PhD Thesis / Teză de doctorat
Advanced Machine Learning Systems
Sisteme Avansate de Învăţare Automată
Semiotics and Information Theory in Neural Network Optimization
Semiotica şi Teoria Informaţiei în Optimizarea Reţelelor Neuronale

Summary

Student Doctorand: Bogdan-Adrian Muşat
Coordonator Ştiinţific: Prof. dr. mat. Răzvan Andonie

Chapter 1

Introduction

1.1 Motivation and Importance of Research

The PhD thesis presented in this research encompasses three significant research directions: Semiotics, Information Theory, and Neural Network Pruning, with the aim of advancing the field of deep learning and unlocking its potential across various domains. These three directions are interconnected and can complement each other to enhance the interpretability, efficiency, and semantic understanding of deep learning models in the context of visual data.

One of the primary issues we focus on concerns the incorporation of semiotics into Convolutional Neural Networks (CNNs). This endeavor holds significant promise in addressing the urgent requirement for enhanced interpretability and explainability in the realm of visual data processing [15, 123]. While CNNs excel at analyzing and extracting features from visual data, they often lack a comprehensive understanding of the underlying semantic context embedded within the data [7, 29, 128, 129]. This is where semiotics, a field dedicated to the study of symbols and signs and their meaning, becomes crucial. By incorporating semiotic principles into CNN architectures, researchers have the potential to bridge this gap and enable deeper levels of understanding.

Integrating semiotic principles into CNN architectures opens up exciting possibilities for enhancing the semantic understanding of CNN models [23]. By considering the meaning and interpretation of symbols and signs within the context of the visual data, these models can provide more meaningful interpretations and explanations for their predictions and decisions. This integration not only offers valuable insights into the reasoning process of CNNs but also allows users and stakeholders to comprehend the underlying factors driving the model's outputs.

However, one critical question arises: How can we effectively incorporate semiotic principles into CNN architectures? While the potential benefits are clear, the implementation details and techniques for representing and interpreting symbols and signs within the

layers and operations of a CNN remain open areas for exploration. Research endeavors could focus on developing methodologies that seamlessly integrate semiotic principles into the existing CNN frameworks, enabling a harmonious fusion of visual data processing and meaningful interpretation [116].

By addressing this question, researchers could pave the way for CNN models that not only excel in their ability to extract visual features but also possess a deeper understanding of the semiotic context in which those features exist. This would lead to more reliable and explainable results, bolstering trust and fostering broader adoption of CNN models in various domains where interpretability is of utmost importance. Our objective in this thesis is to address the challenge of effectively incorporating semiotic concepts into CNNs.

Another challenge we endeavor to tackle involves the integration of the Information Bottleneck (IB) framework with semiotics, offering the potential for additional advantages in terms of interpretability and efficiency. The IB framework, which is rooted in Information Theory, focuses on extracting relevant and meaningful information while discarding irrelevant details [93, 110, 111]. By combining the IB framework with semiotics researchers can leverage the principles of meaningful information extraction to guide the interpretation and understanding of symbols and signs encoded in data.

This integration plays a crucial role in guiding the information bottleneck process within deep learning models. It helps identify the most relevant signs and symbols that contribute the most to the desired output. By incorporating semiotic principles, the interpretation of these signs and symbols becomes more meaningful and interpretable, enabling a deeper understanding of the underlying data.

However, an open question arises: How can we strike a balance during the information bottleneck process, ensuring efficient compression while preserving the integrity of the semiotic meaning? It is crucial to avoid compromising the essential semantic understanding embedded in the signs and symbols. Research endeavors should explore optimization techniques that take into account both information-theoretic principles and semiotic principles. Once again, our objective is to tackle this unresolved question and determine whether our research can effectively address it.

Lastly, the final challenge we aim to address revolves around the integration of neural network pruning and information theory. Neural network pruning techniques offer a solution to optimize model efficiency and computational complexity by selectively removing unnecessary or redundant components from deep neural networks [40,46,56,61]. These networks often have a large number of parameters, making them computationally expensive and resource-intensive. Integrating information theory with neural network pruning could address the challenges of model optimization, computational complexity, and efficiency in deep learning.

Information theory provides fundamental insights into data representation, compression, and transmission. By leveraging information theory, researchers aim to optimize

the flow of information within the network and improve its efficiency. Concepts such as entropy, mutual information, and channel capacity are valuable tools to quantify the information content and redundancy within the network [18].

The combination of information theory and neural network pruning allows for the identification and retention of the most informative components while reducing computational complexity and memory footprint. Information-theoretic measures can assess the importance of different network components, enabling the pruning process to retain the most relevant features [16, 59, 68]. This approach helps address challenges like overfitting and balances model complexity with data efficiency [70, 118].

Efficient pruning algorithms leveraging information-theoretic concepts can accelerate the process. By utilizing information theory, researchers can identify less informative components, enabling faster and more targeted pruning. This reduces the computational overhead associated with training and fine-tuning pruned networks, further enhancing efficiency in deep learning. What are the ways in which efficient pruning algorithms can leverage information-theoretic concepts to accelerate the pruning process? Additionally, how can researchers effectively employ specific techniques, grounded in information theory, to identify less informative components? These questions form a big part of our thesis research, as we aim to uncover some new strategies and approaches that can enhance the efficiency and effectiveness of pruning in deep learning models.

The integration of Semiotics, Information Theory, and Neural Network Pruning in this PhD thesis offers valuable insights and practical applications for the field of deep learning. By examining the connections between these research directions, this study provides a framework to improve the interpretability, efficiency, and semantic understanding of deep learning models when analyzing visual data. These findings contribute to the ongoing advancement of the field and hold promise for future investigations. Through this research, we gain a deeper understanding of the potential and limitations of deep learning, bringing us closer to more effective and meaningful applications in various domains.

1.2 Convolutional Neural Networks

Convolutional neural networks are a type of artificial neural network that have proven to be particularly effective in computer vision tasks such as image classification [42, 53, 97], object detection [60,83], and segmentation [14,63,84]. They are inspired by the structure and function of the visual cortex in the brain, and utilize a series of convolutional layers to extract and learn features from input images.

The basic building block of a CNN is the convolutional layer, which applies a set of filters to the input image and produces a set of output feature maps. These filters are learned during the training process, and are optimized to capture useful patterns and structures in the input data. The output feature maps are then passed through

additional layers such as pooling, normalization, and fully connected layers, to generate a final output prediction.

CNNs have become a widely-used and highly effective tool in computer vision, achieving state-of-the-art performance across a range of tasks. CNNs have been applied in a variety of applications, including autonomous driving [11, 27], medical imaging [92], and facial recognition [106]. Moreover, CNNs have been adapted to other domains, such as natural language processing [113] and audio analysis [47]. The versatility and success of CNNs make them a valuable tool in many different fields of research and industry.

Overall, convolutional neural networks have had a significant impact on the field of computer vision, and are a powerful tool for solving a wide range of tasks. As research in this area continues, it is likely that we will see even more advances in CNN architectures and techniques for improving their efficiency and performance.

1.3 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning that focuses on the interaction between an agent and its environment. It involves learning how to make sequential decisions in order to maximize a long-term objective. This learning paradigm has gained significant attention in recent years due to its ability to tackle complex problems with sparse feedback, making it suitable for a wide range of applications such as robotics, game playing, and autonomous systems [102].

Q-learning is a popular algorithm used in reinforcement learning, particularly in scenarios with large or continuous state spaces. It is a model-free algorithm that estimates the action-values, also known as Q-values, directly. The Q-value of taking action a in state s, denoted as $Q(s, a)$, represents the expected cumulative reward starting from state s, taking action a, and following a certain policy π [102].

The combination of reinforcement learning and deep neural networks, known as deep reinforcement learning, has yielded remarkable advancements in the field. Particularly noteworthy is the achievement of human-level performance by agents in complex tasks like playing Atari games and defeating world champions in board games such as Go [67, 94]. These accomplishments vividly showcase the potency of reinforcement learning in addressing challenging real-world problems, consequently opening up new possibilities for autonomous systems and intelligent agents.

Another powerful paradigm that has emerged in RL is the actor-critic approach. Actor-critic algorithms combine elements of both value-based methods, such as Q-learning, and policy-based methods. They maintain two separate components: an actor that learns a policy and a critic that estimates the value function.

One popular variant of actor-critic algorithms is the Advantage Actor-Critic (A2C)

method. A2C combines the advantages of both on-policy methods and value function approximation. It uses multiple parallel actor-critic agents that interact with the environment to collect data, which is then used for both policy updates and value function estimation. This parallelization improves sample efficiency and allows for more stable updates [66].

The Deep Deterministic Policy Gradient (DDPG) algorithm is a model-free, off-policy reinforcement learning method that combines elements of both deep Q-learning and actor-critic algorithms [58]. DDPG is specifically designed to handle continuous action spaces, making it well-suited for tasks with continuous control [58, 95].

The DDPG algorithm has also found applications beyond traditional reinforcement learning tasks. In the context of neural network pruning, DDPG has been integrated into the AutoML for Model Compression (AMC) framework [45] as an effective method for guiding the pruning process. AMC leverages DDPG to learn a policy for selecting and pruning network channels based on their importance scores. The actor network in DDPG is trained to generate pruning masks that determine the connectivity of the network, while the critic network estimates the performance of the pruned models. Through the iterative training process, DDPG explores the trade-off between model size and performance, ultimately discovering compact and efficient network architectures. By incorporating DDPG into the AMC framework, neural network pruning becomes a reinforcement learning problem, enabling automatic and data-driven pruning decisions that result in highly compressed models without significant loss in accuracy [45]. The combination of DDPG and the AMC framework demonstrates the versatility of reinforcement learning techniques in addressing complex optimization problems in the field of neural network compression and model efficiency.

As ongoing research continues to push the boundaries of reinforcement learning, it holds tremendous potential for addressing complex decision-making problems across various domains. From autonomous vehicles to personalized recommendation systems, reinforcement learning provides a powerful framework for creating intelligent agents that can adapt, learn, and excel in diverse environments. The future of reinforcement learning looks promising, with the potential to shape the way we interact with technology and pave the way for a new era of AI-driven applications.

1.4 Semiotics: The Study of Signs and Symbols

Semiotics, also known as semiology, is a field of study that explores how signs and symbols are used for communication. It focuses on analyzing and interpreting the meanings of these signs in different contexts. Semiotics recognizes that signs go beyond words and visuals and include other sensory experiences like sounds, gestures, smells, tastes, and touch. By studying how signs, their contexts, and people's interpretations interact, semiotics helps us understand how meanings are created, negotiated, and shared within different cultures and societies.

At the center of semiotics is the concept of a sign, which consists of a physical form (signifier) and the mental concept it represents (signified). The relationship between the signifier and the signified is arbitrary and based on cultural conventions. This means that there is no inherent connection between them; it's a shared agreement within a particular culture or community. For example, in English, the word "dog" represents the idea of a four-legged animal, while another language might use a different word for the same concept.

Semiotics builds on the work of Swiss linguist Ferdinand de Saussure [88]. Saussure introduced the idea of the signifier and the signified, highlighting the importance of the relational nature of signs. He argued that meaning comes from the differences and relationships between signs within a system, rather than from their isolated existence. This concept led to the notion of sign systems or signifying systems, where signs are organized and structured to create meaning. These sign systems can be observed in various areas, such as language, visual arts, music, and social rituals.

In addition to Saussure's structuralist approach, semiotics encompasses various theoretical frameworks and perspectives. One influential figure in semiotics is American philosopher Charles Sanders Peirce, who proposed a triadic model of signs [79]. According to Peirce, signs can be classified into three types: icons, indexes, and symbols. Icons are signs that resemble or imitate the objects they represent, like a photograph. Index signs have a causal or contingent connection with their referents, such as smoke indicating fire. Symbols, on the other hand, rely on arbitrary associations with their meanings, based on shared cultural conventions. For example, the red octagonal shape of a stop sign symbolizes halting in many cultures.

Semiotics offers a valuable toolkit for analyzing and interpreting diverse forms of communication and cultural phenomena [6]. It allows scholars to investigate how meaning is constructed, conveyed, and understood within different contexts [22]. For instance, semiotics can be applied to analyze advertising campaigns, examining the symbols, signs, and narratives used to influence consumers [120]. In literature, semiotics helps uncover underlying structures and symbolic systems within a text, shedding light on authors' intentions and readers' interpretations [35]. Moreover, semiotics finds applications in analyzing visual arts, film, music, and non-verbal communication, providing insights into how meaning is created through diverse modes of expression [12].

The potential integration of semiotics and deep learning presents an intriguing avenue for research and exploration. Semiotics, with its emphasis on signs, symbols, and meaning, offers a conceptual framework that can enrich and inform the development and interpretation of deep learning models.

