We introduce a novel Discriminator based on the language model BERT [39]. Differently from the previous literature, our idea is to exploit the strength of the language model characteristics to classify the quality of the input pair $(\mathbf{x}, \mathbf{y})$. This judgement is given as a *regression score*, which is lower for Fake KPs and higher for True KPs. In this way the regression score can be easily interpreted as the *reward* in the Reinforcement Learning module, giving to the system an inherent clarity. Moreover, different BERT-based models and reward configurations have been tested at an early stage, and the choice of a regression model provided the best results.

The language modelling component is able to achieve a better comprehension of the relationship of the two input sequences, while the robust pretraining allows us to use it efficiently even in a low-resource scenario.

In particular, the Discriminator model consists of four subcomponents (see Figure 4.2) :

- **Input preparation**. The input pairs $(\mathbf{x}, \mathbf{y})$ are tokenized and the tokens are concatenated to be compliant with the general pattern:

  ```
  [CLS]<x>[SEP]<y1><;>...<;><yn>[SEP]
  ```

  where `[CLS]` and `[SEP]` are special tokens which signal the start of the input and the end of text sequence respectively, `<x>` is the sequence of tokens for the document $\mathbf{x}$, `<yi>` is the sequence of tokens for the KP $\mathbf{y}^i$. Different KPs are separated by semicolon `<;>`. Note that the `[SEP]` token in the center is used to split the input sequence into document and KPs.

- **BERT modelling**. The input sequence is processed by a pretrained BERT model. It performs a word embedding of all the tokens and then passes them through 12 Encoder blocks. As it is basically a positional language model, it returns the last hidden states for each of the initial tokens.

- **Output aggregation**. Each of the outputs of the preceding step can be seen as an highly abstract embedding of the corresponding token. We aggregate the output of all the hidden states and evaluate their mean to obtain an embedding for the whole input sequence $E = E(\mathbf{x}, \mathbf{y})$. Note that in this way E is not generated using only the output obtained from the `[CLS]` token, but making use of the representations of all the tokens instead. Based on our preliminary experiments as well as literature references [39], this value is considered to represent a better summary of the semantic content of the input.

- **Regression.** E is processed by the regression layer, a fully connected linear classifier, and a regression score is calculated. This is trained to be high for True KPs (human-curated) and low for Fake KPs (artificially generated), and is used as the reward in Reinforcement Learning.

The overall output of the Discriminator is therefore a regression score relative to the combination of input document and the related KPs.

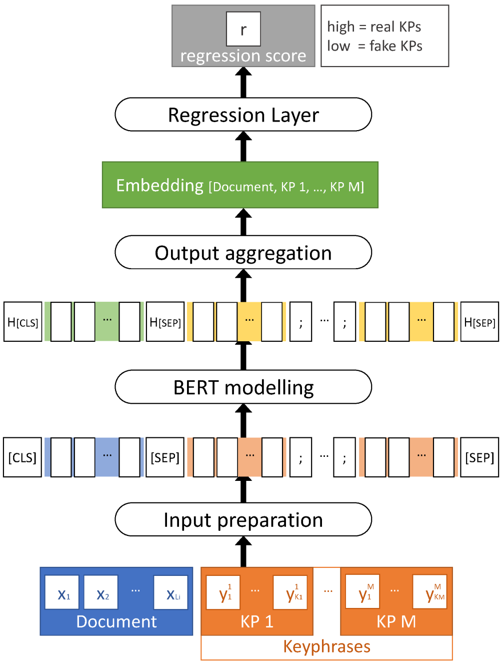


Figure 4.2: Schema of the Discriminator and its four processing phases.

4.3.4 Reinforcement Learning

To overcome the problem of non differentiability of the output layer of our architecture we extend the Reinforcement Learning strategy proposed by [179] in the domain of KP Generation. In particular, we consider the Generator G as an agent whose action a at time step t is to generate a *word* y_t, which is part of the set of predicted KPs $\hat{\mathbf{y}}$ for the document $\mathbf{x}$. In this scenario the Discriminator D plays the role of the environment that evaluates the actions made by G and gives back a reward. Agent G acts following a policy

$$\pi = \pi(y_t | s_t, \mathbf{x}, \theta) \tag{4.1}$$

that is a function representing the probability distribution of y_t given the current state $s_t = (y_1, \dots, y_{t-1})$, the sequence of words so far generated. The policy function is differentiable with respect to the set of parameters θ of G. Once the agent G generates the predictions, the environment D gives back a reward

$$r_t = f(y_1, \dots, y_t|\mathbf{x}) \tag{4.2}$$

and moves to the state s_{t+1}. The reward is a quality measure of the action made by the agent G, and depends on the words generated up to the current time step (subset of $\hat{\mathbf{y}}$) given the input document $\mathbf{x}$. The agent G acts to maximize the reward, that is to maximize a differentiable optimization function $J(\theta)$ that gives a measure of the performance of G. According to the policy gradient theorem and the REINFORCE algorithm [164] the gradient of $J(\theta)$ can be expressed as:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[\sum_t r_t \nabla log(\pi(y_t|s_t, \mathbf{x}, \theta)) \right] \tag{4.3}$$

where the sum extends to all the time steps needed to generate the complete sequence $\mathbf{y}$.

The expectation $\mathbb{E}_\pi$ in Equation 4.3 can be approximated using a complete sequence $\hat{\mathbf{y}}$. In order to calculate the cumulative rewards of Equation 4.2 we use the regression score of a complete sequence of generated KPs: $r = D(\hat{\mathbf{y}})$.

Considering that maximizing the optimization function $J(\theta)$ is equivalent to minimizing its additive inverse, we can define the loss function of G as $L(\theta) = -J(\theta)$ and an estimator of its gradient as:

$$\nabla L(\theta) \approx -\sum_t (r - b) \nabla log(\pi(y_t|s_t, \mathbf{x}, \theta)) \tag{4.4}$$

where the regularization term b is introduced to reduce the variance of the above $\nabla L(\theta)$ estimator. It is essentially the cumulative reward $r = D(\bar{\mathbf{y}})$ where $\bar{\mathbf{y}}$ is a greedy decoded predicted sequence. The aim is to promote rewards that show effective improvements over greedy sequences [132].

4.3.5 GAN training

The first step is to train a first version G_0 of the Generator using the Maximum Likelihood Estimation (MLE). G_0 is then used to generate the Fake KPs $\hat{\mathbf{y}}$. $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}$ are used to train the first version of the Discriminator D_0 with Mean Squared Error (MSE) loss:

$$MSE(x, \hat{x}) = \frac{1}{N} \sum_{i=0}^{N} (x_i - \hat{x}_i)^2 \tag{4.5}$$

Starting from Generator G_1 training is performed using RL, so the loss is given by $L(\theta)$ as shown in Section 4.3.4. The training of the Discriminator remains the same. After each training iteration (G_j, D_j), predictions are tested to evaluate the scores $F1@M$ and $F1@5$.

4.4 Experimental results

4.4.1 Datasets and metrics

We compare our solution with state-of-the-art approaches on five datasets which are commonly used in literature:

KP20k [115] It consists of 567,830 titles and abstracts from computer science papers. The usual split is performed using 20,000 samples for testing, another 20,000 for validation, while the remaining 527,830 samples are used for training. In our low-resource scenario we only use 2,000 out of the >500,000 training samples.

Inspec [72] The complete dataset is composed of 2,000 abstracts from Computers and Control, and Information Technology disciplines. A subset of 500 samples is used for testing.

Krapivin [84] The original released dataset is composed by 500 complete articles belonging to the domain of computer science. For KP Generation purposes only titles and abstract are used. The first 400 samples in alphabetical order are selected for testing.

NUS [124] A set of 211 scientific publications, all used for testing.

Semeval2010 [78] 288 conference and workshop papers from the ACL Computer Library. 100 used for testing.

A brief report of main statistics of the test sets used is given in Table 4.1.

	KP20k		**Inspec**		**Krapivin**		**NUS**		**Semeval2010**	
	#	%	#	%	#	%	#	%	#	%
Present KPs	66,267	62.91	3,602	73.59	1,297	55.57	1,191	52.26	612	42.41
Absent KPs	39,076	37.09	1,293	26.41	1,037	44.43	1,088	47.74	831	57.59
Total KPs	105,343	100.00	4,895	100.00	2,334	100.00	2,279	100.00	1,443	100.00
Test samples	20,000		500		400		211		100	

Table 4.1: Statistics on test samples for the five datasets.

Model	**KP20k**		**Inspec**		**Krapivin**		**NUS**		**Semeval2010**	
	F1@M	*F1@5*	*F1@M*	*F1@5*	*F1@M*	*F1@5*	*F1@M*	*F1@5*	*F1@M*	*F1@5*
catSeqD [180]	-	**0.348**	-	0.276	-	**0.325**	-	0.374	-	**0.327**
catSeqCorr-2RF1 [25]	0.382	0.308	0.291	0.240	0.369	0.286	0.414	0.349	0.322	0.278
catSeqTG-2RF1 [25]	**0.386**	0.321	0.301	0.253	0.369	0.300	**0.433**	**0.375**	**0.329**	0.287
GAN [144]	0.381	0.300	0.297	0.248	**0.370**	0.286	0.430	0.368	-	-
BeGan-KP (our approach)	0.318	0.309	**0.383**	**0.356**	0.332	0.317	0.388	0.366	**0.329**	0.319

Table 4.2: Results of present keyphrases for five datasets. Our approach is BeGan-KP.

All datasets are preprocessed following [25]: duplicate papers are removed from KP20K, and for each document the list of KPs is sorted in order of appearance in the document. Digits in the input texts are replaced with the special token `<digit>`.

Results are evaluated using $F1$ score. In particular $F1@5$ and $F1@M$ are employed: the first is calculated considering only the top 5 high scoring KPs, the second is computed taking into account all the predictions.

All sample documents are annotated with human curated KPs. Of the above mentioned datasets, only KP20k is used for training; all the others are used only for testing and evaluation. Note that the strength of the language model of our Discriminator allows us to use only a small subset of the data samples during training: the whole architecture has been trained with a subset of 2,000 samples instead of the >500,000 used by the other state-of-the-art approaches.

4.4.2 Implementation details

The initial MLE model G_0 is trained with a batch size of 12 and Adam optimizer [80]; during RL training, batch size is 32. The Discriminator is trained with a batch size of 3 and AdamW optimizer [103]. The pretrained BERT model is the base uncased version, with 12 layers, 12 attention heads, and hidden size of 768. The maximum input length after tokenization is fixed to 384 tokens. We use the implementation provided in the python library transformers by huggingface [166][3].

Training and experiments have been executed on a PC with a GeForce RTX 2080 GPU, 11GB.

4.4.3 Results

Our proposed solution BeGan-KP, trained on 2,000 samples, has then been compared with the following state-of-the-art approaches: catSeqD [180]; catSeqCorr-2RF1 and catSeqTG-2RF1 [25], and GAN [144]. The results of our tests are shown in Table 4.2.

First, we can note that BeGan-KP achieves results competitive with the best performing techniques, even using a limited set of samples (all the other approaches were trained on the whole KP20k).

Looking at the results in detail, we obtain by far the best performance for Inspec both in $F1@5$ and $F1@M$.

Our approach has other good results in $F1@5$ metrics, specifically in Krapivin and Semeval2010 where our values are only slightly lower than the best. Since $F1@5$ is calculated considering the 5 predictions with the highest score, we can say that our model is capable of producing high quality Keyphrases reliably, and of outperforming or at least matching other best-performing models in this specific task. This confirms the strength and consistency of our architecture.

In addition, we obtain the best $F1@M$ score for Semeval2010. Note that Semeval2010 is a demanding test dataset as it is the smallest of the five, and the gross amount of KPs to predict is the lowest (612 present KPs out of a total of 1,443), leading to a great variance in the output.

