Graph to sequence syntactic pattern recognition for image classification problems

Gilberto Astolfi

Graph to sequence syntactic pattern recognition for image classification problems

Gilberto Astolfi

Thesis advisor: *Hemerson Pistori*

Thesis submitted to the Computer Science Graduate Program in partial fulfillment of the requirements for the Degree of Doctor in Computer Science, College of Computing, Federal University of Mato Grosso do Sul (UFMS).

UFMS - Campo Grande
June/2021

Acknowledgements

Thanks to all my family for supporting me at all times, especially my mum Diomar Pereira Astolfi, my dad Benedito Astolfi (*in memoriam*), and my wife, Angelica Christina Melo Nunes Astolfi.

To my friends and colleagues from INOVISAO Lab, for great collaboration and friendship.

To my advisor, professor Hemerson Pistori for beliving in me.

To my co-advisor, professor Edson Takashi Matsubara for beliving in me.

To many other people who I have come across along the way, including family, professors, colleagues, and friends. Thank you.

Abstract

A growing interest in applying Natural Language Processing (NLP) models in computer vision problems has recently emerged. This interest is motivated by the success of NLP models in tasks such as translation and text summarization. In this work, a new method for applying NLP to image classification problems is proposed. The aim is to represent the visual patterns of objects using a sequence of alphabet symbols and then train some form of Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), or Transformer using these sequences to classify objects. The visual pattern representation of objects in a syntactic way allows PLN models to be applied to image classification problems in a natural way, i.e., in the same way as applied to natural language problems. Two visual pattern representation approaches of objects in a syntactic way were investigated: representation using keypoints and representation using component parts of objects. In the approach that uses keypoints, the keypoints are identified in the images, associated with alphabet symbols, and then related using a graph to derive strings from images. Strings are the inputs for training an LSTM encoder. Experiments showed evidence that the syntactic pattern representation can represent visual variations in superpixel images captured by Unmanned Aerial Vehicles, even when there is a small set of images for training. In the approach that uses component parts of objects, the component parts are provided by means of bounding boxes in the images. The component parts are associated with alphabet symbols and related with each other to derive a sequence of symbols from the object for representing its visual pattern. Then, some form of GRU, LSTM, or Transformer are trained to learn the spatial relation between component parts of the objects contained in the sequences. An extensive experimental evaluation using a limited number of samples for training has been conducted to compare our method with ResNet-50 deep learning architecture. The results achieved by the proposed method overcome ResNet-50 in all test scenarios. In one test, the method presents an average accuracy of 95.3% against 89.9% of the ResNet-50. Both experiments showed evidence that from a finite set of primitive structures is possible to obtain many variations in the visual pattern of the object same when there are few samples for training. Besides, the experiments evidenced that the NPL models can be applied in a natural way to image classification problems in computer vision.

Keywords: Computer Vision, Natural Language Processing, Syntactic Pattern Representation, Graph.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The fundamental problem in computer vision since the 60s has been to recognize objects at the class level (Andreopoulos and Tsotsos, 2013). This problem consists of identifying the type of object that exists in a given image. To address this problem, for a long time, feature extraction algorithms, such as Scale Invariant Feature Transform (SIFT) (Lowe, 2004) and Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005), have been combined with shallow learning algorithms, such as Decision Trees (Hastie et al., 2001) or Support Vector Machines (SVM) (Luong et al., 2014), for extracting features and learning object patterns in images. Later, the deep Convolutional Neural Networks (CNN) appeared and represented a huge breakthrough in object recognition problems, even surpassing human performance in some cases (He et al., 2016).

In parallel, Natural Language Processing (NLP) also has made significant progress. In particular, language modelling (Bengio et al., 2001), which consists of predicting the next word in a text given the previous words, and sequence-to-sequence learning (Sutskever et al., 2014), which is a model that converts an input sequence into another output sequence. This progress allowed us to deal with the simplest tasks, such as spelling autocorrection and email response suggestions, to more complex tasks, such as machine translation. The great advancement in this area occurred with the introduction of Recurrent Neural Networks (Mikolov et al., 2010) and Long Short-Term Memory Networks (Greff et al., 2017) for language modeling, and attention mechanisms as the Transformer (Vaswani et al., 2017) for machine translation.