Deep learning, a subfield of artificial intelligence, involves training neural networks to learn and extract meaningful representations from large datasets [34]. It has achieved remarkable success in tasks such as image recognition [53], natural language processing [21], and speech synthesis [112]. However, one of the challenges in deep learning lies in

the interpretability of its models. Understanding the reasons behind a specific decision or prediction made by a deep learning system can be challenging due to the complexity and opacity of the underlying algorithms [39].

By incorporating semiotics into the deep learning process, researchers can introduce a layer of interpretability and meaning to the learned representations. Semiotics offers a structured approach to analyzing and interpreting signs and symbols within a specific context, shedding light on the internal representations and decision-making processes of deep learning models. It provides a framework for uncovering the embedded meanings in data and establishing connections between different features or concepts.

One potential application of integrating semiotics and deep learning is in the field of computer vision. While deep learning models have shown remarkable abilities in recognizing and classifying visual objects, interpreting their decisions remains challenging. By incorporating semiotic analysis, researchers can move beyond object identification and explore the symbolic and cultural meanings associated with visual content. This integration can reveal implicit layers of meaning in images, enabling more nuanced interpretations and potentially enhancing the performance of deep learning models in tasks like image understanding and generation.

Furthermore, the integration of semiotics and deep learning can have practical implications for the design of user-centered systems. By incorporating semiotic analysis into the training and decision-making processes of deep learning models, it becomes possible to consider the cultural, social, and contextual aspects of human communication. This integration can lead to the development of more inclusive and context-aware systems that take into account the diverse meanings and interpretations associated with signs and symbols. Such systems can better cater to the needs and preferences of users from different cultural backgrounds, promoting a more inclusive and user-friendly experience.

1.5 Information Bottleneck

The Information Bottleneck (IB) is a framework used in machine learning and information theory to identify and extract relevant information from a complex data set while discarding the redundant and irrelevant parts. The idea behind the IB principle is to find a compressed representation of the input data that retains as much relevant information as possible while minimizing the amount of noise and redundancy. In other words, the IB aims to find a balance between compression and accuracy that allows for efficient and effective information processing. The IB framework has found practical use in various research areas, enabling researchers to extract relevant information from complex data sets and improve the efficiency of information processing. In recent years, the IB has gained increasing attention as a powerful approach to tackle the challenges of data-driven modeling and decision making.

The original formulation of the IB concept was elaborated in [110] as an information

theoretical technique whose purpose is to find the best tradeoff between prediction accuracy of a variable Y and compression of the input random variable X in the code T. This is realized by the minimization of the following Lagrangian [110]:

$$\min_{P_{T|X}} I(X;T) - \beta I(Y;T) \tag{1.1}$$

where $I(\cdot,\cdot)$ is the mutual information of two random variables and β is a trade-off parameter.

Recently, the IB principle was applied to deep learning and presented by Tishby and Zaslavsky [111], as a theoretical concept meant to offer a possible explanation for the underlying mechanisms that govern modern deep learning architectures. The mechanism behind is similar with the one from the original formulation, optimizing for a latent representation T that represents a minimum sufficient statistic for an input X, by compressing any redundant information about it, while preserving the needed information to predict label Y. This is done in the same way as in Equation 1.1. Their work also proposes theoretical bounds on the generalization capability of a neural network. It is argued that good generalization is caused by good compression of the input X in the latent variable T. The IB principle suggests that deeper layers correspond to smaller mutual information values, providing increasingly compressed statistics [32]. It is important to notice that the authors do not provide any training experiments in which they use the IB formulation.

Shwartz-Ziv and Tishby [93] viewed the layers of a DNN as a Markov chain of successive internal representations of the input X. Any latent representation T is defined through the use of an encoder $P(T|X)$ and a decoder $P(\hat{Y}|T)$, where $\hat{Y}$ is the neural prediction. They defined the notion of information plane (IP) as the coordinate plane of the mutual information quantities $I_X = I(X;T)$ and $I_Y = I(T;Y)$ during many training epochs. For a multi-layer perceptron with a few layers, trained on a synthetic data problem, they noticed two important phases during training: a fitting phase, where $I(X;T)$ and $I(T;Y)$ both increase, and a compression phase, where the mutual information $I(X;T)$ starts decreasing, while $I(T;Y)$ stays mostly constant. They associated the decrease of $I(X;T)$ with compression of input X in the latent T, which avoids overfitting, thus explaining the good generalization achieved by overparametrized Deep Neural Networks (DNNs).

Saxe *et al.* [89] criticized the IP hypothesis of Shwartz-Ziv and Tishby, arguing that it is not applicable to general DNNs. They argued that the two phases observed in [93] are caused by the double-sided saturating nature of the *tanh* activation function used in their MLP, and binning of continuous activations to discrete values. The two-phase behaviour, as they empirically proved, is not present in networks which use non-saturating activation functions (like ReLU), employed by most modern DNNs. They also tested the supposed link between compression and generalization using the network

from [93], but trained it on a smaller percentage of the data, showing that even if the compression phase is noticeable in the IP, the train vs test accuracy suffers from severe overfitting.

Wickstrøm *et al.* [119] conducted the first large scale experiment using the IB principle, studying the VGG16 architecture [97] trained on CIFAR-10 [52]. They proposed a matrix-based Rényi's entropy coupled with tensor kernels over convolutional layers to estimate the intractable mutual information, in order to analyze the IP. Using this method, there is no need anymore for binning operations. One of their observation was that compression appears mostly on the training data, and is less visible in the test dataset. Going forward, they used an early stopping criterion based on a patience parameter. The training stops if the validation accuracy does not change after a predefined number of epochs. They noticed that training can be sometimes stopped even before the compression phase starts. The assumption here is that compression is linked to overfitting.

Other comprehensive reviews on the applications of the IB theory can be found in [28, 32]. Some of the conclusions drawn from these reviews are that IB needs further exploration. The compression seen in IPs does not necessarily represent learning a minimum sufficient statistic, nor that it produces good generalization. Yet, it can provide a good geometrical explanation for some of the inherent behaviour underlying DNNs, and might even open the doors for deeper theoretical understandings.

As machine learning continues to advance, the IB framework is expected to become even more relevant, especially as the need for efficient and interpretable machine learning models continues to grow. Therefore, the IB approach will likely continue to play a crucial role in improving the performance and interpretability of modern machine learning models.

1.6 Neural Network Pruning

Pruning is a technique used to decrease the total number of floating point operations per second (FLOPS) and parameters in a neural network by eliminating redundant weights. This method is commonly used to optimize the computational efficiency of deep learning models, particularly in hardware-limited environments.

The first approaches for neural pruning emerged in the '90s with the classical methods of optimal brain damage [55] and optimal brain surgeon [41]. Since then, the importance of pruning was observed in improving training and inference time, better generalization. With the emergence of large deep neural networks, pruning became even more relevant and desirable, since modern network architectures are overparametrized and there is a lot of space for optimization. As such, a large suite of methods for pruning have been proposed in the last years. Comprehensive surveys on neural network pruning can be found in [10, 26].

Automated Machine Learning (AutoML) has become increasingly important in the development of efficient neural network architectures. One of the primary reasons for this is that the search space for neural network architectures is incredibly vast, and the process of manually searching for an optimal architecture can be time-consuming and resource-intensive. AutoML techniques can help automate this process, enabling researchers and practitioners to efficiently explore a broader range of network architectures and hyperparameters [44].

One key finding that forms the basis of one of our work is from He *et al.* [45]. In their study, they employed a Deep Deterministic Policy Gradient (DDPG) agent [58] to determine the optimal sparsity level for each learnable layer, whether it was convolutional or fully connected, by maximizing the network's accuracy post-pruning. For pruning convolutional layers, they identified and sparsified the filters with the lowest total magnitude, while for fully connected layers, they discarded the smallest weights. This method of sparsification is known as structured, as weights are dropped in a specific structure, such as a whole filter, rather than being randomly dropped within a filter. Structured sparsification is more efficient as entire filters can be removed from computations, while random sparsification requires some computations within filters to be performed and others not, complicating the final arithmetic logic.

Despite its many benefits, pruning is not without its challenges. For example, it can be difficult to determine which weights, filters, or neurons to prune, and different pruning methods may yield different results depending on the architecture and task. Moreover, pruning can also lead to a loss of information, which can affect accuracy if not done carefully [25].

Nevertheless, pruning has become a crucial technique in the field of deep learning, enabling the creation of models that are more efficient, faster, and more environmentally friendly. With the ongoing advancements in hardware and the increasing demand for machine learning solutions, pruning is likely to remain a vital area of research and development in the years to come. As such, further improvements in pruning algorithms and techniques will undoubtedly continue to be an area of active research.

1.7 Mathematical Background - Spatial Entropy

When analyzing convolutional neural networks, it is crucial to consider the spatial entropy of the convolutional features rather than relying solely on simple entropy computations. CNNs operate on data with rich spatial structures, such as images or spatially correlated data. In these cases, traditional entropy calculations may overlook the spatial relationships within the data, leading to incomplete or inaccurate representations of the information content. By computing the spatial entropy of convolutional features, we can capture the spatial dependencies and correlations that are essential for understanding the underlying structure of the data. This spatial entropy provides valuable insights into the spatial distribution of information within CNNs, enabling us to make

more informed decisions regarding model compression, feature selection, and network optimization. By emphasizing the significance of spatial entropy analysis in CNNs, we can enhance our understanding of these complex models and leverage this knowledge for improved performance and interpretability.

Throughout the thesis we make use of the spatial aura matrix entropy, as initially defined in [114]. In our analysis, we examine a two-dimensional grid, denoted as X, which can also be extended to three dimensions in the case of a convolutional feature map that includes multiple channels. We define the joint probability of two features cells at spatial locations (i, j) and $(i + k,\ j + l)$ to take the values g, respectively g' as:

$$p_{gg'}(k, l) = P(X_{i,\,j} = g, X_{i+k,\,j+l} = g') \tag{1.2}$$

where g and g' are discretized variables, obtained after binning the values of the action maps. If we assume that $p_{gg'}$ is independent of $(i,\, j)$ (the homogeneity assumption [49]), we define for each pair $(k,\, l)$ the entropy

$$H(k,\, l) = -\sum_{g}\sum_{g'} p_{gg'}(k,\, l) \log p_{gg'}(k,\, l) \tag{1.3}$$

where the summations are over the number of possible binned values. A standardized relative measure of bivariate entropy is [49]:

$$H_R(k,\, l) = \frac{H(k,\, l) - H(0)}{H(0)} \in [0,\, 1] \tag{1.4}$$

The maximum entropy $H_R(k,\, l) = 1$ corresponds to the case of two independent variables. $H(0)$ is the univariate entropy, which assumes all feature cells as being independent, and we have $H(k,\, l) \geq H(0)$.

Based on the relative entropy for $(k,\, l)$, the Spatial Disorder Entropy (SDE) for an $m \times n$ image $\mathbf{X}$ was defined in [49] as:

$$H_{SDE}(\mathbf{X}) \approx \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{n} H_R(i - k,\, j - l) \tag{1.5}$$

Since the complexity of SDE computation is high, we decided to use a simplified version - the Aura Matrix Entropy (AME, see [114]), which only considers the second order neighbors from the SDE computation:

$$H_{AME}(\mathbf{X}) \approx \frac{1}{4}\Big(H_R(-1,\, 0) + H_R(0,\, -1) + H_R(1,\, 0) + H_R(0,\, 1)\Big) \tag{1.6}$$

Putting it all together, starting from a discretized feature map, we compute the probabilities $p_{gg'}$ in equation (1.2), and finally the AME in equation (1.6), which results in the spatial entropy quantity of an activation map X.

The consideration of spatial entropy in the analysis of feature maps and mutual information computation is of utmost importance, particularly within the realm of convolutional neural networks (CNNs). By accounting for the spatial characteristics of feature maps, we delve deeper into the spatial relationships and patterns inherent in the data, leading to more precise and meaningful representations. Furthermore, the integration of spatial entropy in MI computation allows for a comprehensive evaluation of the statistical dependencies between variables, enabling a better understanding of the spatial relationships captured by CNNs. This emphasis on spatiality enhances the interpretability and effectiveness of CNNs, facilitating improved modeling and analysis in various fields such as computer vision, image recognition, and spatial data processing.

Chapter 2

Signs and Supersigns in Deep Neural Models

2.1 Semiotic Aggregation as Information Concentration in Deep Learning

This section is based on our published work [73]. The original text reproduced here is part of our work and by no means is it intended to be plagiarized or used without proper attribution.

2.1.1 Motivation of Research

Convolutional neural networks were first made popular by Yann LeCun *et al.* [54] with their seminal work on handwritten character recognition, where they introduced the currently popular LeNet-5 architecture. At that time, computational power constraints and lack of data prohibited those CNNs from achieving their true potential in terms of computer vision capabilities. Years later, Krizhevsky *et. al* [53] marked the start of the current deep learning revolution, when during the ILSVRC 2012 competition their CNN, entitled AlexNet, overrun its competitor from the previous year by a margin of almost 10%. Since then, research on novel CNN architectures became very popular producing candidates like VGG [97], GoogleNet [104], ResNet [42], and more recently EfficientNet [107].