Finally, consider that in Equation 4.3 the expectation of the policy function is evaluated using only one complete sequence $\hat{\mathbf{y}}$, inducing a high variance in the ∇J. This is a general issue of Reinforcement Learning applied to GANs for text generation and generally leads to unstable training process and slow convergence [179]. Thanks to the

[3]`https://github.com/huggingface/transformers`

capability of the language model embedded in our architecture, in our experiments the training process shows a quick convergence in terms of number of training iterations. In fact, the reported results have been achieved at the second iteration (G_2 generator).

4.5 Considerations about efficiency

Natural Language Understanding and Processing are demanding research topics. Texts have an inherent sequential aspect; in addition, they embed an internal structure which links some parts to other. As a comparison, images can be represented by flat grids of values and can be elaborated in any order. Dealing with texts implies to understand their sequential and structural aspects: the former are related to the syntactic content, the latter to the semantics. The task of Keyphrase Generation is particularly difficult because involves both the aspects: generated KPs are to be syntactically correct (e.g.: not containing verbs or punctuation), and have to be a semantic representation of the input.

To tackle the problem a complex architecture is proposed: it is based on a Generative Adversarial Networks (GAN) architecture which consists of two main components: the Generator and the Discriminator. Generator creates synthetic keyphrases and Discriminator evaluates how good they are. The system needs to be trained with Reinforcement Learning to overcome problems of non-differentiability of the output layer due to the discrete nature of the words. Nonetheless, some key aspects of the architecture are chosen with the aim to increase efficiency.

- The most relevant aspect is the use of a pretrained BERT model for the Discriminator. It is fine tuned with a thin regression module responsible of generating the classification score for the examinated keyphrase. Discriminator leverages the strong language model of the pretrained BERT to learn the structure of the input text and its relations with the set of keyphrases. As a consequence, the amount of data needed to train Discriminator is very low: with only 2.000 samples it learns to discriminate between real and fake keyphrases. This is a crucial aspect because the reward given by the Discriminator in the form of a regression score is the key to make the Generator to create more realistic fake keyphrases.

- The Generator catSeq is an encoder-decoder model highly optimized for keyphrase generation. One of its features is the Copy mechanism, introduced to recover the *long-tail* words. Long-tails are words contained in the corpus of the training texts, but with low frequency; they are removed from the vocabulary used by the model, with the aim to improve computational efficiency: the less the words in the vocabulary, the less the trainable parameters in the decoder implementation. Copy mechanism is basically an attention mechanism which evaluates attention scores for *all* the words of the input text, and recovers the long-tails with high attention scores. These are often very topic-specific and so good candidates to be part of keyphrases. Copy mechanism has the effect of reducing the dimensions of the vocabulary, which is turn means a reduction of the number of trainable parameters.

- Training the architecture is an iterative process. A generator and a discriminator are sequentially trained: the discriminator learns to split the real keyphrases from the generated ones; the generator in turn learns to maximize the reward of the discriminator by predicting more realistic keyphrases. The process is repeated until an optimum in the performance metrics is reached. The whole process is a time consuming one, and requires a great computational effort in terms of number of FLOPs. The use of early stopping strategy on both the training phases greatly reduced the times and the number of computations.

Despite the demanding task and the complex architecture implemented to address it, the use of methodologies for efficient deep learning enabled the system to obtain results comparable with state-of-the-art models, using only a fraction of training data.

5 Atomistic Graph Neural Networks for metals: Application to bcc iron

5.1 Introduction and related work

Recently, Graph Neural Networks (GNNs) have become one of the most active research fields in Artificial Intelligence [188]. GNNs are a class of Deep Learning methods introduced to analyze data which display a graph structure. Graphs represent the topology of a great variety of data structures in which objects (nodes) are connected with each other by some kind of relation (edges). Due to the very general nature of graphs, applications of GNNs are found in very different contexts, such as computer vision [160, 128, 70, 170], natural language processing [174, 71, 16, 107], social sciences [44, 147], and natural sciences including biology [47], particle physics [141] and astrophysics [30].

The topology of a graph can also reflect that of atomistic crystal structures: indeed, a graph can be generated by connecting each atom (nodes) with its neighbors (edges), within a specified cutoff radius [121]. The Message Passing Neural Networks framework (MPNN) [52] has been introduced as a common GNN paradigm for atomistic structures in quantum chemistry applications.

Within an atomistic GNN, the atoms and their connections are associated with numerical lists of "features", also named *embedding* vectors. Features are updated by the Message Passing framework, which is a two-step process. In the first step, each atom receives a message that is an aggregate of its neighbour's embeddings. In the second step, an updated embedding of the atom is evaluated, by means of a function that depends on the message and on the current atomic embedding. By iterating this scheme n times, each atom will receive messages from atoms that are distant up to n connections, thus accounting for long-range interactions.

A GNN model for atomistic graphs is therefore determined: (i) by the nature and the size of the embeddings, which convey the informative content of the specific atomistic system and (ii) by the operations it executes on them, i.e. the procedure used to aggregate and update embeddings. Once the aforementioned characteristics are defined, the model can be trained to predict the system potential energy surface (PES).

A number of GNN models have been proposed in the recent years to model atomistic systems. Most of them were first introduced in molecular research and further applied to crystals. Deep Tensor Neural Networks (DTNN) [139] and PhysNet [152] aggregate the atomic embeddings by means of *filters* that ensure that the resulting message changes smoothly with respect to small changes of the interatomic distances. The main difference between DTNN and PhysNet lies in how distances are represented and how messages are aggregated. Crystal Graph Convolutional Neural Networks (CGCNN) [169] were explicitly developed to deal with materials displaying a crystal structure, such as metals. Unlike DTNN and PhysNet, CGCNN considers both atomic (node) and edge embeddings; however, distances are not regularized with continuous functions: the range of the distances is partitioned in ten equally spaced segments, and interatomic distances are encoded within a single vector in which all components are zero but the one associated with the matching segment. Thus, this model lacks the ability to smoothly change the embeddings with respect to small displacements of the atomic positions. SchNet [137] is based on DTNN and introduces continuous filter convolutions: distances are used as input of a neural layer that generates a continuous mapping to an embedding space. In an updated version [138], periodic boundary conditions (PBCs) have been introduced, and the model has been applied to the prediction of formation energies of bulk crystals. Also inspired by SchNet and sharing its overall architecture, the "Neural message passing with edge updates" [77] uses both node and edge embeddings in the form of a concatenation of the two connected atoms embeddings. This makes edge embeddings directional as they depend on the order of the concatenated elements. MatErials Graph Network (MEGNet) [26] leverages a similar scheme, by incorporating both directional edge and node updates, while also introducing a global state vector which stores the molecule/crystal-level or state attributes, e.g. the temperature of the system. Updates of atoms, bonds and global state vector are performed in a sequence. All these approaches employ filters that rely only on the distance between pairs of atoms to aggregate and update the atomic embeddings.

It is well-established that classical, empirical interatomic potentials [37] that rely on pairwise interactions often fail to reproduce structural changes [93] and some crucial properties of dislocations in metals [108]. In the case of phase transitions, the addition of directionality, i.e. angular dependence of the interatomic potential, as well as second nearest-neighbor interactions, has led to the improved qualitative reproduction of quantum-mechanical PES [93].

Within the context of GNNs, there is a remarkable shortage of approaches that rely also on the angle between edges connecting atomic pairs. Embeddings of edges connecting triplets of atoms convey the angular information, and once they are updated via the message passing scheme, they can be used to update the atomic embeddings. With this aim, DimeNet [82] also leverages the Directional Message (hence the name) by considering the direction of the pairwise connections and by introducing the angle between two edges connected within atomic triplets. DimeNet employs a continuous filter convolution by expanding both distances and angles in a Bessel-Fourier basis. However, to date, DimeNet has been applied merely to isolated molecules and has not been investigated to model crystals such as metals.

Although GNNs have been scarcely explored in the context of interatomic potentials for metals, they introduce a number of advantages with respect to other ML methods [121]. First, interactions among neighbouring atoms are straightforwardly modeled as pair-wise connections. Previous ML approaches need to introduce specific geometrical descriptors of the environment around atoms (within a cut-off radius), such as atom-centered symmetry functions in the Neural Networks Potentials [17], or bispectrum components and then smooth overlap of atomic positions (SOAP) [13] in Gaussian Approximation Potentials (GAP) [14]. Second, iterating the process makes the model able to consider the contributions of distant atoms, so as to mimic the influence of long-range interactions beyond the cut-off distance that limits pairwise interactions. This can be easily achieved by stacking message-passing layers in the network. Previous ML approaches either lack these long-range contributions or account for them by adding extra long-range terms to the total energy, e.g. for electrostatic interactions [18]. Third, the GNN approach guarantees scalability of the system, as the pair-wise nature of the connections means that complex clusters of atoms can be modeled by simply increasing the number of iterations, at a limited computational cost. Finally, since the approach is only dependent on the relative positions of the atoms which determine the connections inside the cut-off radius, it is also invariant with respect to isometric transformations, i.e. reflections, translations, rotations, and combinations of those, and to permutation of atoms.

Here, we use GNNs to explore their ability to reproduce with quantum-accuracy the potential energy surface (PES) of metals, by taking as a reference the challenging and technologically crucial example of ferromagnetic body-centered-cubic (BCC) iron. We consider SchNet as a prototypical GNN framework that is based on the distance of atomic pairs, and we consider DimeNet to assess the performance of a GNN scheme that also includes angular (three-body) interactions. To this purpose, we have implemented periodic boundary conditions (PBCs) and made the new DimeNet implementation that includes PBCs available at `https://github.com/AilabUdineGit/GNN_atomistics/`. In order to machine-learn the GNN interatomic potential, we use an existing database [40] that was previously trained to develop a Gaussian Approximation Potential (GAP) [41].

Currently this work is being submitted to “npj - Computational Materials”[1], one of the leading journals in the research about materials, and member of the “Nature” series of scientific publications (preprint: [31]).

5.2 The proposed approach

5.2.1 Graph Neural Networks and Message Passing

A graph [81] is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $i \in \mathcal{V}$ are the N nodes and $(i, j) \in \mathcal{E}$ are the edges. The connections among the nodes of a graph can be stored in an adjacency matrix $A \in \mathbb{R}^{N \times N}$ containing the pairs $(i, j) \in \mathcal{E}$. At both nodes and edges, vectors of features (or embeddings) are defined as $\mathbf{x}_i \in \mathbb{R}^F$ and $\mathbf{e}_{ij} \in \mathbb{R}^D$, respectively, where F, D are model specific parameters. In the message passing with node update, node

[1]`https://www.nature.com/npjcompumats/`

embeddings are updated iteratively, with each iteration executed in the message passing layers l as follows:

$$\mathbf{x}_i^{(l+1)} = \gamma(\mathbf{x}_i^{(l)}, \sum_{j \in \mathcal{N}_i} \mu(\mathbf{e}_{ij}^{(l)}, \mathbf{x}_j^{(l)})) \tag{5.1}$$

where $\mathcal{N}_i$ is the set of the nodes connected to node i, μ is a differentiable function of the nodal and edge embeddings, the sum aggregates the contributions of atoms j, and γ is a differentiable function which evaluates the update of node embedding.

In the message passing with edge update [76], edge embeddings are updated by following a similar scheme:

$$\mathbf{e}_{ij}^{(l+1)} = \kappa(\mathbf{e}_{ij}^{(l)}, \sum_{k \in \mathcal{N}_j \setminus \{i\}} \nu(\mathbf{x}_j^{(l)}, \mathbf{e}_{jk}^{(l)}, \mathbf{x}_k^{(l)})) \ . \tag{5.2}$$

with the same conventions as the previous case, κ and ν being differentiable functions of the nodal and edge embeddings, analogously to γ and μ. Note that edges connected to (i, j) are the edges (j, k) linking node j and node $k \neq i$, hence the index of the summation.