Despite the ability of generating human-alike predictions, CNNs still lack a major component: interpretability. Neural networks in general are known for their black-box type of behavior, being capable of capturing semantic information using numerical computations and gradient-based learning, but hiding the inner working mechanisms of reasoning. However, reasoning is of crucial importance for areas like medicine, law, finance, where most decisions need to come along with good explanations for taking

one particular action in favor of another. Usually, there is a trade-off between accuracy and interpretability. For instance, extracted IF-THEN rules from a neural network are highly interpretable but less accurate.

Since the emergence of deep learning, there have been efforts to analyze the interpretability issue and come up with potential solutions that might equip neural networks with a sense of causality [65, 91, 96, 98, 99, 126, 130]. The high complexity of deep models makes these models hard to interpret. It is not feasible to extract (and interpret) classical IF-THEN rules from a ResNet with over 200 layers.

We need different interpretation methods for deep models and an idea comes from image processing/understanding. A common technique for understanding the decisions of image classification systems is to find regions of an input image that were particularly influential to the final classification. This technique is known under various names: sensitivity map, saliency map, or pixel attribution map. We will use the term *saliency map*. Saliency maps have long been present and used in image recognition. Essentially, a saliency map is a 2D topological map that indicates visual attention priorities. Applications of saliency maps include image segmentation, object detection, image retargeting, image/video compression, and advertising design [65].

Recently, saliency maps became a popular tool for gaining insight into deep learning. In this case, saliency maps are typically rendered as heatmaps of neural layers, where "hotness" corresponds to regions that have a big impact on the model's final decision. We illustrate with an intuitive gradient-based approach, the Vanilla Gradient algorithm [96], which proceeds as follows: forward pass with data, backward pass to input layer to get the gradient, render the gradient as a normalized heatmap.

Certainly, saliency maps are not the universal tool for interpreting neural models. They focus on the input and may neglect to explain how the model makes decisions. It is possible that saliency maps are extremely similar for very different output predictions of the neural model. An example was provided by Alvin Wan[1] using the Grad-CAM (Gradient-weighted Class Activation Mapping) saliency map generator [91]. In addition, some widely deployed saliency methods are incapable of supporting tasks that require explanations that are faithful to the model or the data generating process. Relying only on visual assessment of the saliency maps can be misleading and two tests for assessing the scope and quality of explanation methods were introduced in [1].

A good visual interpretation should be class-discriminative (i.e., localize the category in the image) and high-resolution (i.e., capture fine-grained details) [91]. Guided Grad-CAM [91] is an example of a visualization which is both high-resolution and class-discriminative: important regions of the image which correspond to any decision of interest are visualized in high-resolution detail even if the image contains evidence for multiple possible concepts.

[1] https://bair.berkeley.edu/blog/2020/04/23/decisions/#fn:saliency

In our approach, we focus on the statistical aspects of the information concentration processes which appear in the saliency maps of successive CNN layers. We analyze the saliency maps of these layers from the perspective of semiotics. In computational semiotics, this aggregation operation (known as superization) is accompanied by a decrease of spatial entropy: signs are aggregated into supersign. A saliency map aggregates information from the previous layer of the network. In computational semiotics, this aggregation operation is known as *superization*, and it can be measured by a decrease of spatial entropy. In this case, signs are synthesized into supersign.

Our contribution is an original, and to our knowledge, the first application of computational semiotics in the analysis and interpretation of deep neural networks. *Semiotics* is known as the study of signs and sign-using behavior. According to [37], *computational semiotics* is an interdisciplinary field which proposes a new kind of approach to intelligent systems, where an explicit account for the notion of sign is prominent. In our work, the definition of computational semiotics refers to the application of semiotics to artificial intelligence. We put the notion of sign from semiotics into service to give a new interpretation of deep learning, and this is new. We use computational semiotics' concepts to explain decision processes in CNN models. We also study the possibility of applying semiotic tools to optimize the architecture of deep learning neural networks. Currently, model architecture optimization is a hot research topic in machine learning.

The inputs for our model are saliency maps, generated for each CNN layer by Grad-CAM, which currently is a state-of-the-art method. We compute the entropy of the saliency maps, meaning that we quantify the information content of these maps. This allows us to study the superization processes which take place between successive layers of the network. In our experiments, we show how the obtained knowledge can be used to explain the neural decision model. In addition, we attempt to optimize the architecture of the neural model employing a semiotic greedy technique.

2.1.2 Semiotic Aggregation and Information Theory in Deep Learning

In this section, our objective is to present a semiotic framework for analyzing visual representations, specifically the saliency maps, generated by multi-layered neural networks. The key focus of our approach lies in the layer-wise aggregation operation employed within these networks. While information theory serves as our fundamental computational tool, the application of the aggregation operation within a semiotic framework distinguishes our work as an interdisciplinary contribution.

In semiotics (or *semiosis*), a *sign* is anything that communicates a meaning, that is not the sign itself, to the interpreter of the sign. This definition is very general. Alternative in-depth definitions can be found in [13, 22, 90]. We consider the triadic model of semiosis, as stated by Charles Sanders Peirce. Peirce defined semiosis as an irreducible triadic relation between Sign-Object-Interpretant [79].

Charles Morris [69] defined semiotics as grouped into three branches:

- Syntactics: relations among or between signs in formal structures without regard to meaning.
- Semantics: relation between signs and the things to which they refer; their signified denotata, or meaning.
- Pragmatics: relations between the sign system and its human (or animal) user.

In a simplistic manner, semiotics already played some role in computer science during the sixties. The distinction of syntactics, semantics, and pragmatics by Charles Morris was at that time imported into programming language theory [127]. More recent results can be found in [108].

Computational semiotics is built upon a mathematical description of concepts from classic semiotics. In [36], is was stated that semantic networks can implement computational intelligence models: fuzzy systems, neural networks and evolutionary computation algorithms. Later, some computational model of Peirce's triadic notion of meaning processes were proposed [33, 37, 38].

In this work, we focus on computational aspects of semiotics in deep learning. Our semiotic infrastructure is at the intersection of Peirce's theory and information theory, a theory developed by Max Bense [8] and Helmar Frank [24].

The usual signs designate material entities which are unconsciously perceived. These so-called *first level signs* may be agglomerated into signs at the next hierarchical level, called *second level supersigns*. Iterating the process, we obtain more abstract *k-th level supersigns*. The transition from k-th level to $(k+1)$-th level supersigns in called *superization*. Frank [24] identified two types of superization:

1. **Type I** "Durch Klassesbildung" (by class formation, in German): building equivalence classes and thus reducing the number of signs. The letters of a text may be considered first level signs. The equivalence class of all types of letter "a" (handwritten, capital, and so on) is a second level supersign.
2. **Type II** "Durch Komplexbildung" (by compound formation, in German): building compound supersigns from simpler component supersigns. Reconsidering the previous example, we may obtain this way words from letters, sentences from words, and more and more complex and abstract syntactic-semantic structures afterward.

Superization is a semiotic aggregation process characterized at each perception level by a specific repertory of supersigns. Hierarchical computer vision data structures (e.g., quadtrees, multi-resolution pyramids) may be considered simplistic superizations [3, 4]. The basic idea is to treat each component as a pixel at the given hierarchical level. In this case, there is a similarity between hierarchical aggregative representation and superization processes. However, there are also differences: superizations are not simple

combinatorial processes, but subtle syntactic-semantic perception frames related to Peirce's triadic model of semiosis.

A multi-resolution image representation can be characterized at each level by an information measure. The resolution-dependent Shannon entropy can be derived from the probability distribution of grey-level events observed at that level [121]. Using the newspaper's reading analogy, at the magnified level, where only white and black patches are visible, the entropy H will be low. As the picture is brought to normal focusing distance, a great variety of grey levels become apparent, and consequently, the entropy increases. As the picture is moved further away from the eyes, the entropy decreases. Finally, it may become nearly uniformly grey in appearance, with $H \approx 0$. The observation that associates with the peak value of the entropy is one of the most meaningful observations of the picture. However, because of other factors, the maximum entropy is not always associated with the "optimal" resolution [4].

From an informational psychology view, the entropy increases until it reaches its peak value. In our opinion [3, 4], this phase may be associated to the informational adaptation of the perceiver. The subsequent entropy decrease is related to the processing of structural information [121]. The rate of decrease depends largely upon the amount of structural information in the picture. The entropy falls quickly when little structural information is available, whereas when major structural information is present, the entropy will remain high over most of its range. The variation of entropy can indicate the type and quantity of structural information in the picture in terms of size and relationships to detailed features. In the current study, we focus only on the entropy decrease phase, since the analyzed CNNs do not adapt to the inputs by changing dynamically the input image resolution.

The idea of considering the CNN layers as multi-resolution representations of the input images is interesting, but not very new [14,43,50]. For instance, in [43] a spatial pyramid pooling layer is introduced between convolutional layers and fully connected layers to avoid the need for cropping or warping of the input images. In [14], the incoming convolution layers at multiple sampling rates are applied to the convolutional layers to capture objects as well as image context at multiple scales.

In our approach, we consider the multi-resolution image representation example in the context of a semiotic recognition process, where the machine (or the interpretant) attempts to classify an input image. We imagine the recognition process as a feedforward multi-layer neural classifier where each layer performs a superization of the previous layer. We assume that the subjective information (measured by the entropy) is made available to an interpretant (i.e., the computer or the human supervisor) who attempts to classify the input image.

Let us consider the entropies computed at two successive layers: H_k and H_{k+1}. The extracted information by the interpretant can be measured by the difference $H_k - H_{k+1}$. Details can be found in [100]. We have the following result:

Theorem 2.1. *(from [24]): Superization tends to concentrate information by decreasing entropy.*

Proof
We consider separately the two types of superization. For a set $Z = (Z_1, \ldots, Z_n)$ of supersigns with the corresponding probabilities $p_1, \ldots, p_n$, $\sum p_i = 1$, using a superization of the first type, we may obtain supersigns of the next level $Z^* = (Z_1, \ldots, Z_{n-2}, \{Z_{n-1}, Z_n\})$ with the corresponding probabilities $p_1, \ldots, p_{n-2}, p_{n-1}+p_n$. We have the following inequality: $H(Z) = \sum p_i \, log \, p_i \geq H(Z^*)$.

□

For two sets of supersigns X and Y, using the second type of superization, we obtain compound supersigns from the joint set $Z = (X, Y)$. A well-known relation completes the proof: $H(X) + H(Y) \geq H(Z)$.

An intuitive application of this theorem is when we consider the neural layers of a CNN. A type I superization appears when we reduce the spatial resolution of a layer $k+1$ by subsampling layer k. This is similar to class formation because we reduce the variation of the input values (i.e., we reduce the number of signs). In CNNs, this is typically performed by a pooling operator. The pooling operator can be considered as a form of non-linear down-sampling which partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, it computes its mean (average pooling) or max value (max pooling). The formula for max pooling applied to a feature map F at layer k and locations (i, j) with a kernel of 2×2 is:

$$O_{i,j}(F) = max(F_{i,j}, F_{i+1,j}, F_{i,j+1}, F_{i+1,j+1}) \tag{2.1}$$

A type II superization is produced when applying a convolutional operator to a neural layer k. As an effect, layer $k+1$ will focus on more complex objects, composed of objects already detected by layer k. The convolutional operator for a feature map F at layer k and pixel locations (i, j) with a $3x3$ kernel W has the following formula:

$$O_{i,j}(F) = \sum_{x=0}^{2} \sum_{y=0}^{2} F(i+x, i+y) W(x, y) \tag{2.2}$$

The output O of the convolutional operator is a linear combination of the input features and the learned kernel weights. Thus, a resulting neuron will be able to detect a combination of simpler object forming a more complex one, by composition of supersigns.

We observe that the effect of superization is a tendency of entropy decrease at each level. This is different than in the case of multi-resolution image representation. In [4] we explained this difference by the following thesis: "The first level signs are perceived at a complexity level which corresponds to the "optimal" resolution." However, this

thesis does not apply to a computer recognition model (a classifier), but to human perception.

In a simplified form, a multi-layered classifier can be interpreted from Morris' semiotic theory as a transition: syntactics-semantics-pragmatics. At the end of a successful recognition process, the entropy of the output layer becomes 0 and no further information needs to be extracted. The last layer (the fully connected layer in a CNN network) is connected to the outer world, the world of objects. This may be considered the pragmatic level in Morris' semiotic theory, since it shows the relation between the input signs and the output objects which can be related to decisions and actions.

2.1.3 Signs and Supersigns in CNN Saliency Maps

Theorem 1 is a simplification of the superization processes taking place in the successive layers of saliency maps. We have both class formation and the compound formation superization, and the computed entropy is spatial. We calculate superizations at the level of saliency maps. In other words, our signs and supersigns refer to values computed in successive saliency maps computed by the Grad-CAM method.

Our hypothesis is that at the core of a CNN, both types of superizations exist. For type I superization (by class formation), the pooling operation combines signs (scalar values) by criteria like average value or maximum value, resulting in a single sign, and thus reducing their number and building equivalence classes. Another potential interpretation of the pooling operation is that it builds equivalence classes by grouping spatially neighboring elements. In our experiments (as we will see in Section 2.1.4), this phenomenon could be noticed after each pooling layer, where the magnitude of the spatial entropy of the saliency maps would have a big drop. Visually, the saliency maps start to become more concentrated around connected regions as more complex signs are formed.

For type II superization (by compound formation), it is known that CNNs compose whole objects starting from simple object parts [126]. This phenomenon describes exactly the second type of superization, as it builds compound supersigns from simpler component supersigns. They manage to do so by gradually enlarging the receptive field after each convolutional layer is applied. As the receptive field grows, a single neuron inside a hidden layer can cover a much larger region of interest from the input image and thus get activated for more and more complex objects.

What complicates the interpretation in case of CNN networks is the fact that for some layers both superizations operate simultaneously and it can be difficult to separate their effects.

Our hypothesis is that in order to decrease the spatial entropy noticeably, the first type of superization is more effective, while the second type is more responsible with building supersigns with semantic roles, not affecting spatial entropy that much.

2.1.4 Experiments and Discussion

The goal for the next experiments is to explore the variation of the spatial entropy of the saliency maps computed with Grad-CAM on some representative CNN architectures. We expect the entropy to decrease along with depth and that this can be related to type I superization processes.

We consider three standard network architectures: AlexNet [53], VGG16 [97], and ResNet50 [42]. In addition, we also study the entropy variation on a custom LeNet-5-like network[2].

The experiments are performed in different contexts on the following datasets:

1. A subset of ImageNet [19] composed of the "beaver" class from the training set, to test the pretrained and randomly initialized use-cases.
2. CIFAR-10 [52] to: *a)* train the custom network without downsampling; and *b)* test the newly trained network and a randomly initialized one, with the same architecture, using this dataset as a test set.
3. "kangaroo" class from Caltech101 [57] to test a network pretrained on ImageNet. The fact that we train and test on different (but somehow similar) datasets can have an impact on the generalization performance of the network and expose possible overfitting on the training data. This is known as *zero-shot* learning, and it can be viewed as an extreme case of domain adaptation.
4. Caltech101 [57] to test for the case where the network is pretrained on ImageNet, then trained (fine-tuned) on Caltech101. This is the *transfer learning* approach.

Experiments on standard CNN architectures

We present the experimental results for each of the considered CNN architectures. In the next tables, we use the following terms: (i) Pretrained - publicly available pretrained weights on ImageNet, (ii) Random - randomly initialized weights, (iii) Fine-tuning - fine-tuned weights starting from the pretrained ones trained on ImageNet, (iv) ImageNet - "beaver" class from the ImageNet training set, (v) Caltech101 - "kangaroo" class from the Caltech101 training set.

AlexNet [53] is composed of a sequence of convolutional, max-pooling and ReLU layers, followed at the end by fully connected layers which linearly project the extracted features from the convolutional backbone to the desired number of output classes. Table 2.1 captures the experimental result values for each layer of the network.

[2]The original LeNet-5 was introduced in [54].

AlexNet				
Layer	Pretrained ImageNet	Random ImageNet	Pretrained Caltech101	Fine-tuning Caltech101
conv1	0.6830	0.6816	0.6786	0.6829
relu1	0.6806	0.6802	0.6746	0.6795
maxpool1	0.5252	0.5113	0.5264	0.5356
conv2	0.5311	0.5100	0.5395	0.5352
relu2	0.5231	0.5096	0.5297	0.5191
maxpool2	0.4147	0.3952	0.4241	0.4116
conv3	0.4423	0.3861	0.4508	0.4474
relu3	0.4326	0.3864	0.4437	0.4454
conv4	0.4272	0.3867	0.4375	0.4292
relu4	0.4214	0.3934	0.4222	0.4304
conv5	0.4056	0.3934	0.4019	0.3925
relu5	0.3928	0.3949	0.3878	0.3784
maxpool3	0.3114	0.3038	0.3077	0.3071

Table 2.1: Entropy values for saliency maps for AlexNet at different levels in the network. Table reproduced from [73].

VGG16 [97] has a relatively simple and compact architecture, consisting of only 3×3 convolutions, max-pooling and ReLU, followed by multiple fully connected layers. The trick behind the VGG16 architecture is to use two 3×3 sequential convolution to replace a bigger 5×5 one, thus obtaining the same receptive field coverage by using less parameters. The caveat of VGG16 is that most of its parameters reside in the fully connected layers, making the network very parameter and memory inefficient. Table 2.2 depicts the entropy values at different levels of the network.

VGG16				
Layer	Pretrained ImageNet	Random ImageNet	Pretrained Caltech101	Fine-tuning Caltech101
conv1	0.8516	0.785	0.8418	0.8369
conv3	0.8017	0.7322	0.7883	0.7731
conv5	0.6742	0.6308	0.6681	0.648
conv10	0.5491	0.5155	0.5556	0.5429
conv12	0.5112	0.5155	0.5127	0.4901
conv13	0.4213	0.4035	0.4281	0.4135
conv14	0.3868	0.4288	0.3994	0.3599
maxpool5	0.3131	0.3443	0.3238	0.3086

Table 2.2: Entropy values for saliency maps for VGG16 at different levels in the network. Table reproduced from [73].

The novelty of ResNet [42] stands in the residual connections which alleviate the vanishing gradient problem, an issue that followed deep neural networks since their early

days. During backpropagation, gradients would start to gradually decrease in magnitude because of the chain rule applied to very small values, until they become 0, and in consequence many layers would lack any gradient signal on which basis to update their respective weights. ResNet solves this problem by creating residual branches from an input block to an output block in the form of $y = x + f(x)$, where x is the block's input and $f(x)$ is a sequence of multiple layers. Instead of learning a function, as in earlier architectures like AlexNet or VGG16, ResNets are trying to learn a residual for the input x, hence the name of the architecture. Entropy values for various layers are shown in Table 2.3.

For all three networks we observe a tendency of the spatial entropy to decrease, especially after max-pooling layers, which in our hypothesis are layers responsible for type I superization. Type II superization can be noticed by applying multiple consecutive convolutional layers. In this case, the spatial entropy does not necessarily decrease, but the general purpose is to enlarge the receptive field of the network, such that neurons activate for more complex objects while progressing through the layers.

Considering our above experiments and the well known fact that CNNs compose complex objects starting from simpler ones, this supports our hypothesis that type I superization is more effective for the entropy decrease. We did not notice a systematic entropy decrease for type II superization, and conclude that it is more responsible for building supersigns with semantic roles.

ResNet50				
Layer	Pretrained ImageNet	Random ImageNet	Pretrained Caltech101	Fine-tuning Caltech101
conv1	0.7854	0.6574	0.7705	0.7633
block1	0.6849	0.5108	0.6807	0.6794
block2	0.5912	0.4193	0.5901	0.582
block3	0.4574	0.3398	0.4588	0.4607
block4	0.2847	0.3019	0.2754	0.2862

Table 2.3: Entropy values for saliency maps for ResNet50 at different levels in the network. Table reproduced from [73].

CNN architecture optimization

It is known that modern neural network architectures are overparametrized [76], and so, an important emerging trend in deep learning is the optimization of such deep neural networks to satisfy various hardware constraints. An overview of such optimization techniques can be found in [17,103]. Among them, pruning is regarded as a fundamental method which has been studied since the late 1980s [70], and consists of reducing redundant operations by means of removing unnecessary or weak connections at the level of weights or layers. In the last couple of years, the state of the art pruning methods have advanced considerably and are now capable of reducing the computational

overhead of a deep neural network by a few times without incurring any loss in accuracy [10].

The experiments described in Section 2.1.4 showed that the spatial entropy of the CNN saliency maps generally decreases layer by layer, and we can relate this to semiotic superization. We aim to show how this interpretation could also help to optimize (or simplify) the architecture of the network. We perform an ablation study to see if we can determine redundant layers for pruning based on the spatial entropy information of the saliency maps. It is beyond the scope of this paper to systematically compare our approach with other CNN architecture optimization techniques. We only explore this area as a proof-of-concept, since it is the first time that such a semiotic method is used for neural architecture optimization.

On the VGG16 network, we iteratively apply the following greedy algorithm: (i) train the network on CIFAR-10 using the SGD optimizer with a learning rate of 0.01; (ii) compute the spatial entropy for each saliency map; (iii) remove a layer for which the entropy does not decrease; and (iv) repeat steps (i)-(iii) while the performance does not degrade too much.

From the results, we notice that up to 8 convolutional layers can be completely removed from the network, and this affects the performance by less than 1%. When removing the 9^{th} layer, the accuracy decreases significantly; therefore, we stop the iterative process at this stage.

An interesting finding is that the order in which we remove layers matters significantly. If small layers with few parameters from the beginning of the network are removed first, the accuracy goes down by 2% after the 3^{rd} removal. When removing from the big (over-parametrized) layers starting from the mid-end level of the network, the accuracy is maintained. The accuracy degrades especially fast after the 2^{nd} convolutional layer with 64 output channels is removed.

Our explanation is that the first two convolutional layers are crucial for the downstream performance of the network. This first part of a network, before a subsampling operation is applied, is known in the literature as stem [105]. Some variants of ResNets implement this stem as three 3×3 convolutional layers or a big 7×7 layer. These early layers are responsible with detecting low level features like edge detectors. Having only a 3×3 convolutional layer, instead of two or three, means that the receptive field before the first max-pooling operation is 3×3, which might be too small to properly detect basic strokes and edges.

The resulted network has the following configuration: 64, 64, M, 128, M, 256, M, 512, M, M, where "M" stands for max-pooling and the integers represent a convolutional layer with the respective number of output channels, followed by a ReLU non-linearity. The fully connected layers do not change from the original architecture. We compare our resulted network with VGG11, which is the smallest architecture from the VGG family. The results are displayed in Table 2.4. It can be noticed that, even when

reducing the network capacity by a factor of approximately 7.5×, the accuracy is still maintained, meaning that the network is too over-parametrized for this task.

Network	Number of parameters	Accuracy
VGG16	15.245.130	89.55%
VGG11	9.750.922	87.83%
VGG16 after 4 layers removed	9.345.354	89.57%
VGG16 after 8 layers removed	2.118.346	89.49%

Table 2.4: Comparisons on CIFAR-10 - top 1 accuracy between VGG16, VGG11 (the smallest configuration from the VGG family), VGG16 after 4 layers removed (which has roughly the same number of parameters as VGG11) and VGG16 after 8 layers removed (which is the smallest configuration which maintains the accuracy within 1% difference). Table reproduced from [73]

2.1.5 Conclusions

We introduced a novel computational semiotics interpretation of CNN architectures based on the statistical aspects of the information concentration processes (semiotic superizations) which appear in the saliency maps of successive CNN layers. At the core of a CNN, the two types of superization co-exist. According to our results, the first type of superization is effective at decreasing the spatial entropy. Type II superization is more responsible for building supersigns with semantic roles.

Beyond the exploratory aspect of our work, our main insights are twofold. On the knowledge extraction side, the obtained interpretation can be used to visualize and explain decision processes within CNN models. On the neural model optimization side, the question is how to use the semiotic information extracted from saliency maps to optimize the architecture of the CNN. We were able to significantly simplify the architecture of a CNN employing a semiotic greedy technique. While this optimization process can be slow, our work tries to use the notion of computational semiotics to prune an existing state of the art network in a top-down approach instead of constructing one using a bottom-up approach like neural architecture search. Thorough analysis has to be done in future work to consider other network architectures and robustness of the method.

Some computational improvements for calculating the spatial entropy were proposed by Razlighi *et al.* [81, 82]. The computational overhead can be significantly reduced if we accept a reduction of the approximation accuracy. We plan to use this trick in the future.

In this work we considered only one type of neural network topology: CNNs. Since CNNs are mostly suited for images, those became the subject of our study. In the future, we intend to study the connection with other fields (audio, text) and architecture types (recurrent neural networks). The semiotic approach can be extended to other deep learning models, since semiotic superization appears to be present in many architectures. The computational semiotics approach is very promising especially for the explanation and optimization of deep networks, where multiple levels of superization are implied.

2.2 A Semiotic Interpretation of the Information Bottleneck Principle

The following section is based on our published work [87]. The original text reproduced here is part of our work and by no means is it intended to be plagiarized or used without proper attribution.

2.2.1 Motivation of Research

Modern deep neural networks are computational machines capable of representing very complex functions which can solve a suite of extremely difficult tasks ranging from computer vision [75], [115] to natural language processing [125], [31] and robotics [74], [80]. Although possessing a high expressive power, model interpretability has always been a limiting factor for use cases requiring explanations of the features involved in modelling. The field of interpretability/explainability in deep learning has witnessed an explosion of published papers in recent years. Even if there is no fundamental theory that can elucidate all underlying mechanisms present in those networks, multiple works tried to deal with this issue by coming up with partial solutions, either by visual explanations [126], [91] or theoretical insights [111], [93], [73]. Therefore, we can say that the black box interpretation of deep learning is not true anymore, and what we need are better techniques to interpret these models. In the following, we will refer to three methods used for the interpretation of deep learning.