At the next iteration, the message is evaluated by the layer $l + 1$ by aggregating embeddings $\mathbf{x}_i^{(l+1)}$ ($\mathbf{e}_{ij}^{(l+1)}$) from the neighbours, which in turn have received a message from their own neighbours: stacking together L layers means that the final update is performed by using messages coming from a distance up to L neighbors away, see fig. 5.1.

Once iteratively updated via the message passing, embeddings are elaborated by a readout function

$$y = f(\{\mathbf{x}_i^{(L)}, \mathbf{e}_{ij}^{(L)}\}) \tag{5.3}$$

which performs a further aggregation of all the embeddings and outputs the prediction $y \in \mathbb{R}$ of the network.

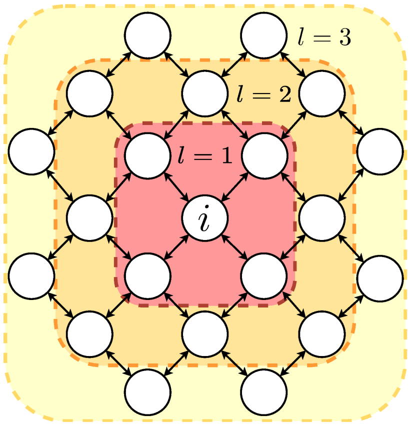


Figure 5.1: Message Passing with node update. Atomic environment as seen by node i: layer $l = 1$ aggregates messages from connected neighbours (red area); layer $l = 2$ acts the same but connected neighbours have been already updated by messages from their neighbours at previous layer, so signals received by atom i are now from a distance of up to two edges (orange area). A sequence of L layers means messages coming from nodes at a distance of up to L edges. A similar scheme works for message passing with edge update.

Within the context of crystalline materials, it is straightforward to consider the nodes of a graph as the atoms and to connect by edges the pairs of atoms that lie within a specified interaction radius. Let $\mathbf{r}_i \in \mathbb{R}^3$ be the coordinates of the atom i. Then, the graph is defined by connecting all the atoms j that are inside the cutoff radius $r_{cut} > ||\mathbf{r}_i - \mathbf{r}_j||$. Atomic embeddings $\mathbf{x}_i$ are vectors of learnable numerical features. They are randomly initialized, and atoms with the same set of relevant atomic properties, such as atomic number Z, have the same initial embeddings. Edge embeddings $\mathbf{e}_{ij}$ are similar, with properties related to pairs of connected atoms, such as the interatomic distance d_{ij}. The message and update functions μ, γ (5.1, 5.2) are neural layers which add learnable weights and define the form of the convolutional filter and of the update procedure. Hence, the prediction of the potential energy $E \in \mathbb{R}$ of a crystal as a function of the atomic coordinates is a regression task performed on such a graph (5.3).

5.2.2 Network models

We use two recent models of Graph Neural Networks based on the Message Passing framework: SchNet [138] and DimeNet [82]. There are two main differences between them. The first is related to the embeddings: SchNet relies on atomic embeddings, while DimeNet also uses edge embeddings in the form of pairs of atom embeddings to account for the directionality of the message passing. The second difference is the learned convolutional filters used to aggregate embeddings: while SchNet employs a filter that accounts only for the distance between pairs of atoms, DimeNet considers also the angles formed by pairs of edges, or triplets of atoms. The general scheme of both the models is shown in fig. 5.2. At an high level of abstraction, they can be described in terms of block diagrams, with each block representing a set of specific neural layers that performs some operations on input data and generates output data. Blocks with the same name in both models perform similar general operations.

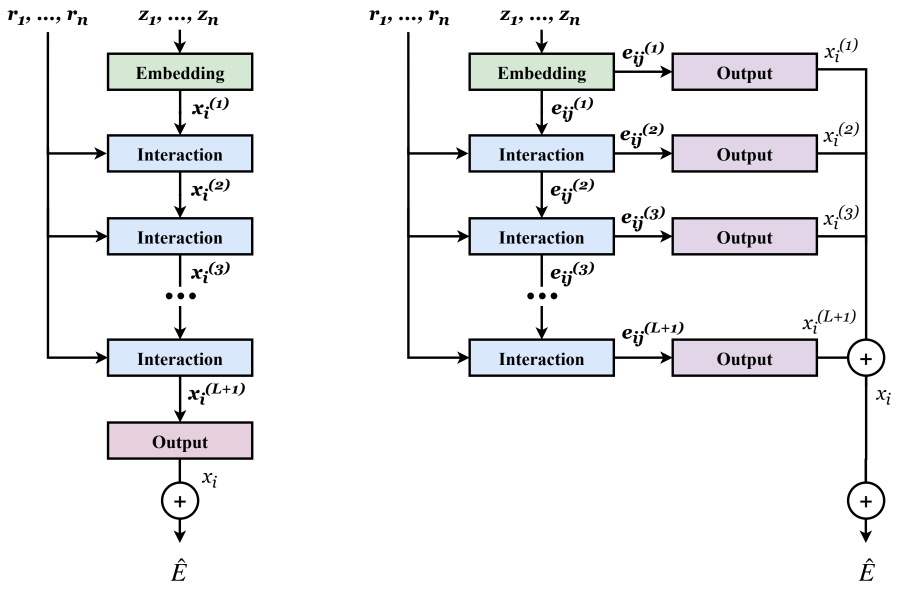


Figure 5.2: Block diagram for SchNet (left) and DimeNet (right). Outputs generated by each block are shown near the arrows. In both models the starting point is the Embedding block that maps atom or edge features in a vector space, generating embeddings. For SchNet, the Interaction blocks are in a sequence, the output of one being the input of the next; the final Output block evaluates the total energy $\hat{E}$. In DimeNet the output of each block is both passed sequentially to the next and further elaborated by the Output block, and then summed to obtain energy E.

Filters and physical representation of the atomic environment.

To take into account the physical environment surrounding each atom, the convolutional filter assigns weights to the embeddings received by the neighbours (see below the description of Embedding and Interaction blocks). Filter weights are learned during training and have to change smoothly with respect to small atomic shifts. Therefore, distances and angles are *expanded*, that is represented as feature vectors whose components are sets of continuous basis functions. In SchNet the filter depends only on the interatomic distance d, expanded by a set of radial, Gaussian (G) basis functions:

$$\phi_k^G(d) = \exp\left(-\frac{(d-\mu_k)^2}{2\sigma^2}\right) \tag{5.4}$$

with μ_k equally spaced in the interval $[0, r_{cut}]$, and σ representing the scale of the distances. Hyperparameters k and σ define the granularity of the representation, and determine the precision of the filter. The spacing r_{cut}/k and the scale σ are both set to 0.1 Å in the original paper [138]; in order to improve the precision to better compare with DimeNet we set them to 0.04 Å.

DimeNet introduces two different filters: one radial depending only on distances, used to weight the embeddings received by atoms; and one radial-angular which takes into account also the angles to weight the embeddings passed to the edges. Both distances and angles are expanded in a 2D Bessel-Fourier basis which are the solutions of the related time-independent Schrödinger equation, and represent the electron density of the system inside the cutoff radius. For the first only radial filter, distances d are expanded in a feature vector whose components are given by the Radial Basis Functions (RBF):

$$\phi_k^{RBF}(d) = \sqrt{\frac{2}{r_{cut}}} \frac{\sin\left(\frac{k\pi}{r_{cut}}d\right)}{d} \, . \tag{5.5}$$

The second filter depends on the distance d and the angle θ. The components of the bidimensional feature vectors are given in terms of the Spherical Basis Functions (SBF):

$$\phi_{l,k}^{SBF}(d,\theta) = \sqrt{\frac{2}{r_{cut}^3 j_{l+1}^2\left(z_{lk}\right)}} j_l\left(\frac{z_{lk}}{r_{cut}}d\right) Y_l^0(\theta) \tag{5.6}$$

where j_l are the spherical Bessel functions of the first kind and Y_l^m are the spherical harmonics; z_{lk} is the k-th root of the l-order Bessel function. Settings for the non learnable parameters are as per the original paper [82], namely: for eq. 5.5 $k \in [1, \dots, 6]$ while in eq. 5.6 $k \in [1, \dots, 6]$, $l \in [0, \dots, 5]$. To avoid the discontinuity given by the boundary condition $\phi(d) = 0$ for $d > r_{cut}$, functions 5.5 and 5.6 are multiplied by a smoothing polynomial $u(d) \sim \mathcal{O}(d^8)$: a step function with a root of multiplicity 3 at $d = r_{cut}$.

For both SchNet and DimeNet the expanded representations are passed through dense neural layers (see below) which add the learnable weights. The filter is therefore a mapping of the physical representation of angles and distances to a vector space with dimensions matching the ones of the embeddings to weight. The general aspect of the filters and an intuition of how they work are shown in fig. 5.3.

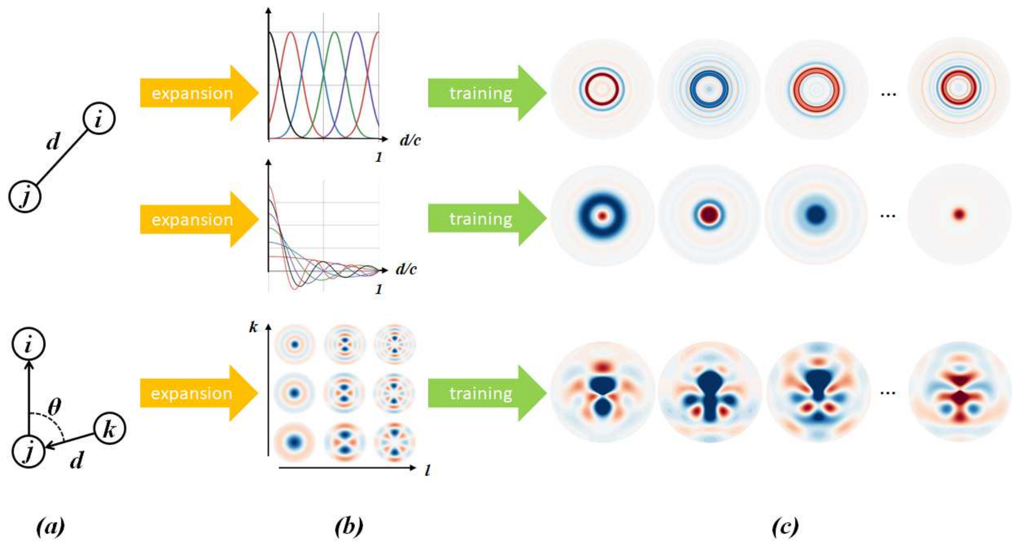


Figure 5.3: Filters and physical representation of the atomic environment. (a) Starting from the positions of atoms, distances between pairs (top, SchNet and DimeNet) and distances and angles between triplets (bottom, DimeNet) are evaluated. (b) Distances d are expanded in a Gaussain basis of functions (top, SchNet, eq. 5.4) or in Radial Bessel basis (middle, DimeNet, eq. 5.5), while distances d and angles θ for triplets are expanded in a 2D Bessel-Fourier basis (bottom, DimeNet, 5.6). (c) The convolutional filters: expansions are passed through dense layers whose weights are optimized during training. Learned weights are the convolutional filters. The first three and the last component are extracted and shown for all the cases.

Dense layers.

Dense layer is the very basic element of a neural network. Given an input $\mathbf{x} \in \mathbb{R}^k$ it is defined as

$$\mathbf{y} = \sigma(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) \tag{5.7}$$

where $\mathbf{W} \in \mathbb{R}^{m \times k}$, $\mathbf{b} \in \mathbb{R}^m$ are the learnable weights and bias, $\cdot$ is the matrix multiplication operator and σ is the *activation*, i.e. a differentiable nonlinear function. Activation is the shifted softplus for SchNet: $ssp(x) = \ln(0.5 \cdot e^x + 0.5)$, and the self-gated Swish for DimeNet: $sgs(x) = x \cdot \text{sigmoid}(x)$. In terms of vector algebra a dense layer projects the input vector $\mathbf{x} \in \mathbb{R}^k$ into a vector space $\mathbb{R}^m$ with $m \neq k$ in general, and then applies the function σ element-wise.