The main motivation for our work is to test the IB hypothesis on a variety of new situations, considering that there are contradictory opinions and results about the IB theory (see, for instance, [89]). Our thesis it that there is a significant similarity between the IB principle and semiotic superization. To the extent of our knowledge, this synergetic aspect was never discussed before.

We investigate the IB theory in the context of semiotic superization processes in CNN learning. In practical terms, we study the evolution of spatial entropy of CNN saliency maps to validate/invalidate the IB principle. In our experiments, we noticed a pattern similar to the fitting and compression phases appearing in the evolution of spatial entropy. We use these experimental results to train a network by freezing the redundant layers with low spatial entropy variability. This enables us to discover interesting analogies between the IB theory and semiotic superization.

Our contributions are twofold: we establish a link between the IB hypothesis of fitting and compression, and semiotic superization via the evolution of spatial entropy applied to saliency maps. As a practical application, we design a heuristic training strategy for layer-wise early stopping based on spatial entropy variability through time, which may be used to prevent overfitting during learning.

2.2.2 Superization and the Information Bottleneck Principle

This section presents our thesis on the analogy between information adaptation via superization and the IB principle.

Beside superization, it is also interesting to study another semiotic aspect in a CNN model - informational adaptation. This aspect was never discussed before. In our preliminary experiments, we observed that during training the entropy of each neural layer increases until it reaches its peak value. This phase may be associated with informational adaptation of the model. The subsequent decrease of the entropy is related to the processing of the structural information. The rate of decrease is largely dependent upon the amount of structural information in the input layer. When there is little structural information, the entropy falls quickly, whereas when there are major structural elements, the entropy of the neural layers stays high over most of its range. The manner in which the entropy changes indicate the type of information in the input layer [4]. In our opinion, this information adaptation can be related to the two distinct phases of the IB principle - fitting and compression.

In order to explore the presence of the IB hypothesis in saliency maps, we investigate:

- The evolution of the mutual information during training, between input and intermediate saliency maps, and between intermediate and output saliency maps.
- The evolution of spatial entropy for saliency maps during training.

The IB plane analysis as described in [93] tracked the two mutual information quantities $I(X;T)$ and $I(Y;T)$ and noticed the fitting and compression patterns emerging. As such, we analyze the information planes between $I(X;T)$ and $I(Y;T)$ by computing the mutual information from equation 3.1 between the first and an intermediate saliency map, and between the last and the same intermediate saliency map. The proposed experiment is meant to uncover any resemblance to the original results from [93], but applied to a different concept like saliency maps.

Going a bit further, we test a possible link between the fitting and compression patterns present in IB theory with the spatial entropy of saliency maps. In [73] the spatial entropy was studied at a single point in time (after training), going along the depth of the network. We now intend to capture the dynamics of the entropy during the whole training to see if it is governed by the same patterns. As we will see, while we can not draw any conclusions from the first scenario, in the second case there is a visible trend, similar to the fitting-compression phases studied in IPs.

Regarding saliency maps, while the superization process acts in the depth of the network, the IB principle acts on a single layer. In order to connect those two concepts, we verify the dynamics of the spatial entropy of saliency maps through the whole training process in conjuction with its layer-wise behaviour. We uncover some form of continuity pattern, presented in the next section.

2.2.3 Experiments and Discussion

We experimentally discover in this section a connection between the IB theory of fitting-compression and the evolution of spatial entropy applied to saliency maps based on similar forming patterns. We also verify the practical applicability of the spatial entropy patterns and a possible connection with the superization process. For training, we use the deep learning programming framework PyTorch [78] (version 1.6.0) and the public implementation of Grad-CAM, modified to our needs.

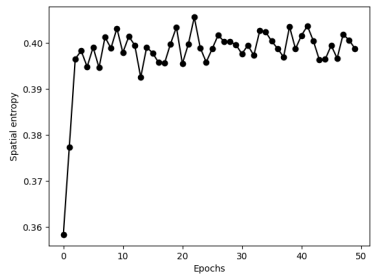
Evolution of entropy

We derive a bit from the study of mutual information and analyze the evolution of spatial entropy for saliency maps through time for the same VGG16 architecture. Whereas in [73] the spatial entropy was studied at a single point in time (after training), going along the depth of the network, we intend to capture now the dynamics of the entropy during the whole training, looking for any patterns.

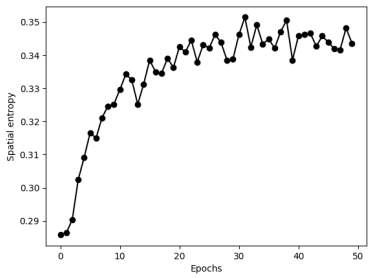
The spatial entropy was computed using the formulas from Section 1.7 and averaged over 50 random samples from the training dataset. In Table 2.5, there are plots for the computed spatial entropy through time for selected layers. We visualize again, only four layers.

A pattern is now visible, where the spatial entropy increases during the initial phase of the training and at some point flattens out. Very interesting, we noticed the same patterns on other well-known network architectures: ResNet [42], DenseNet [48] and GoogleNet [104].

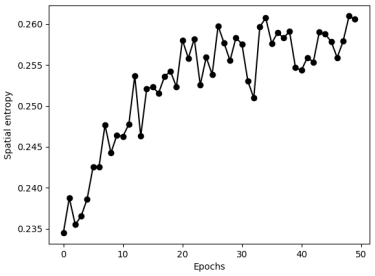
We notice that early layers exhibit a more abrupt increase of the entropy values during the first few epochs. This phenomenon could be attributed to the fact that the first layers of a CNN learn to detect easy concepts like edges, and the network learns to



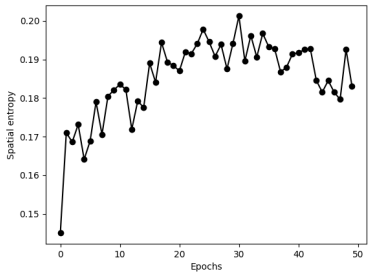
Entropy for 4^{th} layer



Entropy for 12^{th} layer



Entropy for 18^{th} layer



Entropy for 27^{th} layer

Table 2.5: Spatial entropy through time for saliency maps. Table reproduced from [87].

perform this task faster than latter layers which are responsible for detecting more complex concepts, like whole object parts [126].

In [5] it is stated that early layers are faster to learn by employing a self-supervised pre-training scheme on a single heavily augmented image. The authors prove that a single image is sufficient to learn good representations for the first few layers. In conjunction with [5], we also hypothesize that the abrupt entropy increase, observed for the first layers, is due to easier concepts being learned faster.

There are however some exceptions to the patterns in Table 2.5, present only for a few layers. While we were not able yet to find a good explanation for those different patterns, we make the following supposition. As in [73], where an entire layer was pruned if a drop in spatial entropy would not happen, we assume that we could prune a layer if the spatial entropy does not follow the patterns presented in Table 2.5 because that layer contains redundant information.

Freezing layers during training

From the patterns observed in Table 2.5, we test the hypothesis that there are links between the dynamics of the spatial entropy and the evolution of the training process. As such, we train the same VGG16 on the CIFAR-10 dataset and freeze the layers in

which the spatial entropy of the saliency map averaged over the last five epochs enters a compression phase with little variation and is below some threshold ϵ, and observe if it achieves the same accuracy as a fully trained network in the same or less number of epochs.

For a large ϵ, the layers are frozen early in the training and learning becomes prohibitive. For a small ϵ, layers are generally not frozen and the network is trained as usual. In practice, we found an ϵ value of $5e-05$ to work best. In Table 2.6, there are empirical results for a fully trained VGG16 vs a VGG16 with some layers frozen during training. The max accuracy column indicates the maximum accuracy achieved on the CIFAR-10 test set by the two versions until the specified epoch in the first column.

Epoch	**Max accuracy**		**Layers frozen**	**Running time (minutes)**	
	Frozen	Normal		Frozen	Normal
30	85.67%	85.42%	0, 17, 28	66	36
40	86.59%	86.13%	0, 7, 17, 26, 28	88	48
50	86.89%	86.81%	0, 7, 14, 17, 26, 28	110	60
60	87.45%	87.45%	0, 7, 14, 17, 26, 28	133	72

Table 2.6: VGG16 performance - normal vs frozen layers. Experiments performed on a Tesla K80 GPU on Google Colaboratory [9]. Table reproduced from [87].

As can be seen, the network is trained with some layers frozen, but still performs as good or better than the version with all the layers trained continuously. This training scheme can be considered as a form of early stopping applied at layer level, usually used to prevent a network from overfitting. Hence, we make a connection between the patterns observed in the spatial entropy of saliency maps and the training dynamics of a DNN. We observe that layer 0, which is the first convolutional layer, is among the first ones to be frozen, which empirically proves our assumption from Subsection 2.2.3 that early layers are the fastest to be learned. The downside of this method is an overhead to the computational running time, but it was not the target of our experiment.

The most similar experiment with ours is in [119], where the authors used the validation accuracy as a proxy to apply the early stopping procedure and notice that the training can be stopped before the compression phase starts. Unlike their work, we use quantities observed directly in the training dynamics of the network, and apply early stopping to prove that it has effect on the validation accuracy as well.

Creating a link between IB and semiotics

In Table 2.7, we noticed an interesting property of the spatial entropy for saliency maps. After the superization process takes place (i.e., after a drop of the entropy value), the magnitude resulted after the compression phase is approximately the same with the magnitude of where the entropy starts before superization. We observed a tendency of continuation among layers, directed by evolution of entropy and the superization

process. This might represent another inherent property of a DNN's training dynamics: the need to increase the spatial entropy up to an upper bound determined by previous layers through superization.

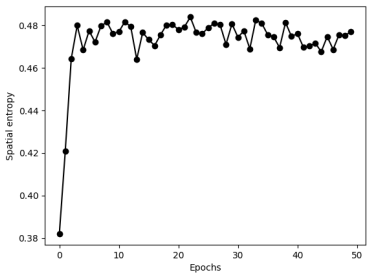
Driven by these empirical observations, we noticed an interesting connection between IB and superization. From [73] we know that a superization process takes place inside a DNN, which concentrates information, resulting in a decrease of spatial entropy. In order to reach the starting entropy from previous layers, an increase in entropy value is required for latter layers. This increase can be of any form: linear, polynomial, exponential, but as it turns out it follows very closely the same trend of fitting and compression observed in the information bottleneck theory, described in Subsection 2.2.3. The phenomenon visible in those plots is possible only if there is a mutual dependency between the IB theory (fitting-compression) and superization. This empirical observation might explain some of the training dynamics governing modern DNNs, from an information-theoretical perspective.

2.2.4 Conclusions

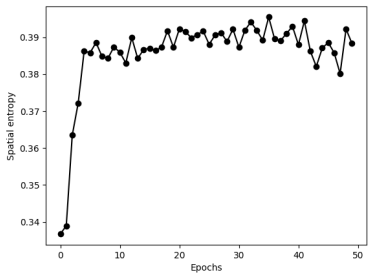
According to our experiments, there is a connection between the evolution of spatial entropy of saliency maps through time and the IB theory of fitting-compression. We noticed a mutual dependency relation between the IB theory and superization, present in DNNs where there is a drop in spatial entropy magnitude and latter layers reach the same spatial entropy from which former layers start.

We analyzed if the patterns present in the spatial entropy affect the training dynamics of DNNs. We noticed that some layers can be stopped earlier from training, based on the variability of the spatial entropy during the compression phase, and still achieve on par accuracies with fully trained counterparts. This can be regarded as a form of early stopping, applied layer-wise.

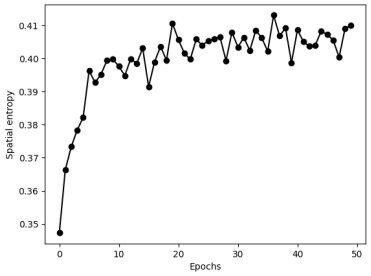
To the extent of our knowledge, this is the first application of the IB concept to saliency maps and semiotic superization. Additional experiments are due in order to draw stronger conclusions from the observations described in this work: different DNN architectures, more practical applications of the spatial entropy variability through time, a more robust theoretical understanding of the phenomena described in this work.