Embedding block.

For SchNet, atom embeddings are defined as vectors $\mathbf{x}_i \in \mathbb{R}^F$; initial values $\mathbf{x}_i^{(0)}$ are randomly initialized. For DimeNet, similarly defined atom embeddings are concatenated in pairs to generate an initial edge embedding $\mathbf{e}_{ji}^{(0)} = (\mathbf{x}_j^{(0)} || \mathbf{x}_i^{(0)} || \phi_k^{RBF}(d_{ji}))$. Note that this definition guarantees the directionality, as in general $\mathbf{e}_{ji} \neq \mathbf{e}_{ij}$. Once initialized, embeddings are passed through dense layers.

Interaction block.

Message passing paradigm is implemented in Interaction blocks. Multiple Interaction blocks are generally stacked together. Each of them performs a convolution by aggregating embeddings from the directly connected entities, and then updating them. The output of one block is passed as the input to the next.

Let l be the generic Interaction block. In SchNet, the embedding $\mathbf{x}_j^{(l)}$ received by atom i from neighbour $j \in \mathcal{N}_i$ is first weighted by the gaussian radial filter depending on $\phi^G(d)$ (eq. 5.4). Then embeddings are aggregated and the resulting embedding is summed to $\mathbf{x}_i^{(l)}$ and passed through a dense layer to update it to $\mathbf{x}_i^{(l+1)}$ (eq. 5.1). In DimeNet, the edge (j, i) receives message embeddings $\mathbf{e}_{kj}^{(l)}$ from edges (k, j) that are first weighted by means of the radial filter based on $\phi^{RBF}(d)$ (eq. 5.5) with $d = d_{ji}$, and then by the radial-angular filter based on the Bessel-Fourier basis $\phi^{SBF}(d, \theta)$ (eq. 5.6), where θ is the angle formed by (j, i) and (k, j) and $d = d_{kj}$. Again, exchanged messages are aggregated, summed over k to the embedding $\mathbf{e}_{ji}^{(l)}$ relative to edge (j, i) and then passed through the dense layers to obtain the updated $\mathbf{e}_{ji}^{(l+1)}$ edge embedding (eq. 5.2). For an intuition of how the filters are applied see fig. 5.4. In DimeNet, updated messages are given as input to the next interaction block *and* to the related output block, see below.

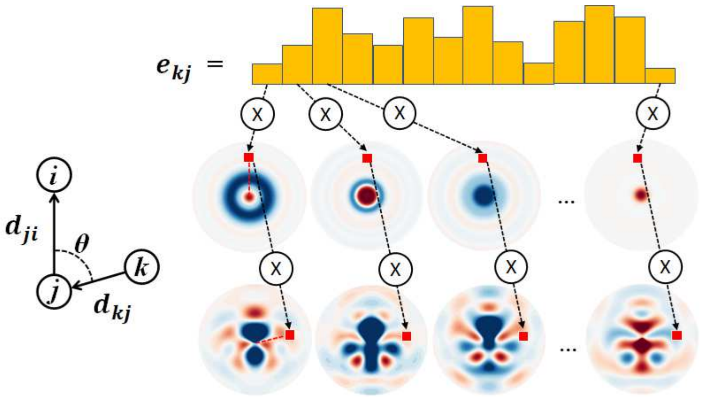


Figure 5.4: Schematic of the application of the filters for DimeNet. Features of the edge embedding e_{kj} are first multiplied element-wise with the value at the point d_{ji} of the components of the radial filter. Then another element-wise multiplication is performed with the value at the point (d_{kj}, θ) of the components of the radial-angular filter. An analogous scheme works for SchNet, limited to the purely radial filter.

Output block

Output blocks are responsible for lowering the dimensions of the atom embeddings, reducing them to scalars.

SchNet has one only Output block at the end of the Interaction blocks stack; it is a sequence of dense layers whose task is to reduce the embedding to a scalar $\mathbf{x}_i^{(L+1)} \rightarrow x_i^{(L+1)}$, to be interpreted as the atom-wise contribution to the total potential energy. The final prediction is evaluated as the sum of atom-wise contributions $\hat{E} = \sum_i x_i^{(L+1)}$.

DimeNet performs another convolution here, resulting in the update of the atomic embeddings. Embeddings $\mathbf{e}_{ij}^{(l+1)}$ from the related interaction block l (and of the embedding block, $l = 0$) are further convoluted by means of a radial filter based on $\phi^{RBF}(d)$: $\mathbf{e}_{ij}^{(l+1)} \rightarrow \tilde{\mathbf{e}}_{ij}^{(l+1)}$. The update of the embedding of atom i is then evaluated as $\mathbf{x}_i^{(l+1)} = \sum_j \tilde{\mathbf{e}}_{ij}^{(l+1)}$. Multiple dense layers are applied to reduce dimensions to 1: $\mathbf{x}_i^{(l+1)} \rightarrow x_i^{(l+1)}$, to be intended as the per-atom contribution of the level l blocks to the output of the model. Finally they are summed atom-wise and level-wise to evaluate the final prediction of the network $\hat{E} = \sum_l \sum_i x_i^{(l+1)}$.

5.3 Experimental results

5.3.1 Implementation and training details

To model bulk crystal structures, the simulated atomic cluster must be embedded in an effectively infinite medium. This is achieved by using periodic boundary conditions (PBCs), which are already implemented in SchNet. Here, PBCs have been implemented also for DimeNet. The training strategy is the same for both SchNet and DimeNet. All data used for the training are from a large, existing, highly-converged DFT database [40] of bcc ferromagnetic iron that includes both pristine configurations and configurations with defects such as free surfaces, vacancies and interstitials (see Database section for details). A GAP potential that reproduces accurately DFT vibrational and thermodynamic properties [41] is also trained, and employed as a baseline in the comparison of the GNN models.

The training dataset is built as a subset of 80% of the database; samples are randomly shuffled to avoid bias. The remaining 20% of the samples is used to test the trained model; samples are not shuffled in this case. To regularize the distribution of the data and improve training efficiency, the per-atom energies of the whole dataset have been standardized by subtracting the mean value and dividing by the standard deviation. Data samples are then batched with batch size $N = 6$. A random seed is set to enable reproducibility of the process. The objective function, or loss, to minimize is the mean absolute error (MAE) of the difference between the predicted energy $\hat{E}_i$ and its target value E_i, averaged over the batch:

$$\mathcal{L}_{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{E}_i - E_i| \; . \tag{5.8}$$

For each batch, the gradient of the loss is evaluated with respect to all the trainable parameters (weights and biases) of the network. Then, the optimization algorithm minimizes the loss by adapting the parameter values. At the end of each epoch (when all the batches are evaluated) the training convergence is assessed by evaluating the MAE over all the test data. In our setting an Adam [79, 104] optimizer was adopted. The initial learning rates, $\alpha = 10^{-4}$ for DimeNet and $\alpha = 10^{-3}$ for SchNet, have been fixed by performing preliminary tests. A linear scheduler was used to reduce the learning rate if the loss did not decrease significantly; more precisely, for DimeNet (SchNet respectively) the learning factor is reduced by a rate of 1/10 (respectively, 1/2) each time the test loss was detected not to have improved by at least 1% (respectively, 5%) over the last 10 (respectively, 3) epochs. The more strict requirements adopted for SchNet are due to its observed higher computational cost and difficulty for the loss to converge to the minimum. The training is stopped when 100 training epochs have been performed.

Using a Tesla P100 GPU with 16GB RAM, the training time amounts to $\sim$ 11 min/epoch for DimeNet and $\sim$ 22 min/epoch for SchNet, which means a total training time of $\sim$ 18 and $\sim$ 37 hours, respectively. For a rough comparison, we also trained GAP on the same dataset, by using Intel Xeon E7 4860v2 CPU with $\sim$ 317GB RAM, and the training lasted $\sim$ 60 hours. Final values of the test MAE are in the order of magnitude of tens of meV. Inference latencies have been evaluated for 54 and 128 atoms lattices and are of the order of tens of milliseconds, with the exception of a value of 104 milliseconds for SchNet on the smaller lattice: being a lighter model, SchNet relies less on GPU than DimeNet and uses only $\sim$ 1% of resources during 54 atoms inference, while DimeNet uses $\sim$ 25%. With the more demanding 128 atoms lattice, latencies are closer and in the order of tens of milliseconds, as both the models use better the resources. Metrics about training time, test MAE and inference latency are summarized in Table 5.1.

Metric	**Unit**	**SchNet**	**DimeNet**
Training time	min./epoch	$\sim$ 22	$\sim$ 11
Test MAE	meV	54.8	23.3
Inference latency (54 atoms cube)	sec.	0.104	0.040
Inference latency (128 atoms cube)	sec.	0.041	0.053

Table 5.1: Training time, test MAE and inference latency for SchNet and DimeNet.

In the original papers [138, 82] atomic embeddings have size of $F = 64$ for SchNet and $F = 128$ for DimeNet. We tested both values on both models, and obtained that while DimeNet improves slightly from 64 to 128 (test MAE from 24.85 to 23.3), SchNet makes a sensible leap forward (test MAE from 76.0 to 54.8). Consequently, an embedding size of 128 was set for both models. We consider this aspect interesting and being worth of future investigation.

The cutoff value is determined as a trade-off between two competing requirements: on one hand, the higher is the value of the cutoff, the higher is the number of connected atoms within an interaction block; on the other hand, the higher is this number, the higher is the computational cost during training. For this reason, and considering that DimeNet is a much more complex network in which also triplets of atoms are considered, the cutoff radius is different for the two models: $r_{cut} = 5.0$ Å for SchNet; 3.5 Å for DimeNet. Using a larger cutoff (up to 4 Å) for DimeNet did not increase the accuracy but did increase the computational time.

The presence of seven interaction blocks in DimeNet with respect to three in SchNet alleviates for the shorter r_{cut}, allowing the network to receive messages from distant atoms and to adequately model long-range interactions.

The code generated to obtain the results reported in this work can be found at the GitHub repository of the project: `https://github.com/AilabUdineGit/GNN_atomistics/`

5.3.2 Dataset

We use a DFT database [41] of bcc ferromagnetic iron in our study. The database is generated by delicate collinear spin-polarized plane wave DFT computations, which includes the following subsets.

- DB1: Primitive unit cell under arbitrary pressures at 300K
- DB2: $3\times3\times3$ and $4\times4\times4$ supercell under a range of pressures and temperatures
- DB3: $3 \times 3 \times 3$ supercell containing a vacancy under a range of pressures and temperatures the same as DB2
- DB4: $4 \times 4 \times 4$ supercell with divacancies at 800K
- DB5: $4 \times 4 \times 4$ supercell with 3, 4 and 5 vacancies at 800-1000K
- DB6: $4 \times 4 \times 4$ supercell containing self- and di-interstitials at 100-300K
- DB7: $1 \times 1 \times 6$ supercell with (100), (110), (111) and (112) free surfaces
- DB8: $1 \times 1 \times 6$ supercell with γ surfaces on (110) and (112) plane

All structures in DBs other than DB1 are bcc lattices; structures in DB1 are primitive unit cells of bcc lattices. More details about the database can be found in the original paper [41]. The DFT database is computed by using the open source codes QUANTUMESPRESSO [51, 50]. An ultrasoft GGA PBE pseudopotential from 0.2.1 pslibrary is employed. The kinetic energy cutoff for wavefunctions and charge density are set to be 90 and 1080 Ry, respectively. The k spacing is set to be less than 0.03 Å^{-1}.

The database is publicly available at the *Materials Cloud* site: `https://archive.materialscloud.org/record/2017.0006/v2`.

5.3.3 Results

The SchNet and DimeNet Fe potentials are benchmarked against either published DFT data [41] or data computed with Quantum Espresso based on settings (k-mesh and energy convergence) consistent with the training database [41]. The equation of state is computed with GNNs by varying the lattice constant $a_0 = 2.834$ Å of the primitive unit cell within a range of $\pm5\%$ volumetric change around the equilibrium volume computed with DFT. As shown in Fig. 5.5, both GNNs reproduce the DFT data with high accuracy.

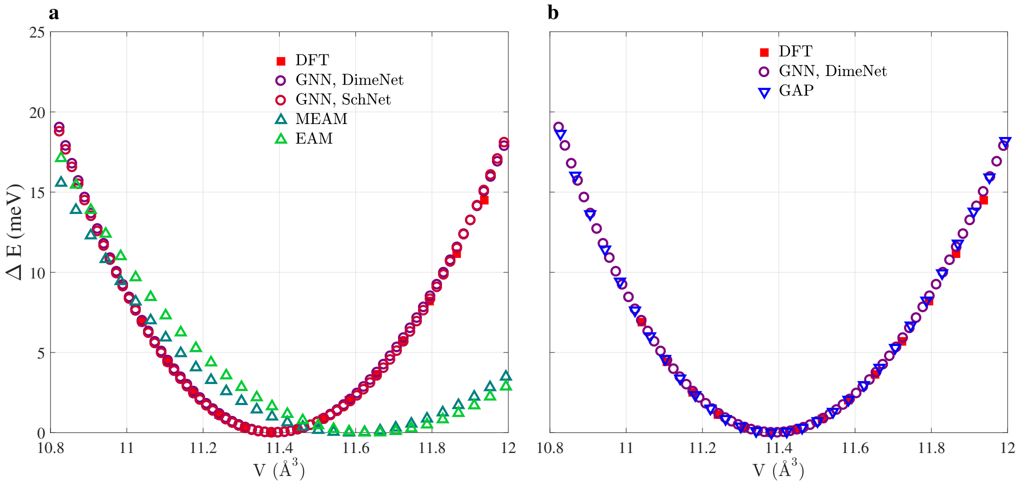


Figure 5.5: **a** Equation of state of SchNet and DimeNet compared with DFT data and the EAM [129] and MEAM [102] empirical potentials. **b** Equation of state of the DimeNet compared with DFT data and the state-of-the-art GAP iron potential [41].

To compare the performance of GNNs with empirical potentials, we compute the equation of state, equilibrium volume and buk modulus with two broadly used empirical potentials: EAM [129], which is based on pairwise interactions; and MEAM [102], that includes higher-order interactions (e.g. angular-dependent terms). Both GNN potentials reproduce the DFT results with high accuracy, while both the equilibrium volume and the curvature of the empirical potentials are far from the DFT results (see Fig. 5.5a). One reason for the discrepancy is that the empirical potentials are fitted to the experimental data of the equilibrium volume $V_0 = 11.7$ Å^3, which is obtained by extrapolation to T=0K [102]. However, despite being fitted to such value, both EAM and MEAM visibly underpredict the experimental equilibrium volume. In contrast, both GNNs can reproduce closely the dataset they have been trained to and, as shown in Fig. 5.5b, the level of accuracy is comparable with the state-of-the-art GAP interatomic potential for BCC iron [41].

The equilibrium volume and bulk modulus of iron are computed by fitting the Birch-Murnaghan equation of state to the energy-volume curve. The result of the fitting for the GNNs and DFT data is reported in Table 5.2.

Property	**Unit**	**DFT**	**SchNet**	$\boldsymbol{\varepsilon_{SN}}$	**DimeNet**	$\boldsymbol{\varepsilon_{DN}}$	**GAP** [41]	$\boldsymbol{\varepsilon_{GAP}}$
a_0	Å	2.834	2.834	0.0%	2.834	0.0%	2.834	0.0%
B_0	GPa	199.8 ± 0.1	199.0	-0.4%	199.4	-0.2%	198.2	-0.8%

Table 5.2: T=0K lattice parameter a_0 and bulk modulus B_0 for α-iron. GNN results are compared to DFT data. The relative errors of SchNet (ε_{SN}), DimeNet (ε_{DN}) and GAP (ε_{GAP}) with respect to DFT are also shown.

As indicated by the relative errors ε_{SN} and ε_{DN}, both SchNet and DimeNet reproduce the equilibrium lattice parameter and the bulk modulus with an accuracy comparable to GAP. Both the models achieve DFT-accurate results in the equation of state, with a maximum energy difference < 0.1 meV in the volume range [11.0, 12.0] Å^3. These results thus reveal no apparent difference between the performance of SchNet and DimeNet.

In order to assess the ability of GNNs to reproduce tetragonal lattice distortions, the Bain path is evaluated and compared with DFT data (Figure 5.6).

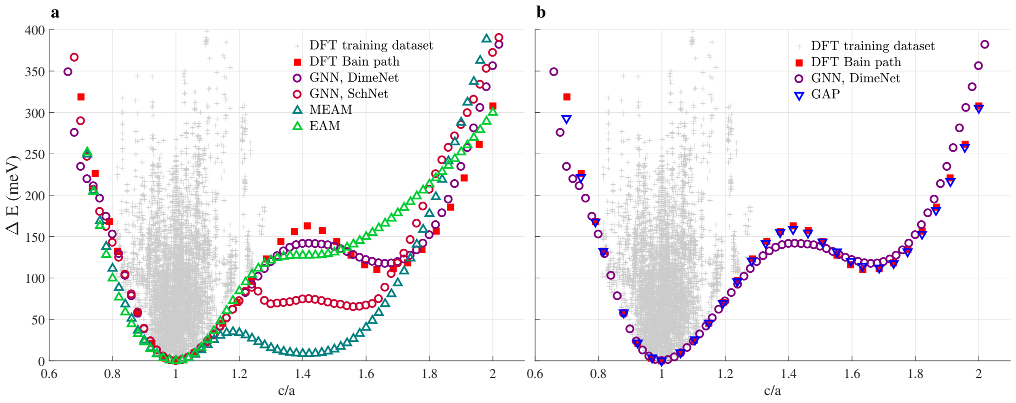


Figure 5.6: **a** DFT Bain path compared to the Bain path obtained using the GNN potentials and the empirical potentials. **b** DFT Bain path compared to the Bain path computed with DimeNet and GAP. In both panels, grey dots represent the cloud of the training data.

In the figure, DFT is used to compute the energy as a function of a distorted primitive cell. The cell is distorted at constant volume, that is by increasing one axis, c, while reducing the two other axes, a, and keeping the volume constant. Both the Fe SchNet and DimeNet potentials are then used to compute the same path. Volume optimization, i.e. finding the minimum energy configuration at the prescribed c/a by adjusting the volume, has also been performed with the GNNs potential to verify that the path does not deviate strongly from the assumed tetragonal distortion at constant volume, and no strong qualitative changes were found with respect to the result obtained with the constrained Bain path. The plot also shows the c/a distortion of the training database. Fig. 5.6a shows that SchNet interpolates well within the training set while it extrapolates poorly, with a discontinuous behaviour of the energy *vs* the c/a ratio. Instead, DimeNet can extrapolate fairly well outside of the training database, also reproducing qualitatively the energy barrier at $c/a \sim 1.4$ as well as the subsequent local energy minimum around $c/a \sim 1.65$. This specific capability of DimeNet sets it aside from SchNet, making it a more promising GNN for atomistic simulations of metals with structural transformations. Moreover, DimeNet outperforms the EAM potential, which shows no metastable minimum for BCC ferromagnetic Fe at $c/a \sim 1.65$. MEAM was fitted on data including both BCC and FCC configurations, and for this reason it deviates strongly from the DFT results, which are based on BCC ferromagnetic configurations only. Fig. 5.6b shows that DimeNet approaches the transferability of GAP for the Bain path.

Finally, the vacancy formation energy and the surface energies have been predicted for a number of crystal planes. The vacancy formation energy is calculated by using a $3 \times 3 \times 3$ cubic supercell. First, one atom of the supercell is removed and a DFT calculation is performed to relax the atoms around the vacancy. Then, the DFT total energy E_{def} of the vacancy-containing configuration is computed. The total energy E_{bulk} of the bulk defect-free supercell is also computed. The vacancy formation energy equals

$$E_{\text{v}} = E_{\text{def}} - \frac{N-1}{N} E_{\text{bulk}} \ , \tag{5.9}$$

where N is the number of atoms in the bulk system ($N = 54$ atoms in this case).

The surface energy is evaluated for four crystallographic planes, i.e. {100}, {110}, {111} and {112}. The surface is generated by creating a supercell with a vacuum region, the energy of which is indicated as E_{split}. The vacancy formation energy is computed as

$$E_{\text{surf}} = (E_{\text{split}} - E_{\text{bulk}})/2A \tag{5.10}$$

where A is the newly created surface area. The results are plotted in Fig. 5.7.

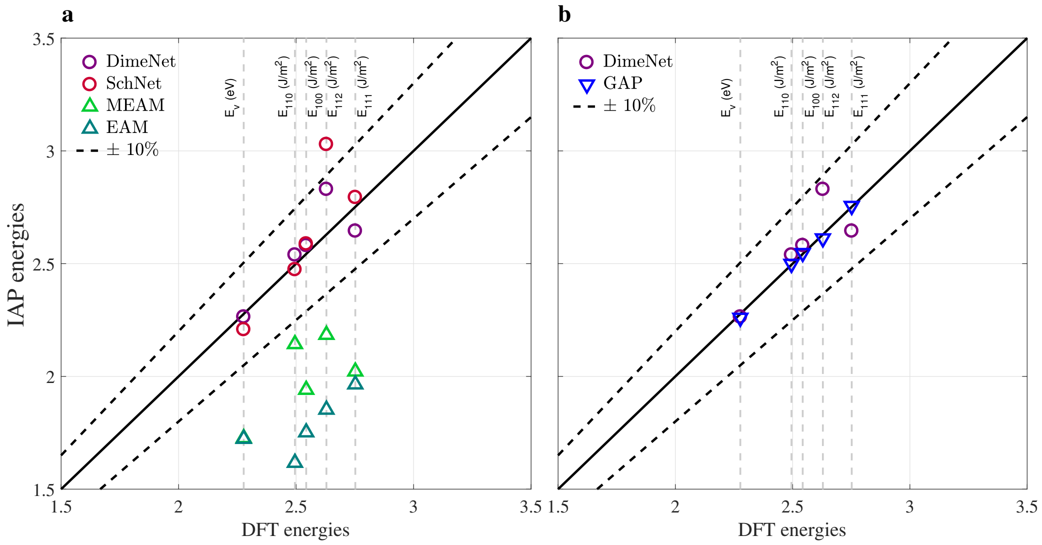


Figure 5.7: Vacancy formation energies and surface energies. The {100}, {110}, {111} and {112} surfaces are considered.

Fig. 5.7a shows that both GNNs can reproduce mostly within 10% accuracy all the considered DFT energies. The GNN potentials largely outperform both EAM and MEAM empirical potentials, that consistently underpredict the energies. DimeNet shows the largest deviation ($\sim 7\%$) for the {112} surface energy, and is otherwise approaching the predictive capabilities of the GAP potential (Fig. 5.7b).

5.3.4 Discussion

We presented a comparative study on the application of two GNN models, SchNet and DimeNet, to the prediction of properties of bcc iron. Since DimeNet was previously tested only on molecules and not on periodic structures/crystals, we implemented a version with PBCs and made it publicly available. Both models predict with DFT accuracy the energy-volume curve and related properties such as the bulk modulus and the equilibrium lattice parameter. This result is consistent with the fact that the energy-volume curve includes datapoints close to those of the training database. The investigated GNN potentials outperform closed-form empirical interatomic potentials (e.g. EAM and MEAM) and approach the accuracy of state-of-the-art interatomic potentials such as GAP. This makes the present GNNs implementation interesting for application to other metallic systems.