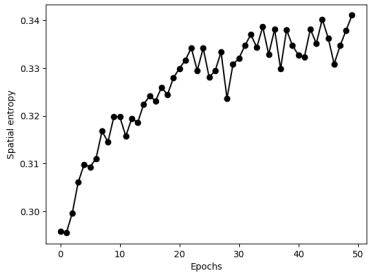
Entropy for 3^{rd} layer



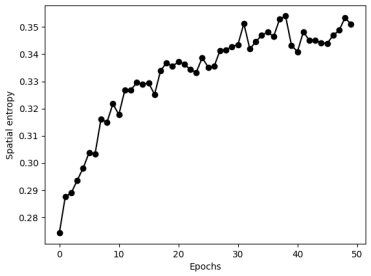
Entropy for 5^{th} layer



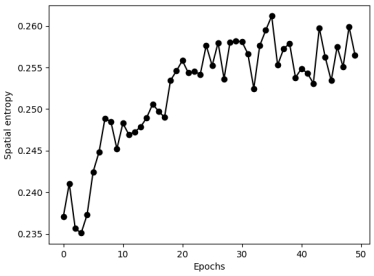
Entropy for 8^{th} layer



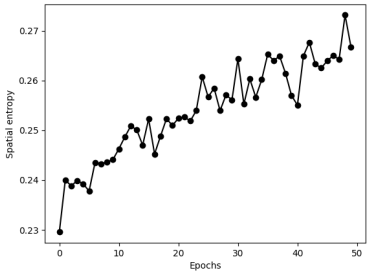
Entropy for 10^{th} layer



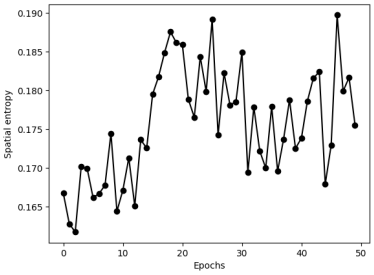
Entropy for 15^{th} layer



Entropy for 17^{th} layer



Entropy for 22^{th} layer



Entropy for 24^{th} layer

Table 2.7: Continuation of spatial entropy for saliency maps after superization. Table reproduced from [87].

Chapter 3

Information-Theoretic Pruning of Neural Networks

3.1 Pruning Convolutional Filters via Reinforcement Learning with Entropy Minimization

The following section is part of our work which will be published this year [72]. The original text reproduced here is part of our work and by no means is it intended to be plagiarized or used without proper attribution.

3.1.1 Motivation of Research

Modern convolutional neural networks (CNNs) emerged with the publication of AlexNet [53] in 2012, which paved the way for other architectures like VGG [97], ResNet [42] and EfficientNet [107]. Although these networks posses a very high capacity and perform at a super human level, they are often overparametrized [77], which induces high latency and power consumption on battery powered devices. Techniques like pruning and quantization [10, 20, 26, 30, 117] have recently become very popular, since they can generate power-efficient sub-versions of these overparametrized networks.

While pruning deals with removing unimportant weights from a network by applying a certain heuristic, quantization operates by using less bits for weights and activations, thus speeding up overall computations. In our work we only focus on structured pruning, which translates into removing entire filters from a convolutional kernel. For this, we use an automated machine learning (AutoML) framework to select the most suited percentage of structured sparsity for each neural layer [45].

AutoML is a powerful strategy used for many tasks like neural architecture search (NAS), hyperparameter search, data preparation, feature engineering [44, 124]. The principle behind it is to automate manual searching tasks and find optimal solutions

faster than we can manually do. Recently, AutoML was applied for network compression via pruning [45]. By usage of a reinforcement learning agent [58], the system can automatically choose the sparsity percentage per layer, and then a magnitude-based pruning heuristic is applied, which removes the top percentage filters with the smallest magnitude. The reward criterion that the agent uses is the accuracy of the network obtained on a randomly chosen subset from the training set or the validation set at the end of the pruning phase.

We discovered that the accuracy of the network is not the only reward criterion that can be used for AutoML network compression. Our central contribution is an information-theoretical reward function (entropy minimization) for the agent, which is completely different than the accuracy used in [58]. We utilize this information-theoretical criterion for network pruning.

The intriguing result of our work is the discovery of an interesting connection between entropy minimization and structural pruning. This could be related to the structural entropy measure recently introduced in [2], where "structural entropy refers to the level of heterogeneity of nodes in the network, with the premise that nodes that share functionality or attributes are more connected than others".

In practice, we utilize the AutoML framework from [45] to sparsify a neural network and propose as an optimization reward criterion the minimization of the spatial entropy (as defined in [73]) at each convolutional layer. Through our experiments, we empirically show that this minimization acts as a proxy for maintaining accuracy. The novelty of our work consists in discovering that there are other, more principled approaches, to neural network pruning than directly optimize the accuracy of the agent's reward function.

3.1.2 A Method for Pruning Neural Networks via Spatial Entropy Minimization

This section describes our method, which is a modification of the AMC framework [45] for structural pruning. The main difference between our approach and [45] is that our agent's reward function minimizes the spatial entropy of convolutional activations, instead of maximizing the accuracy of the model.

The AMC framework is an AutoML tool for pruning which selects the percentage of sparsity for each layer of a neural network (layer by layer), then an algorithm based on L_2 magnitude marks the top percentage filters with lowest magnitude for removal. Since the accuracy of a model is a non-differentiable function for which gradients can not be computed and used during backpropagation, a RL technique has to be used to optimize this criterion, which will be treated as a reward function. Hence, the engine driving the percentage selection for pruning is a DDPG agent [58], trained via an actor critic technique [51], using the accuracy computed on a separate dataset as a reward criterion. The reward function can be computed either on a split from the training

dataset or the validation dataset. By pursuing optimal actions (with high reward), the agent will be rewarded correspondingly and encouraged to perform similar actions in the future, while generally discouraging actions with poor reward.

AMC stores inputs and outputs for each layer at the very beginning of optimization by feed-forwarding a batch of input samples (called calibration samples). After pruning using the magnitude-based heuristic, the channels from the inputs of the calibration samples which have the same indexes as the discarded filters will be dropped as well. A least squares regression is applied to adjust the remaining weights to the new inputs and already stored outputs. After the AMC framework finds the best subnetwork configuration which maximizes the given reward function, and the weights have been adjusted as well via the least squares regression, the new network configuration is fine-tuned, as is standard in pruning literature.

Different from the original AMC formulation, we modify the accuracy based reward by introducing a function which minimizes the average of the spatial entropies of convolutional activations. Our goal is to observe if entropy minimization can be used as a criterion in place of directly computing accuracy, establishing thus a potential interesting link between the fields of neural pruning and information theory. Because the framework of AMC tries to constantly increase the amount of reward it receives, and since the mean spatial entropy formulation we use is bounded between 0 and 1 [49], we subtract it from 1 in order to minimize the term. Thus, the optimization problem for the agent becomes finding the amount of sparsity for each layer which would eventually lead to minimize the spatial entropy. In order to compute the mean value (per layer) of the spatial entropy, we use the convolutional outputs from 100 samples, which of course represents only an estimate of the full dataset. Computing the mean spatial entropy using the full dataset would be computationally too costly, and as such resorting to a high smaller sample size is enough.

Our hypothesis is that by minimizing the spatial entropy, we can achieve on par or better results than when the goal is to maximize the accuracy. If this is the case, then we can establish an empirical connection between pruning and information theory, showing that by removing redundant information from a model we can achieve the same accuracy as when we directly try to maximize it.

3.1.3 Experiments and Discussion

In this section we experimentally asses our hypothetical connection between information theory and pruning, verifying whether pruning achieved via AutoML, using minimization of spatial entropy for convolutional activations, can lead to a more compact model with similar accuracy.

For training, we used the deep learning programming framework PyTorch [78] (version 1.10.0) and the public implementation of AMC, modified to our needs.

We started by training a standard VGG16 [97] on the CIFAR-10 dataset [52]. For that, we trained for 200 epochs using the SGD optimizer with a learning rate of 0.01 and cosine annealing scheduler [64].

In order to establish a baseline to compare our method with, we used the original formulation of the AMC framework and optimize first the network using the accuracy criterion. To achieve a certain level of pruning, AMC pushes up the level of sparsity until only a predefined percentage of the total FLOPS are maintained. The ratio between the number of FLOPS after compression and the number of FLOPS before compression can measure indirectly the amount of sparsity in a network. Table 3.1 depicts the results for various FLOPS preservation percentages after fine-tuning on the CIFAR-10 dataset. For fine-tuning we used the same training optimizer and hyperparameters as described previously.

	Standard VGG16	VGG16 with 50% FLOPS	VGG16 with 20% FLOPS	VGG16 with 10% FLOPS
Accuracy	**93.58%**	**93.85%**	**93.26%**	**92.18%**
No. parameters	**14728266**	**4768242**	**912186**	**483402**

Table 3.1: Accuracy for a VGG16 network using the original AMC framework for different FLOPS preservation percentage. Original table will be published at ICAISC 2023 in one of our accepted papers.

We notice that with entropy minimization we achieved the same performance as when accuracy is used as a reward. The solution found by this method has $10\times$ less FLOPS and $\approx 38\times$ less parameters than the original VGG16 network. For entropy maximization the framework produces a solution which has indeed fewer parameters, but uses the same number of FLOPS as the method with entropy minimization. We can see though that the resulting network architecture has a much poorer accuracy performance.

	Minimization	Maximization
Accuracy	**92.36%**	**83.23%**
No. parameters	**386442**	**91290**

Table 3.2: Experiments with entropy minimization and maximization and FLOPS preservation of 10%. The accuracies are computed on the CIFAR-10 test set after fine-tuning. Original figure will be published at ICAISC 2023 in one of our accepted papers.

In the above experiments, we used a bin size of 256 for quantizing the convolutional activations before computing the spatial entropy.

In order to test the generality of our method for various other architectures, we repeated the same experiments for other popular networks: MobileNetV2 [85] and ResNet50 [42]. The results are depicted in Table 3.3. Our method is on par with the original

AMC framework for various architectures and FLOPS preservation percentages. The only noticeable drop in performance is for ResNet50, which was previously observed to contain less redundancy [117] and was the most difficult to compress, even when using accuracy as a criterion.

Architecture	Original performance	Accuracy 50% FLOPS	Accuracy 20% FLOPS	Entropy 50% FLOPS	Entropy 20% FLOPS
MobileNetV2	**94.58%**	**94.62%**	**93.59%**	**94.31%**	**93.75%**
ResNet50	**95.21%**	**95.34%**	**95.09%**	**95.27%**	**94.27%**

Table 3.3: Accuracy on CIFAR-10 test set with other architectures and various FLOPS preservation percentages. Original figure will be published at ICAISC 2023 in one of our accepted papers.

Using an information-theoretical optimization criterion, which aims to minimize entropy, we achieved the same performance as when we optimize directly the accuracy of the model. We were able to reduce the total number of FLOPS of a VGG16 architecture by $10\times$ and the number of parameters by $\approx 38\times$, while incurring minimal accuracy drop, with similar results for other popular architectures.

3.1.4 Conclusions

A standard neural network's output should ideally have a close to zero entropic value in order to confidently predict a class - not necessarily the correct one. Usually, this behavior is achieved by minimization of the cross-entropy between the network's output and the one hot encoding of the true class. This task can be sometimes burdensome, because internal layers of the network are not forced in any way to minimize the final entropy of the output layer.

In our experiments, we explicitly forced the spatial entropy of internal convolutional activations to decrease with the goal to achieve neural pruning. According to our results, using the spatial entropy as an optimization criterion in an AutoML pruning framework, we can achieve good performance for an object recognition task, without directly optimizing the final evaluation metric (accuracy in this case). Because of the overparametrization of a neural network, removing unessential information via entropy minimization helps reduce the model to its relevant (essential) components.

We established an interesting connection between information theory and neural pruning. Our result creates the premises for future applications in neural network pruning.

3.2 Accelerating Convolutional Neural Network Pruning via Spatial Aura Entropy

The following section is based on our work which will be published this year [71]. The original text reproduced here is part of our work and by no means is it intended to be plagiarized or used without proper attribution.

3.2.1 Motivation of Research

Deep Convolutional Neural Networks have achieved state-of-the-art performance on a wide range of computer vision tasks, such as image classification, object detection, and semantic segmentation [42, 53, 63]. These models typically consist of a large number of parameters, making them computationally expensive and memory-intensive to deploy on resource-limited devices, such as mobile phones and embedded systems. Training these models requires significant computational resources and time, which limits the ability to explore large-scale architectures and hyperparameters [101].

One promising approach to alleviate these challenges is pruning, which refers to the process of reducing the size of a neural network by removing unimportant weights, neurons, or filters, without significant loss in accuracy [55]. Pruning can result in more efficient models that require fewer parameters, consume less memory, and have faster inference time. This can be particularly important for real-time applications, where latency and energy consumption are critical factors [122].

Despite the potential benefits of pruning, there are also some challenges that need to be addressed. For example, pruning can lead to a significant increase in the number of training iterations required to recover the accuracy of the original model, which can offset the benefits of the reduced model size [62]. Moreover, the choice of the pruning method, the pruning rate, and the fine-tuning strategy can affect the final accuracy and the efficiency of the pruned model [25]. Therefore, it is essential to carefully design and evaluate pruning methods for different applications and network architectures.