A different performance of DimeNet with respect to SchNet is found for configurations including tetragonal distortions (Bain path), point defects and planar defects. DimeNet can predict the energy of these configurations within the MAE, while the predictive capability of SchNet is limited. We attribute this difference to the fact that, in DimeNet, the energy depends explicitly on the angular, three-body contributions that are essential for structural transformations and for local shape distortions, while SchNet only depends on pairwise contributions. It is also remarkable that DimeNet has better transferability, e.g. considering the Bain path.

By showing the capabilities of GNNs and especially the importance of three-body terms, this work supports the further investigation of GNNs and specifically DimeNet. Activity is currently ongoing in the following directions:

- There is a number of potential improvements in terms of efficiency and accuracy of the model, which is related to the hyperparameter optimization. Further investigations will involve finding a tradeoff between chosing larger cutoff radii and/or increasing the number of interaction layers, in order to ensure the efficient description of short- and long-range interactions with high accuracy.

- Another aspect to be investigated is the number of features of both atom and edge embeddings, and their initialization. These are crucial characteristics in modeling the atomic environment, encoding properties such as the nature of the atom and of the pair interactions, and are expected to impact the model efficiency, e.g. in the convergence of the training.

- The implementation of the developed GNN Fe DimeNet potential within the LAMMPS [126] open-source package is currently ongoing and will enable the systematic simulation of thermoelastic properties, as well as linear and planar defects such as dislocations and cracks that are relevant for the investigation of the mechanical properties of metals.

- We expect that, in the spirit of Atomic Cluster Expansions (ACE) [42], the transferability of GNN potentials will be improved by including more terms in the angular descriptions, by using a different choice of the radial function (e.g. based on Chebyshev polynomials), or by setting different values of the parameters l, m in the angular functions (which, in the current DimeNet implementation, are spherical harmonics with $m = 0$), and/or by introducing higher-body terms, beyond the three-body term currently used in DimeNet. This is also the subject of current research.

5.4 Considerations about efficiency

To understand efficiency contribution of Deep Learning in the field of atomistic systems, a brief introduction of the previous approaches is necessary.

Atoms and their environment are described by the quantistic theory, and the problem of finding the energy of a system of atoms is solved exactly by the Schrödinger equations [136]. Solutions of the equation are the *wave functions*, each of which describes a particular state of the nuclei and electrons of an atomic system (for a variety of problems, and for those treated here, only the first or *groud state* is relevant). By assuming atomic nuclei as massive, classic particles, and by considering their motion much slower than that of the electrons, some simplification can be introduced in the theory without loss of accuracy, and remaining in the framework of the quantum theory. Even so, the Schrödinger equations remains unfeasible to address realistic problems: its variables are the wave functions which depend on the positions of the electrons, so also for a single atom of iron, Fe26, there are $26 \times 3 = 78$ degrees of freedom. As Hartree stated [36] in 1948, an era in which computers were not available and complex functions had to be numerically evaluated and reported in tables, "the full specification of a single wave function of neutral Fe is a function of seventy-eight variables. It would be rather crude to restrict to ten the number of values of each variable at which to tabulate this function, but even so, full tabulation of it would require 10^{78} entries ... there would still not be enough atoms in the whole solar system to provide the material for printing such a table". As a final consideration, the straight application of the quantum theory can accurately describe only systems with $\mathcal{O}(10^0)$ electrons: single, light atoms.

Further studies and considerations about the general form of the energy in the Schrödinger equations showed that the potential energy of the system in the ground-state can be written as the sum of three terms: the first, which describes the electrostatic interaction nucleus-electron; the second, which takes care of non-interacting electrons, and a third term which is *a priori* unknown. The good news is that all these terms depends only on the electron density, given by the modulus of all the wave functions: it depends only on the 3 coordinates and not on the $3N$ of all the N electrons. There is also a bad news, indeed: the unknown term is related to the specific characteristic of the system, and has to be determined by experimental observations. The Density Functional Theory (DFT) [67, 83], based on the just described approach, has marked a great leap forward with respect to Schrödinger equations, yet remaining a quantum-compliant method and obtaining correct results. It extends the range of applicability to systems with $\mathcal{O}(10^2)$ atoms.

To deal with realistic problems about materials, typically involving some $\mathcal{O}(10^6)$ atoms, a totally different approach is needed. The Molecular Dynamics (MD) methodologies abandon the quantum theory and rely on classical considerations: atoms are classical particles, and the nucleus-electron and electron-electron interactions are cumulatively described by functions of the coordinates known as the *interatomic potentials*. The evolution of the system is basically described by the classical Newton laws. Being phenomenological descriptors, interatomic potentials need to be accurately fitted to experimental evidences, or to DFT predictions which are considered to be exact. A lot of different potentials have been introduced in the last decades. One of the prominent is the Embedding Atom Method (EAM) [37]: the atom is imagined as embedded in a environment fully described by the electron density. Interactions are pairwise and each atom interacts with atoms inside a given cutoff radius, whose value is determined by experimental evidence. The Modified EAM (MEAM) [15] introduces the angular dependence of interactions and proved able to describe a wider range of atomic configurations with respect to EAM. The limits of these family of methodologies are related to their phenomenological approach: there are a large set of parameters to be fitted for a given configuration to be described, and as a consequence transferability of the models is low.

More recently, machine learning has been applied to search more flexible interatomic potentials. One example is Gaussian Approximation Potentials (GAP) [14]: it consists in the definition of the energy as linear combination of a basis functions with weights. Weights are supposed to follow a Gaussian distribution, and basis functions are functions of the descriptors (e.g., descriptors can be the interatomic distance of pairs). Then a correlation function, the *kernel*, is evaluated between different atomic environments, and used as an estimate of the similarity between them. The prediction of the energy of the environment is evaluated as the conditional probability given the observations over training environments, that is, given the kernels of the training samples. ML models perform in general better than the empirical interatomic potential like EAM or MEAM, since they are simpler models and do not need to be exactly fitted on a specific problem, so avoiding overfitting and requiring a smaller amount of training data to be trained.

Graph Neural Networks are a relatively recent introduction in Machine Learning methods for the search of interatomic potentials, and have a series of advantages [121] in terms of flexibility and effiency, as outlined below.

- Convolution is widely used in Computer Vision and is applied to input data with an ordered grid structure, represented by rows and columns, such as images. The Message Passing paradigm introduces convolutions in data domains with a graph structure, such as the atomistic structure of the materials, thus allowing the weights of the filters, radial for SchNet and radial-angular for DimeNet, to be shared for all the atoms of the structure. This solution keeps the total amount of trainable parameters at a relatively low number, in favour of a lower computational effort.

- Interactions between atoms are modeled as pair interactions inside a cutoff radius. A relatively short cutoff means a low number of atoms and edges connecting them, and small filters with a low amount of trainable weights, and in general a relatively low computing effort. However, pair-wise edges inside the cutoff radius can only account for short-range interactions. To take care of the long-range contributions an iterations of message passing steps is performed: n steps means messages coming from atoms at a distance of up to n edges. This simple schema permits scalability of the system at a relatively low computational cost.

- Embeddings contain all the info about the nature of the atoms of the system (and of their connections in the case of edge embeddings), and are linked to the very nature of the atom only by the atomic number Z which is used during initialization. So, portability to other atomic species is straightforward, as well as the extension to multi-component systems with many different atomic species, and requires a moderate or null increment in the computational cost.

- The models show an interesting ability to extrapolate and to transfer knowledge in regions outside the training dataset, as is shown in the Bain path (see Fig. 5.6). In particular DimeNet performs well in the whole represented range, that is well outside the region covered by the training dataset, see the cloud of the training samples in the figure. SchNet fails to predict the potential barrier at $c/a \sim 1.4$, but reconnects to DFT trend for higher distortion c/a. This means a great efficiency in the use of training data: using only bcc samples, models are able to extrapolate the behaviour outside the bcc region.

The presented results demonstrate that it is possible to obtain an accuracy comparable with *ab initio* methodologies like DFT, or with specifically developed models like GAP, by introducing a simple and flexible model which is based on short-range pair interactions and relies on weight sharing. This approach is efficient with respect to the scalability of the sizes and of the components, and has a relatively small amount of trainable parameters with a gain in the required computation.

6 Conclusions

Research in Deep Learning has experienced a major step forward in the last decade, and obtained some outstanding results in many fields. They are claimed to be, and indeed are, historical achievements. Further, for the first time in AI history the results obtained by research groups are become widely available in real life, and contribute to change and often improve the life experience of many people. It is easy to say that Deep Learning is nowadays in its Golden Age.

Up to this point research was only focused on the improvement of the performance in the metrics specific of the task. Improvements are possible if more complex systems are implemented, that is networks with an increasing amount of parameters able to address the increasing amount of data needed to better learn, and to better predict. This trend to "gigantism" has been recently recognized by many authors, and always shows the same side effect: the increasing of the required computation. All the story of Neural Networks and Deep Learning is tightly linked to the story of the increment in the available computing power, with great leaps forward but also frustrating stops during the so called AI winters. Indeed, this is also an era of great improvement in the hardware offering, and the recent introduction of GPUs contributed greatly to ignite this new age in Deep Learning. Nevertheless, as recent studies show and this thesis reports, improving the performance of a model requires an increment in the computational burden that is far more than proportional. So, the day in which a new shortage in available computing power will be experienced is maybe not so close, but unavoidable.

In the current scenario the scientific community is becoming aware of the trends, and it is beginning to search for containment strategies. The aim is to cut the dimensions of the models in terms of number of parameters, or total amount of training data, or required computation: in one word, to reduce the footprint metrics, but without significant loss in the performance. This effort is referred to as the search for *Efficient Deep Learning*. Efficient Deep Learning will probably be the paradigm that will drive the research in the next future.

6.1 Summary of contribution

This thesis makes a contributions in the research of Efficient Deep Learning. After an historical outline, an analysis of the current and next future scenario is carried out. Then some of the most useful methodologies and techniques of the Efficient Deep Learning approach are illustrated. They have been applied and tested to three main research projects:

- **Video Based Person re-Identification**, a trend topic in Computer Vision. Videos taken by non overlapping cameras are compared to determine if the represented person is the same or not. Here the efficiency is gained mainly by means of weight sharing introduced by the siamese architecture: test and candidate videos are simultaneously processed by two identical modules, with the same set of trainable parameters. Then, data augmentation provides a method to enrich the content of the training dataset, and has the effect to reduce the amount of required data.

- **Automatic Keyphrarse Generation**, a demanding task in Natural Language Processing. A text is given as input to the system, and it generates the related keywords and keyphrases. The use of pretrained BERT, a powerful linguistic model, greatly reduces the need for specific training data, making them necessary only in the fine-tuning phase.

- **Interatomic Potentials in Materials**, a relatively new trend in Deep Learning research. Given the microscopic configuration of a material (e.g.: the positions of the atoms), physical properties such as energy are predicted. The problem is correctly addressed by quantistic equations: a straight and well known approach, but not computationally feasible for any significant macroscopic material. Machine Learning methodologies, and recently Neural Networks models, mark a major step forward in the reduction of the computation burden. The choice of Graph Neural Networks and the message passing paradigm introduces weight sharing, since they apply convolution to the graph which represent the atomic structure.

In all the listed projects, performance metrics were comparable with state-of-the-art approaches.

6.2 Directions for future work

It is to be noted that the applied methodologies do not exhaust all the set of the possible actions. There is indeed a great margin in the research of new efficient strategies. Here we briefly introduce two of the most promising and trending approaches: *Distillation* and *Neural Architecture Search (NAS).*

Distillation [61] aims to transfer the knowledge of a large network, or ensamble of networks, to a smaller one, with the aim to improve its performance. The larger network is the teacher and is fully trained to generate soft-labels for the training data. In a typical classification task, soft labels are the outputted distribution of probability for all the classes, and are normalized with a softmax function. The smaller model is the student and is trained with a loss function given by the sum of two terms: the first is the loss given by the traditional supervised learning of the student, the second is the loss between the outputs of the student and the teacher. Student not only learns the true labels, but also to mimic the behaviour of the teacher: the knowledge of the larger teacher is transferred to the smaller student. The efficiency introduced by distillation is in the down-sizing of the model, and can be expressed by the ratio between the sizes of the teacher and the student.