Sarvani *et al.* introduced an Information Bottleneck theory [110] based filter pruning method that utilizes Mutual Information to determine filter significance. High Relevance (HRel) filters, those with higher MI with class labels, are considered more important and retained. The proposed method outperforms recent state-of-the-art pruning methods and has been demonstrated on LeNet5, VGG16, ResNet56, ResNet110, and ResNet50 architectures using MNIST, CIFAR-10, and ImageNet datasets.

The HRel method of estimating MI is based on Rényi's alpha entropy estimator [119]. However, kernel-based MI estimation has several issues, including kernel width selection, curse of dimensionality, and computational complexity. These issues can result in an inaccurate estimation of MI and limit the practical application of kernel-based methods.

In this paper, we build on top of their work and propose a more efficient method to

compute the MI required for filter importance selection, which reduces the necessary optimization time from almost a week of compute time to a single day. It is based on the formulation of MI as defined in our previous work [87], based on spatial aura entropy. The spatial aura entropy method is more efficient and straightforward than kernel-based estimators and does not require the selection of kernel width. Our method manages to preserve or improve the results obtained by the original work, but at a much lower computational cost.

3.2.2 A Method for Accelerating Mutual Information Estimation using Spatial Entropy

We introduce in this section our CNN filter pruning method. Primarily, it follows the HRel workflow [86].

The entropy of Y is guaranteed to be 0 because Y is encoded using one-hot encoding, which places all the magnitude on a single position. As such, in our case, the MI formula becomes:

As in [86], we compute the MI between each activation map and the one-hot encoded ground truths. The MI between two random variables X and Y is computed as:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{3.1}$$

$$I(X,Y) = H(X) - H(X,Y) \tag{3.2}$$

In order to calculate the value of equation (3.2), we utilize the outcome of equation (1.6) for the entropy of X, as well as equation (1.5) for the joint entropy of X and Y.

First, we train a CNN until convergence. Then we compute the MI between the activation maps and the ground truth labels for each layer in the network. The activation maps represent the output of each filter in the layer when the input is passed through the layer.

Next, we rank the filters in descending order of MI and remove a certain percentage of filters with the lowest MI. The number of filters to be removed is determined for each layer based on the maximum total filters removed target and the layer's contribution to the network's overall performance.

After the filters are removed, we perform a fine-tuning step to recover the network's original performance. Fine-tuning involves training the pruned network for some epochs using a smaller learning rate than the original training rate.

Once the fine-tuning step is complete, we repeat the process of computing MI, ranking filters, and MI. This process continues until the maximum number of filters removed for each layer is achieved.

The advantage of using MI as a criterion for pruning is that it considers the information content of each filter in the network, rather than just its weight magnitude or gradient. Additionally, MI can capture the interactions between filters and their contribution to the overall performance of the network.

In the HRel method, MI is estimated using Rényi's alpha entropy estimator [119], which relies on kernel functions and is highly sensitive to the optimal kernel bandwidth of the dataset [109], as noted by the authors themselves. This sensitivity results in a computationally expensive procedure, as the kernel bandwidth needs to be updated continuously during training and fine-tuning, making the pruning mechanism burdensome. As such, pruning a network using the public code that the authors provided takes almost a week for a VGG16 network [97], rendering it impractical for real-life applications.

In contrast to the original HRel technique, our method employs a different approach by using a MI estimation that does not rely on kernel functions. Instead, we estimate MI using the spatial aura entropy (the AME simplified version) described in Section 1.7. This allows for efficient computation of MI using as few as 100 samples. We eliminate the computational burden of continuously updating the kernel bandwidth during training and fine-tuning, making it significantly faster and more practical for real-life applications. Our method still preserves or even improves upon the results achieved by the original HRel paper, demonstrating its effectiveness in efficient filter selection.

3.2.3 Experiments and Discussion

This section presents empirical proof of the effectiveness of our approach. We evaluate our method on the widely used CIFAR-10 benchmark dataset [52]. We modified the publicly available code provided by the HRel authors [86] altering the MI estimation procedure. Although we observed some variations in the baseline performance of ResNet architectures from what was reported in the original paper, we compare our method and HRel employing on the same starting baseline performance.

VGG16

To evaluate the effectiveness of our proposed method, we apply it to the popular VGG16 architecture [97], which consists of 13 convolutional layers and two fully connected layers. We prune filters from the convolutional layers and train the network for 300 epochs with an initial learning rate of 0.1, which we reduce by a factor of 10 at epoch numbers 80, 140, and 230, until the baseline accuracy is achieved. Subsequently, we prune and retrain the network for 90 epochs with a learning rate of 0.01, which we reduce by a factor of 10 at epochs 40 and 70.

Table 3.4 presents a comparison of the accuracy achieved by the original HRel formulation and our proposed method for different configurations of the remaining number of filters per layer on the CIFAR-10 test set. Our method outperforms HRel for both

configurations, demonstrating its effectiveness in the pruning process. Specifically, the original VGG16 network achieves a test accuracy of 93.95%, while our proposed method yields test accuracies of 93.22% and 93.4% for Configuration 1 and Configuration 2, respectively. These results highlight the potential of our method in enhancing model efficiency and accuracy.

	HRel	**Our Method**
Original network: 64-64-128-128-256-256-256 -512-512-512-512-512-512	93.95%	93.95%
Configuration 1: 19-48-64-64-95-107-107 -175-71-71-44-44-56	93.15%	93.22%
Configuration 2: 24-40-64-77-176-134-120 -141-56-56-56-56-56	93.22%	93.4%

Table 3.4: Comparison of accuracy achieved by HRel and our method on the CIFAR-10 test set across different pruning configurations. The table shows that our method outperforms HRel for both configurations, demonstrating its effectiveness in the pruning process. Original figure will be published at IV2023 in one of our accepted papers.

ResNet56

ResNet56 [42] is a more complex and deeper neural network architecture compared to VGG16. It consists of 55 convolutional layers and 1 fully connected layer, with all convolutional layers (except the first one) grouped into three blocks, each containing 18 convolutional layers. The first, second, and third blocks have 16, 32, and 64 filters, respectively. To achieve the baseline accuracy, the network is trained for 180 epochs with an initial learning rate of 0.1, which is then decreased by a factor of 10 at epoch numbers 91 and 136.

After pruning, the network is retrained for 200 epochs with a learning rate of 0.01, which is then decreased by a factor of 10 at epochs 100 and 150. We observe that the ResNet56 architecture requires twice the number of fine-tuning epochs using our method compared to the original HRel framework. However, the additional computational time is negligible when compared to the overall runtime of the pruning process from the original HRel.

After pruning, the final remaining number of filters in the convolutional layers of each block are 8, 15, and 30, respectively. Table 3.5 shows that our method outperforms HRel in terms of the number of remaining filters and achieved similar accuracy. Specifically, the original HRel framework achieves an accuracy of 92.74% with a filter configuration

of 8-16-32, while our method achieves an accuracy of 92.76% with a filter configuration of 8-15-30.

	HRel	**Our Method**
Original network: 16-32-64	93.45%	93.45%
Configuration: 8-15-30	92.74%	92.76%

Table 3.5: Comparison of accuracy achieved by HRel and our method on the CIFAR-10 test set using the ResNet56 architecture with the filter configuration of 8-15-30 after pruning. Original figure will be published at IV2023 in one of our accepted papers.

ResNet110

ResNet110 [42] is a deep neural network architecture that is composed of 109 convolutional layers and a single fully connected layer. The structure of ResNet110 is similar to that of ResNet56, where the convolutional layers are grouped into three blocks, except for the first convolutional layer. However, in ResNet110, each block contains 36 convolutional layers with 16, 32, and 64 filters, respectively.

To achieve the baseline accuracy, the network is trained for 240 epochs with an initial learning rate of 0.1, which is decreased by a factor of 10 at epoch numbers 88, 160, and 190. The first convolutional layer is not pruned, similar to other pruning methods. After pruning, the network is fine-tuned for 70 epochs with a learning rate of 0.01, which is decreased by a factor of 10 at epochs 30 and 50. The number of remaining filters in each block's convolutional layer is 8, 15, and 30, respectively, after pruning.

Table 3.6 presents a comparison of the accuracy achieved by our proposed method and HRel on the CIFAR-10 test set using the ResNet110 architecture with a filter configuration of 8-15-30 after pruning. The original network with a filter configuration of 16-32-64 achieved an accuracy of 93.27%. Our method achieved a high accuracy of 92.42%, which is slightly improved compared to HRel's accuracy of 92.36%, with a difference of 0.06%.

	HRel	**Our Method**
Original network: 16-32-64	93.27%	93.27%
Configuration: 8-15-30	92.36%	92.42%

Table 3.6: Comparison between the accuracy achieved by HRel and our proposed method on the CIFAR-10 test set using ResNet110 architecture with a filter configuration of 8-15-30 after pruning. Original figure will be published at IV2023 in one of our accepted papers.

The experiments demonstrate the effectiveness of our method in improving the accuracy and efficiency of CNN pruning. The incorporation of spatial aura entropy into MI calculation provides a more robust and efficient pruning method. This is achieved by improving the accuracy of filter importance selection criteria and reducing the optimization time required for MI computation. Our method not only outperforms the baseline HRel method in terms of pruning performance but also significantly reduces the computational cost, making it a practical and scalable solution for deep learning model compression.

3.2.4 Conclusions

We introduced an alternative solution to the matrix-based Rényi's alpha entropy estimator used in the HRel method proposed in [86]. This improvement significantly reduces the optimization time from almost a week to a single day, making it a more practical and efficient method for large-scale model pruning. Our method is an efficient and effective solution to reducing the computational complexity and memory footprint of deep learning models, providing a viable alternative to existing methods with improved pruning performance and computational efficiency.

Chapter 4

Conclusions

4.1 Conclusions

The works presented in this thesis share a common thread that revolves around the subject of information theory, specifically with a focus on entropy. The first two articles in this collection explore the intersection of semiotics and deep learning by examining the fluctuating patterns of spatial entropy within saliency maps. On the other hand, the last two articles propose practical approaches that incorporate spatial entropy into the realm of neural network pruning. Consequently, for each individual article, we present a comprehensive overview of the main contributions and provide relevant conclusions.

The results presented in Section 2.1 are centered around a novel application of computational semiotics in the analysis and interpretation of deep neural networks, which brings a fresh perspective to the field. By integrating concepts from semiotics, a discipline focused on signs and their usage, we offer a unique understanding of how decision processes unfold in CNN models. Moreover, we explore the potential of leveraging semiotic tools to optimize the architecture of deep learning neural networks, an area currently under active investigation in the field of machine learning.

Section 2.2 showcases our investigation, where we found a compelling connection between the evolution of spatial entropy in saliency maps and the IB theory of fitting-compression. Through our experiments, we observed a mutual dependency relationship between the IB theory and semiotic superization in Deep Neural Networks. Specifically, as we progressed through the layers of the network, we noticed a significant drop in the magnitude of spatial entropy, with later layers reaching the same spatial entropy level as the earlier layers. This finding suggests that the IB principle of compressing relevant information plays a role in the superization process of DNNs.

In Section 3.1, our central contribution is the introduction of an information-theoretical reward function based on entropy minimization for AutoML network compression. This novel criterion differs from the traditional accuracy-focused approach and offers a prin-

cipled alternative for network pruning. By considering the entropy of neural activations in hidden layers, we tap into a rich source of information that is often overlooked in the context of pruning. While cross-entropy is commonly used to measure the error between a network's predicted output and the true class distribution, we argue that the entropy of neural activations provides valuable insights into the distribution of information within the network.

Finally, in Section 3.2, we introduce a novel and efficient method to compute Mutual Information for filter importance selection in the context of model pruning. Building upon the Information Bottleneck theory-based filter pruning method proposed by Sarvani et al., our approach addresses the limitations of existing matrix-based Rényi's alpha entropy estimators. By leveraging the concept of spatial aura entropy from our previous work, we devise a more practical and computationally efficient solution for large-scale model pruning. Remarkably, our method significantly reduces the optimization time from almost a week to a single day while maintaining or surpassing the pruning performance achieved by previous approaches.

4.2 Future Work

The results presented in Section 2.1 represent an original application of computational semiotics in the analysis and interpretation of deep neural networks, which, to the best of our knowledge, has not been done before. In future work, it would be beneficial to expand our analysis beyond the sole focus on convolutional neural networks (CNNs), which are primarily used for image processing. While our current study concentrated on CNNs due to their relevance, it is important to explore the connections with other domains such as audio and text, as well as investigate different architecture types like recurrent neural networks. Additionally, the semiotic approach we employed can be extended to various deep learning models, as the concept of semiotic superization appears to be present in many architectures. Overall, the computational semiotics framework shows promise in contributing to the explanation and optimization of deep networks, particularly in cases where multiple levels of superization are involved.

The contributions presented in Section 2.2 can be divided into two parts. Firstly, a connection is established between the IB hypothesis of fitting and compression and semiotic superization by examining the evolution of spatial entropy in saliency maps. Secondly, a heuristic training strategy for layer-wise early stopping is designed based on the variability of spatial entropy over time. This strategy can be employed practically to prevent overfitting during the learning process. To enhance the validity of our findings, further experiments are necessary. These forthcoming experiments should involve a broader spectrum of DNN architectures, explore the practical applications of spatial entropy variability over time in more depth, and contribute to a more comprehensive theoretical comprehension of the phenomena investigated in this study.

The original contribution presented in Section 3.1 entails the introduction of a novel

approach to neural network pruning within an existing AutoML framework. Within this context, an optimization reward criterion is proposed, centered around the minimization of spatial entropy at each convolutional layer. Extensive empirical experiments establish the effectiveness of this entropy minimization strategy in maintaining accuracy levels. The significance of this work lies in the exploration of alternative and more principled methods for neural network pruning, departing from the traditional approach of directly optimizing the agent's reward function based on accuracy. In future research, our focus will be on exploring innovative approaches for utilizing entropy in the optimization of neural architectures. One potential avenue is considering the entropy measure as a heuristic for selecting preserved channels, instead of relying solely on the commonly used L2 magnitude criterion. Additionally, we aim to investigate whether a connection exists between entropy-based pruning and semiotic aggregation. These directions represent promising areas to delve into and can provide further insights into the optimization and understanding of neural networks.

The work presented in Section 3.2 contributes to addressing the limitations of kernel-based mutual information estimation methods, including issues with kernel width selection, curse of dimensionality, and computational complexity. Building upon the HRel method that utilizes Rényi's alpha entropy estimator [86], a more efficient approach is proposed for computing MI specifically for filter importance selection. The proposed method significantly reduces the optimization time required, from nearly a week of compute time to just a single day. Based on the formulation of MI as defined in previous work [87], which utilizes spatial aura entropy, the proposed method offers a more efficient and straightforward alternative to kernel-based estimators without the need for kernel width selection. Notably, the results obtained by the proposed method maintain or even improve upon those achieved by the original work, while reducing computational costs to a great extent. Future work could focus on exploring the applicability of our method on other benchmark datasets and model architectures, as well as investigating its potential in other areas of machine learning and computer vision.

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.

[2] Assaf Almog and Erez Shmueli. Structural entropy: monitoring correlation-based networks over time with application to financial markets. *Scientific reports*, 9(1):1–13, 2019.

[3] Răzvan Andonie. A semiotic approach to hierarchical computer vision. In J. Ross, editor, *Cybernetics and Systems (Proceedings of the Seventh International Congress of Cybernetics and Systems, London, Sept. 7-11, 1987)*, pages 930–933, Lytham St. Annes, U.K., 1987. Thales Publication.

[4] Răzvan Andonie. Semiotic aggregation in computer vision. *Revue roumaine de linguistique, Cahiers de linguistique théorique et appliquée*, 24:103–107, 1987.

[5] YM Asano, C Rupprecht, and A Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *International Conference on Learning Representations*, 2020.

[6] Roland Barthes. *Image, Music, Text*. Hill and Wang, 1977.

[7] David Bau, Bolei Zhu, Hendrik Strobelt, Bo Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6541–6549, 2017.

[8] Max Bense. *Semiotische Prozesse und Systeme in Wissenschaftstheorie und Design, Ästhetik und Mathematik*. Agis-Verlag, Baden-Baden, 1975.

[9] Ekaba Bisong. *Google Colaboratory*. Apress, Berkeley, CA, 2019.

[10] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning?, 2020.

[11] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, ..., and Xi Zhang. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[12] Norman Bryson. *Vision and Painting: The Logic of the Gaze*. Yale University Press, 1994.

[13] Daniel Chandler. *Semiotics: The basics*. Taylor & Francis, 2017.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[15] Xiaotong Chen, Li Liu, Jing Yang, Chang Yang, and Wangmeng Zuo. Semantics-assisted interpretation of deep neural networks for visual recognition. *Pattern Recognition*, 104:107312, 2020.

[16] Yu Cheng, Chen Gao, Guoqiang Xu, and Xin Lu. Exploration of information in deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–9, 2015.

[17] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282, 2017.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2012.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[20] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2018.

[22] Umberto Eco. *A Theory of Semiotics*. Indiana University Press, 1976.

[23] Jiashi Feng, Junzhou Huang, Ivan Laptev, and Shuicheng Wang. Semantic adversarial deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11649–11658, 2020.

[24] Helmar Frank. *Kybernetische Grundlagen der Pädagogik: eine Einführung in die Informationspsychologie und ihre philosophischen, mathematischen und physiologischen Grundlagen*. Agis-Verlag, Baden-Baden, second edition, 1969.

[25] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[26] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks, 2019.

[27] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[28] Bernhard C. Geiger. On information plane analyses of neural network classifiers - a review. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2021.

[29] Robert Geirhos, Dann Janssen, Heiko H Schütt, Jonas Rauber, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(10):665–673, 2020.

[30] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *ArXiv*, abs/2103.13630, 2022.

[31] Yoav Goldberg and Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.

[32] Ziv Goldfeld and Yury Polyanskiy. The information bottleneck problem and its applications in machine learning. *CoRR*, abs/2004.14941, 2020.

[33] Antônio Gomes, Ricardo Gudwin, and João Queiroz. Towards meaning processes in computers from peircean semiotics. *SEED Journal–Semiotics, Evolution, Energy, and Development*, 3(2):69–79, 2003.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[35] Algirdas Julien Greimas. *Structural Semantics: An Attempt at a Method*. University of Nebraska Press, 1983.

[36] Ricardo Gudwin and Fernando Gomide. A computational semiotics approach for soft computing. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 4, pages 3981–3986. IEEE, 1997.

[37] Ricardo Gudwin and João Queiroz. Towards an introduction to computational semiotics. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 393–398. IEEE, 2005.

[38] Ricardo R Gudwin. Semiotic synthesis and semionic networks. *SEED Journal (Semiotics, Evolution, Energy, and Development)*, 2:55–83, 2002.

[39] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)*, 51(5):93, 2018.

[40] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[41] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993.

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

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 346–361, Cham, 2014. Springer International Publishing.

[44] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[45] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, 2018.

[46] Yihui He, Xiangyu Zhang, and Shaoqing Sun. Channel pruning for accelerating very deep neural networks. *International Conference on Computer Vision*, pages 1398–1406, 2017.

[47] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35. IEEE, 2016.

[48] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[49] A. G. Journel and C. V. Deutsch. Entropy and spatial disorder. *Mathematical Geology*, 25(3):329–355, 1993.

[50] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5454–5463, 2017.

[51] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[52] Alex Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, 2012.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[54] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[55] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[56] Hao Li, Asit Kadav, Ivana Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *Proceedings of the International Conference on Learning Representations*, 2017.

[57] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178, 2004.

[58] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[59] Henry W Lin and Max Tegmark. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1705.11029*, 2017.

[60] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[61] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumian Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.

[62] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations*, 2019.

[63] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[64] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[65] Ali Mahdi, Jing Qin, and George Crosby. DeepFeat: A bottom-up and top-down saliency model based on deep features of convolutional neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 12(1):54–63, 2020.

[66] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783*, 2016.

[67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[68] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):1–11, 2018.

[69] C.W. Morris and M. Charles William. *Writings on the General Theory of Signs*. Approaches to semiotics. Mouton, 1972.

[70] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[71] Bogdan Musat and Razvan Andonie. Accelerating convolutional neural network pruning via spatial aura entropy. In *To be published in the proceedings of the 27 International Conference Information Visualisation*, 2023.

[72] Bogdan Musat and Razvan Andonie. Pruning convolutional filters via reinforcement learning with entropy minimization. In *To be published in the proceedings of the 22nd International Conference on Artificial Intelligence and Soft Computing*, 2023.

[73] Bogdan Muşat and Răzvan Andonie. Semiotic aggregation in deep learning. *Entropy*, 22(12), 2020.

[74] Hai Nguyen and Hung La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 590–595, 2019.

[75] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Adolfo Velasco-Hernández, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. *CoRR*, abs/1910.13796, 2019.

[76] Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *CoRR*, abs/1902.04674, 2019.

[77] Samet Oymak and Mahdi Soltanolkotabi. Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1:84–105, 2020.

[78] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8026–8037, 2019.

[79] Charles S. Peirce. *Collected papers of Charles Sanders Peirce*, volume 2. Harvard University Press, 1960.

[80] Harry A. Pierson and Michael S. Gashler. Deep learning in robotics: A review of recent research. 2017.

[81] Q. R. Razlighi, N. Kehtarnavaz, and A. Nosratinia. Computation of image spatial entropy using Quadrilateral Markov Random Field. *IEEE Transactions on Image Processing*, 18:2629–2639, 2009.

[82] Qolamreza Razlighi and Nasser Kehtarnavaz. Fast computation methods for estimation of image spatial entropy. *J. Real-Time Image Processing*, 6:137–142, 06 2011.

[83] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA, 2015. MIT Press.

[84] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[85] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[86] C.H. Sarvani, Mrinmoy Ghorai, Shiv Ram Dubey, and S.H. Shabbeer Basha. Hrel: Filter pruning based on high relevance between activation maps and class labels. *Neural Networks*, 147:186–197, 2022.

[87] Bogdan Mușat and Răzvan Andonie. Information bottleneck in deep learning - a semiotic approach. *International Journal of Computers Communications & Control*, 17(1), 2022.

[88] Ferdinand de Saussure. *Course in General Linguistics*. Philosophical Library, New York, NY, 1916.

[89] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[90] T.A. Sebeok. *Signs: An Introduction to Semiotics*. Toronto Studies in Semiotics. University of Toronto Press, 1994.

[91] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization. In *CoRR*, volume abs/1610.02391, 2016.

[92] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, 19:221–248, 2017.

[93] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017.

[94] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[95] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 387–395, 2014.

[96] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[97] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[98] Dominic Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.

[99] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

[100] Ioan Stan and Răzvan Andonie. Cybernetical model of the artist-consumer relationship (in Romanian). *Studia Universitatis Babes-Bolyai*, 2:9–15, 1977.

[101] Chengyue Sun, Liang Wang, Xiaodong Liu, and Jingping Shi. Patient knowledge distillation for bert model compression. 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 3873–3879, 2019.

[102] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (2nd Edition)*. MIT Press, 2018.

[103] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *CoRR*, abs/1703.09039, 2017.

[104] C Szegedy, W Liu, Y Jia, P Sermanet, SE Reed, D Anguelov, D Erhan, V Vanhoucke, and A Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[105] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[106] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708. IEEE, 2014.

[107] M Tan and QV Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[108] Kumiko Tanaka-Ishii. *Semiotics of Programming*. Cambridge University Press, USA, 1st edition, 2010.

[109] Nicolás I. Tapia and Pablo A. Estévez. On the information plane of autoencoders. *CoRR*, abs/2005.07783, 2020.

[110] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

[111] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015.

[112] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, ..., and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[113] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[114] E. Volden, G. Giraudon, and M. Berthod. Modelling image redundancy. In *1995 International Geoscience and Remote Sensing Symposium, IGARSS '95. Quantitative Remote Sensing for Science and Applications*, volume 3, pages 2148–2150, 1995.

[115] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018:1–13, 02 2018.

[116] Xiaojie Wang, Chaoqun Wang, Xilin Chen, and Liqing Zhang. Symbolic adversarial learning. *Pattern Recognition*, 103:107260, 2020.

[117] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14913–14922, June 2021.

[118] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[119] Kristoffer Wickstrøm, Sigurd Løkse, Michael Kampffmeyer, Shujian Yu, Jose Principe, and Robert Jenssen. Information plane analysis of deep neural networks via matrix-based Renyi's entropy and tensor kernels. 2019.

[120] Judith Williamson. *Decoding Advertisements: Ideology and Meaning in Advertising*. Marion Boyars Publishers, 1978.

[121] Andrew KC Wong and Mark A Vogel. Resolution-dependent information measures for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(1):49–61, 1977.

[122] Xiaofei Wu, Rongrong He, Zhenan Sun, and Tieniu Tan. A survey of compressing deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(3):705–723, 2021.

[123] Yinchong Yang, Zhenqiang Li, Xiaomeng Song, Chenghao Liu, Junhao Hou, and Lianli Gao. Interpretable convolutional neural networks. *arXiv preprint arXiv:1911.02508*, 2019.

[124] Quanming Yao, Mengshuo Wang, Hugo Jair Escalante, Isabelle Guyon, Yi-Qi Hu, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306, 2018.

[125] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. 2018.

[126] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *CoRR*, volume abs/1311.2901, 2013.

[127] Heinz Zemanek. Semiotics and programming languages. *Communications of the ACM*, 9(3):139–143, 1966.

[128] Qinglong Zhang, Yifan Zhu, and Lei Zhang. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[129] Qinglong Zhang, Yifan Zhu, and Lei Zhang. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 818–833, 2018.

[130] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.