A recent research topic is the Neural Architecture Search [43]: in a framework inspired to Reinforcement Learning, a controller generates a single layer to be added to a network, and an evaluation of the expanded model is performed in terms of some performance metric. If the layer score is good, a new step starts with the controller generating a new layer. Up to now the approach has been used mainly to improve the performance, but a combined loss can be evaluated in which also the footprint metrics are considered [146, 69]. NAS promises to generate Deep Learning architectures which are natively efficient; nevertheless, it is an extremely expensive technique in terms of computational cost, as it requires the hyperparameters of the added layers to be automatically set with a grid search, or similar approach. These aspects can further be studied to mitigate the problem.

A Publications

This research activity has led to publications in journals and conferences. These are summarized below.

A.1 International Journals

- Lorenzo Cian, **Giuseppe Lancioni**, Lei Zhang, Mirco Ianese, Nicolas Novelli, Giuseppe Serra, and Francesco Maresca. Atomistic Graph Neural Networks for metals: Application to bcc iron.

  Note: currently in the process of being submitted to *npj Computational Materials*; preprint: `https://arxiv.org/abs/2109.14012`

A.2 International Conferences and Workshops

- Marco Zamprogno, Marco Passon, Niki Martinel, Giuseppe Serra, **Giuseppe Lancioni**, Christian Micheloni, Carlo Tasso, and Gian Luca Foresti. Video-based convolutional attention for person re-identification. In Elisa Ricci, Samuel Rota Bulò, Cees Snoek, Oswald Lanz, Stefano Messelodi, and Nicu Sebe, editors, *Image Analysis and Processing – ICIAP 2019*, pages 3–14, Cham, 2019. Springer International Publishing.

- **Giuseppe Lancioni**, Saida S. Mohamed, Beatrice Portelli, Giuseppe Serra, and Carlo Tasso. Keyphrase generation with GANs in low-resources scenarios. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 89–96, Online, November 2020. Association for Computational Linguistics.

A.3 National Conferences

- **Giuseppe Lancioni**, Saida S. Mohamed, Beatrice Portelli, Giuseppe Serra, and Carlo Tasso. Efficient Keyphrase Generation with GANs. In *Proceedings of the 17th Italian Research Conference on Digital Libraries*, pages 1–12, Online, February 2021.

Bibliography

[1] Google Translate home page. https://translate.google.com/, 2006. [Online; accessed 07-Octber-2021].

[2] Waymo home page. https://waymo.com/, 2009. [Online; accessed 07-Octber-2021].

[3] Tesla Autopilot home page (italian). https://www.tesla.com/it_IT/autopilot, 2013. [Online; accessed 07-Octber-2021].

[4] Google Assistant home page. https://assistant.google.com/, 2016. [Online; accessed 07-Octber-2021].

[5] DeepL Translate home page. https://www.deepl.com/translator, 2017. [Online; accessed 07-Octber-2021].

[6] ICML 2020 Style and Author Instructions. https://icml.cc/Conferences/2020/StyleAuthorInstructions, 2020. [Online; accessed 19-September-2021].

[7] Dictation home page. https://dictation.io/, 2021. [Online; accessed 07-Octber-2021].

[8] SustaiNLP 2021. Second Workshop on Simple and Efficient Natural Language Processing. https://sites.google.com/view/sustainlp2021, 2021. [Online; accessed 19-September-2021].

[9] Dario Amodei and Danny Hernandez. AI and Compute. https://openai.com/blog/ai-and-compute/, 2019. [Online; accessed 14-September-2021].

[10] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[12] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[13] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Phys. Rev. B*, 87:184115, May 2013.

[14] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.*, 104:136403, Apr 2010.

[15] M. I. Baskes. Application of the embedded-atom method to covalent materials: A semiempirical potential for silicon. *Phys. Rev. Lett.*, 59:2666–2669, Dec 1987.

[16] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[17] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.

[18] Jörg Behler. Constructing high-dimensional neural network potentials: A tutorial review. *International Journal of Quantum Chemistry*, 115(16):1032–1050, 2015.

[19] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003.

[20] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 153–160, Cambridge, MA, USA, 2006. MIT Press.

[21] Gábor Berend. Opinion Expression Mining by Exploiting Keyphrase Extraction. In *IJCNLP*, 2011.

[22] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.*, 59(4–5):291–294, September 1988.

[23] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[24] Luis Camuñas-Mesa, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Neuromorphic spiking neural networks and their memristor-cmos hardware implementations. *Materials*, 12:2745, 08 2019.

[25] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. In *ACL*, 2019.

[26] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, Apr 2019.

[27] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. Keyphrase Generation with Correlation Constraints. In *EMNLP*, 2018.

[28] Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction. In *NAACL-HLT*, 2019.

[29] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. Title-Guided Encoding for Keyphrase Generation. In *AAAI*, 2019.

[30] Nicholas Choma, Federico Monti, Lisa Gerhardt, Tomasz Palczewski, Zahra Ronaghi, Mr Prabhat, Wahid Bhimji, Michael Bronstein, Spencer Klein, and Joan Bruna. Graph neural networks for icecube signal classification. pages 386–391, 12 2018.

[31] Lorenzo Cian, Giuseppe Lancioni, Lei Zhang, Mirco Ianese, Nicolas Novelli, Giuseppe Serra, and Francesco Maresca. Atomistic graph neural networks for metals: Application to bcc iron, 2021.

[32] Dung Nghi Truong Cong, Catherine Achard, Louahdi Khoudour, and Lounis Douadi. Video sequences association for people re-identification across multiple non-overlapping cameras. In *International Conference on Image Analysis and Processing*, pages 179–189, 2009.

[33] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing*, 27(10):5142–5154, 2018.

[34] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[35] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1990.

[36] Charles Galton Darwin. Douglas rayner hartree, 1897-1958. *Biographical Memoirs of Fellows of the Royal Society*, 4:102–116, 1958.

[37] Murray S. Daw and M. I. Baskes. Embedded-atom method: Derivation and application to impurities, surfaces, and other defects in metals. *Phys. Rev. B*, 29:6443–6453, Jun 1984.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2018.

[40] Daniele Dragoni, T. Daff, Gábor Csányi, and Nicola Marzari. Gaussian Approximation Potentials for iron from extended first-principles database (Data Download). *Materials Cloud Archive*, 2017.0006/v2, 2017.

[41] Daniele Dragoni, Thomas D. Daff, Gábor Csányi, and Nicola Marzari. Achieving dft accuracy with a machine-learning interatomic potential: Thermomechanics and defects in bcc ferromagnetic iron. *Phys. Rev. Materials*, 2:013808, Jan 2018.

[42] Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Phys. Rev. B*, 99:014104, Jan 2019.

[43] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[44] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[45] Keith D. Foote. A brief history of artificial intelligence. `http://www.dataversity.net/brief-history-artificial-intelligence/`, 2016. [Online; accessed 02-September-2021].

[46] Keith D. Foote. A brief history of deep learning. `https://www.dataversity.net/brief-history-deep-learning/`, 2017. [Online; accessed 02-September-2021].

[47] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[48] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (5):826–834, 1983.

[49] Niloofar Gheissari, Thomas B Sebastian, and Richard Hartley. Person reidentification using spatiotemporal appearance. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1528–1535, 2006.

[50] P Giannozzi, O Andreussi, T Brumme, O Bunau, M Buongiorno Nardelli, M Calandra, R Car, C Cavazzoni, D Ceresoli, M Cococcioni, N Colonna, I Carnimeo, A Dal Corso, S de Gironcoli, P Delugas, R A DiStasio, A Ferretti, A Floris, G Fratesi, G Fugallo, R Gebauer, U Gerstmann, F Giustino, T Gorni, J Jia, M Kawamura, H-Y Ko, A Kokalj, E Küçükbenli, M Lazzeri, M Marsili, N Marzari, F Mauri, N L Nguyen, H-V Nguyen, A Otero de-la Roza, L Paulatto, S Poncé,

D Rocca, R Sabatini, B Santra, M Schlipf, A P Seitsonen, A Smogunov, I Timrov, T Thonhauser, P Umari, N Vast, X Wu, and S Baroni. Advanced capabilities for materials modelling with quantum ESPRESSO. *Journal of Physics: Condensed Matter*, 29(46):465901, oct 2017.

[51] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21(39):395502, sep 2009.

[52] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[53] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[55] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014.

[56] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*, 2016.

[57] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[58] Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. CorePhrase: Keyphrase Extraction for Document Clustering. In *MLDM*, 2005.

[59] Babak Hassibi, David G. Stork, Gregory Wolff, and Takahiro Watanabe. Optimal brain surgeon: Extensions and performance comparisons. In *Proceedings of the*

6th International Conference on Neural Information Processing Systems, NIPS'93, page 263–270, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[61] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[62] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[63] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[64] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995.

[65] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

[66] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[67] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–B871, Nov 1964.

[68] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[69] Chi-Hung Hsu, Shu-Huan Chang, Da-Cheng Juan, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Shih-Chieh Chang. MONAS: multi-objective neural architecture search using reinforcement learning. *CoRR*, abs/1806.10332, 2018.

[70] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. *CoRR*, abs/1711.11575, 2017.

[71] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3444–3450, Hong Kong, China, November 2019. Association for Computational Linguistics.

[72] Anette Hulth. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *EMNLP*, 2003.

[73] Anette Hulth and Beáta Megyesi. A Study on Automatically Extracted Keywords in Text Categorization. In *ACL*, 2006.

[74] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision, ICCV 2009*, Proceedings of the IEEE International Conference on Computer Vision, pages 2146–2153, 2009. Copyright: Copyright 2010 Elsevier B.V., All rights reserved.; 12th International Conference on Computer Vision, ICCV 2009 ; Conference date: 29-09-2009 Through 02-10-2009.

[75] Steve Jones and Mark S. Staveley. Phrasier: A System for Interactive Document Retrieval Using Keyphrases. In *SIGIR*, 1999.

[76] Peter Bjørn Jørgensen, Karsten Wedel Jacobsen, and Mikkel Nørgaard Schmidt. Neural message passing with edge updates for predicting properties of molecules and materials. 2018. 32nd Conference on Neural Information Processing Systems, NIPS 2018 ; Conference date: 02-12-2018 Through 08-12-2018.

[77] Peter Bjørn Jørgensen, Karsten Wedel Jacobsen, and Mikkel N. Schmidt. Neural message passing with edge updates for predicting properties of molecules and materials, 2018.

[78] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. SemEval-2010 Task 5 : Automatic Keyphrase Extraction from Scientific Articles. In *Workshop on Semantic Evaluation*, 2010.

[79] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[80] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[81] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[82] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[83] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140:A1133–A1138, Nov 1965.

[84] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. Large Dataset for Keyphrases Extraction. Technical Report DISI-09-055, University of Trento, 2009.

[85] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR*, abs/1806.08342, 2018.

[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[87] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[88] Andrey Kurenkov. A brief history of neural nets and deep learning. *Skynet Today*, 2020. [Online; accessed 02-September-2021].

[89] Giuseppe Lancioni, Saida S.Mohamed, Beatrice Portelli, Giuseppe Serra, and Carlo Tasso. Keyphrase generation with GANs in low-resources scenarios. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 89–96, Online, November 2020. Association for Computational Linguistics.

[90] Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases. In *Advances in Artificial Intelligence*, 2016.

[91] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December 1989.

[92] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1990.

[93] T. Lee, M. Baskes, S. Valone, and J. Doll. Atomistic modeling of thermodynamic equilibrium and polymorphism of iron. *Journal of physics. Condensed matter : an Institute of Physics journal*, 24 22:225404, 2012.

[94] Youjiao Li, Li Zhuo, Jiafeng Li, Jing Zhang, Xi Liang, and Qi Tian. Video-based person re-identification by deep feature guided pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 39–46, 2017.

[95] Long-Ji Lin. Reinforcement learning for robots using neural networks (No. CMU-CS-93-103). *Carnegie-Mellon Univ Pittsburgh PA School of Computer Science*, 1993.

[96] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[97] Giuseppe Lisanti, Niki Martinel, Alberto Del Bimbo, and Gian Luca Foresti. Group Re-Identification via Unsupervised Transfer of Sparse Features Encoding. In *International Conference on Computer Vision*, pages 2449–2458, 2017.

[98] Giuseppe Lisanti, Niki Martinel, Christian Micheloni, Alberto Del Bimbo, and Gian Luca Foresti. From person to group re-identification via unsupervised transfer of sparse features. *Image and Vision Computing*, 83-84:29–38, 2019.

[99] Hao Liu, Jiashi Feng, Meibin Qi, Jianguo Jiang, and Shuicheng Yan. End-to-end comparative attention networks for person re-identification. *IEEE Transactions on Image Processing*, 26(7):3492–3506, 2017.

[100] Kan Liu, Bingpeng Ma, Wei Zhang, and Rui Huang. A spatio-temporal appearance representation for video-based pedestrian re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3810–3818, 2015.

[101] Yu Liu, Junjie Yan, and Wanli Ouyang. Quality aware network for set to set recognition. *CoRR*, abs/1704.03373, 2017.

[102] Laalitha S. I. Liyanage, Seong-Gon Kim, Jeff Houze, Sungho Kim, Mark A. Tschopp, M. I. Baskes, and M. F. Horstemeyer. Structural, elastic, and thermal properties of cementite (fe_3c) calculated using a modified embedded atom method. *Phys. Rev. B*, 89:094102, Mar 2014.

[103] Ilya Loshchilov and Frank Hutter. Fixing Weight Decay Regularization in Adam. *ICLR*, 2017.

[104] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[105] Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. Scientific Information Extraction with Semi-supervised Neural Tagging. In *EMNLP*, 2017.

[106] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981.

[107] Diego Marcheggiani, Jasmijn Bastings, and Ivan Titov. Exploiting semantics in neural machine translation with graph convolutional networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 486–492, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[108] Francesco Maresca, Daniele Dragoni, Gábor Csányi, Nicola Marzari, and William A. Curtin. Screw dislocation structure and mobility in body centered cubic Fe predicted by a Gaussian Approximation Potential. *npj Computational Mathematics*, 4:69, December 2018.

[109] Niki Martinel. Accelerated low-rank sparse metric learning for person re-identification. *Pattern Recognition Letters*, 112:234–240, 2018.

[110] Niki Martinel, Matteo Dunnhofer, Gian Luca Foresti, and Christian Micheloni. Person Re-Identification via Unsupervised Transfer of Learned Visual Representations. In *International Conference on Distributed Smart Cameras*, pages 1–6, 2017.

[111] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Unsupervised Hashing with Neural Trees for Image Retrieval and Person Re-Identification. In *International Conference on Distributed Smart Cameras*, 2018.

[112] Niki Martinel, Christian Micheloni, and Claudio Piciarelli. Distributed Signature Fusion for Person Re-Identification. In *International conference on Distributed Smart Cameras*, pages 1–6, 2012.

[113] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[114] Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller. Recurrent convolutional network for video-based person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1325–1334, 2016.

[115] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep Keyphrase Generation. In *ACL*, 2017.

[116] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *CoRR*, abs/2106.08962, 2021.

[117] Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Text. In *EMNLP*, 2004.

[118] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 09 2017.

[119] William B. T. Mock. *Pareto Optimality*, pages 808–809. Springer Netherlands, Dordrecht, 2011.

[120] Abdel-rahman Mohamed. Leading breakthroughs in speech recognition software at Microsoft, Google, IBM. `http://news.utoronto.ca/leading-breakthroughs-speech-recognition-software-microsoft-google-ibm`, 2012. [Online; accessed 02-September-2021].

[121] Tim Mueller, Alberto Hernandez, and Chuhong Wang. Machine learning for interatomic potential models. *The Journal of Chemical Physics*, 152(5):050902, 2020.

[122] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[123] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[124] Thuy Dung Nguyen and Min-Yen Kan. Keyphrase Extraction in Scientific Publications. In *ICADL*, 2007.

[125] Marco Passon, Massimo Comuzzo, Giuseppe Serra, and Carlo Tasso. Keyphrase Extraction via an Attentive Model. In *Italian Research Conference on Digital Libraries*, 2019.

[126] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of computational physics*, 117(1):1–19, 1995.

[127] Xuelin Qian, Yanwei Fu, Yu-Gang Jiang, Tao Xiang, and Xiangyang Xue. Multi-scale deep learning architectures for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5399–5408, 2017.

[128] Alyssa Quek, Zhiyong Wang, Jian Zhang, and Dagan Feng. Structural image classification with graph neural networks. In *2011 International Conference on Digital Image Computing: Techniques and Applications*, pages 416–421, 2011.

[129] Ashwin Ramasubramaniam, Mitsuhiro Itakura, and Emily A. Carter. Interatomic potentials for hydrogen in α–iron based on density functional theory. *Phys. Rev. B*, 79:174101, May 2009.

[130] Asha Rani, Gian Luca Foresti, and Christian Micheloni. A neural tree for classification using convex objective function. *Pattern Recognition Letters*, 2015.

[131] Shivansh Rao, Tanzila Rahman, Mrigank Rochan, and Yang Wang. Video-based person re-identification using spatial-temporal attention networks, 2018.

[132] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-Critical Sequence Training for Image Captioning. In *CVPR*, 2017.

[133] F. Rosenblatt. The perceptron - A perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[134] Frank Rosenblatt. Perceptron Simulation Experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

[135] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[136] E. Schrödinger. Quantisierung als eigenwertproblem. *Annalen der Physik*, 384(4):361–376, 1926.

[137] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 992–1002, Red Hook, NY, USA, 2017. Curran Associates Inc.

[138] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. Schnet – a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

[139] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8(1), Jan 2017.

[140] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

[141] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, Jan 2021.

[142] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[143] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *CoRR*, abs/1906.02243, 2019.

[144] Avinash Swaminathan, Raj Kuwar Gupta, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, and Rajiv Ratn Shah. Keyphrase Generation for Scientific Articles using GANs. In *AAAI*, 2019.

[145] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[146] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[147] Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in Big Data*, 2:2, 2019.

[148] Gerald Tesauro. Temporal Difference Learning and TD-Gammon. *Commun. ACM*, 38(3):58–68, March 1995.

[149] Neil C. Thompson, Kristjan H. Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning. *CoRR*, abs/2007.05558, 2020.

[150] Sebastian Thrun. Learning to play the game of chess. In *Advances in Neural Information Processing Systems 7*, pages 1069–1076. The MIT Press, 1995.

[151] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *ACL workshop on Multiword expressions*, 2003.

[152] Oliver T. Unke and Markus Meuwly. Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, May 2019.

[153] Evgeniya Ustinova, Yaroslav Ganin, and Victor Lempitsky. Multi-region bilinear convolutional neural networks for person re-identification. In *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 1–6, 2017.

[154] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[155] Rahul Rama Varior, Bing Shuai, Jiwen Lu, Dong Xu, and Gang Wang. A siamese long short-term memory architecture for human re-identification. In *European Conference on Computer Vision*, pages 135–153, 2016.

[156] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[157] Roberto Vezzani, Davide Baltieri, and Rita Cucchiara. People reidentification in surveillance and forensics: A survey. *ACM Comput. Surv.*, 46(2):29:1–29:37, December 2013.

[158] Minmei Wang, Bo Zhao, and Yihua Huang. PTR: Phrase-Based Topical Ranking for Automatic Keyphrase Extraction in Scientific Publications. In *ICONIP*, 2016.

[159] Taiqing Wang, Shaogang Gong, Xiatian Zhu, and Shengjin Wang. Person re-identification by video ranking. In *European Conference on Computer Vision*, pages 688–703, 2014.

[160] X. Wang, Y. Ye, and A. Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6857–6866, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[161] Jonathan Waskan. Connectionism. `https://iep.utm.edu/connect/`, 2016. [Online; accessed 02-September-2021].

[162] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[163] Bernard Widrow. *Adaptive "adaline" Neuron Using Chemical "memistors".* 1960.

[164] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 1992.

[165] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical Automatic Keyphrase Extraction. In *ACM*, 1999.

[166] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv:abs/1910.03771*, 2019.

[167] Aimee Wynsberghe. Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics*, 1, 02 2021.

[168] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1249–1258, 2016.

[169] Tian Xie and Jeffrey C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14), Apr 2018.

[170] J. Xu, Y. Cao, Z. Zhang, and H. Hu. Spatial-temporal relation networks for multi-object tracking. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3987–3997, Los Alamitos, CA, USA, nov 2019. IEEE Computer Society.

[171] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[172] Shuangjie Xu, Yu Cheng, Kang Gu, Yang Yang, Shiyu Chang, and Pan Zhou. Jointly attentive spatial-temporal pooling networks for video-based person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4733–4742, 2017.

[173] Yichao Yan, Bingbing Ni, Zhichao Song, Chao Ma, Yan Yan, and Xiaokang Yang. Person re-identification via recurrent feature aggregation. In *ECCV*, pages 701–716, 2016.

[174] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7370–7377, Jul. 2019.

[175] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constr. Approx*, pages 289–315, 2007.

[176] Hai Ye and Lu Wang. Semi-Supervised Learning for Neural Keyphrase Generation. In *EMNLP*, 2018.

[177] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *ICPR*, pages 34–39, 2014.

[178] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.

[179] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*, 2016.

[180] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. Generating Diverse Numbers of Diverse Keyphrases. *ArXiv:abs/1810.05241*, 2018.

[181] Marco Zamprogno, Marco Passon, Niki Martinel, Giuseppe Serra, Giuseppe Lancioni, Christian Micheloni, Carlo Tasso, and Gian Luca Foresti. Video-based convolutional attention for person re-identification. In Elisa Ricci, Samuel Rota Bulò, Cees Snoek, Oswald Lanz, Stefano Messelodi, and Nicu Sebe, editors, *Image Analysis and Processing – ICIAP 2019*, pages 3–14, Cham, 2019. Springer International Publishing.

[182] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a discriminative null space for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1239–1248, 2016.

[183] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter. In *EMNLP*, 2016.

[184] Wei Zhang, Shengnan Hu, and Kan Liu. Learning compact appearance representation for video-based person re-identification. *IEEE Transactions on Circuits and Systems for Video Technology*, abs/1702.06294, 02 2017.

[185] Yongzheng Zhang, A. Nur Zincir-Heywood, and Evangelos E. Milios. World Wide Web site summarization. *Web Intelligence and Agent Systems*, 2004.

[186] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. Mars: A video benchmark for large-scale person re-identification. In *European Conference on Computer Vision*, volume 9910, pages 868–884, 10 2016.

[187] Liang Zheng, Yi Yang, and Alexander G. Hauptmann. Person re-identification: Past, present and future. *CoRR*, abs/1610.02984, 2016.

[188] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[189] Zhen Zhou, Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. See the forest for the trees: Joint spatial and temporal recurrent neural networks for video-based person re-identification. In *CVPR*, pages 4747–4756, 2017.

[190] Xiaoke Zhu, Xiao-Yuan Jing, Xinge You, Xinyu Zhang, and Taiping Zhang. Video-based person re-identification by simultaneously learning intra-video and inter-video distance metrics. *IEEE TIP*, 27(11):5683–5695, 2018.

[191] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.