Although in different areas, computer vision and NPL have made significant progress after adopting neural networks to solve problems. This has motivated research to combine NLP and CNN models for building hybrid architectures to solve computer vision tasks. Dosovitskiy et al. (2021) proposed the Vision Transformer, a simple model that split an image into 16×16 patches for providing a patch sequences and inputting it into a standard Transformer. Touvron et al. (2021) improved the proposed model by Dosovitskiy et al. (2021) by adding Knowledge Distillation (Hinton et al., 2015) and class tokens along with the Vision Transformer patches. Chen et al. (2020b) proposed a model trained in an unsupervised way that pre-processes images to reduce resolution and color space for reshaping it into a 1-dimensional pixel sequence for then applying standard Transformer to pixel sequence. Wu et al. (2020) applied convolutional layers to obtain image features and build tokens for inputting them into a standard Transformer. These approaches try to find patterns at the pixel level, such as CNN-based models, and not at the level of high-level representative information, such as decomposable parts of objects. Thus, like the CNN-based models (Gu et al., 2018), these hybrid models need large amounts of data for training to have a good generalization (Dosovitskiy et al., 2021).

Addressing object recognition problems requires dealing with differences between objects of the same type. For example, cats can have different breeds, colors, and sizes. Besides, objects in images can have poses and scales variation, poor illumination conditions, occlusions, among other variations. Considering these variations in the visual of the object, the number of images vision computer approaches have to deal with is almost infinite.

An alternative is to treat the object structure from a compositional viewpoint. The central idea is that objects are composed of primitive structures that, when joined together, form a visual pattern (Bienenstock et al., 1997). This suggests that from a finite set of primitive structures, it is possible to obtain many variations in the visual pattern of the object, allowing to extrapolate the visual variations of it beyond the samples offered for training. Thus, the sensitivity of the object recognition method is not only limited to the training set.

The representation of the visual pattern of objects in a compositional way has been a problem addressed since the 70s (Fu, 1974). It is a representation strategy in which the visual pattern of an object is treated from a compositional perspective, where a pattern is composed of simpler sub-patterns, and the most basic sub-patterns are known as primitives (de Souza Pio et al., 2006). However, some issues must be considered when adopting this representation strategy. The main one is how to define and represent primitives, as well as defining a set of rules for representing the visual patterns formed by the interrelation between primitives. The representation of primitives is determined by basic features extraction from objects, while the rules are defined by the visual pattern complexity that is intended to be represented.

The strategy used in this work is to represent visual patterns of objects using a syntactic approach. The syntactic approach naturally has the ability to represent visual patterns of objects in a compositional way, considering logical or probabilistic rules of composition, addressing the strategy of the simple to the complex (de Souza Pio et al., 2006).

The first issue to be addressed is to represent primitives syntactically. The first step is to define what will be considered a primitive structure of an object. Jiang and Ma (2015), for example, treat body parts (head, arm, legs, etc.) as primitives. Martinovic and Van Gool (2013) consider geometric figures that compose building facades such as windows, doors, balconies as primitives. In this work, the possibility of using two types of primitives is investigated: object keypoints and component parts of objects. The keypoints are computed by feature extractors such as SIFT. They are invariant to rotation and scale (Lowe, 2004). By having this property, they can contribute positively to the visual variations of objects problem. On the other hand, component parts of objects, which are high-level primitives, are the parts of a given object we look at to identify it, i.e., object parts that attract attention. For example, the component parts of an insect could be the head, body, paw, etc. Both types of primitives are represented by associating alphabet symbols to them. With this strategy, object primitives can be represented using a syntactic approach.

However, isolated primitives cannot represent visual patterns of objects or part of them. Then, for representing the visual pattern of objects, primitives must be related respecting composition rules. Thus, a visual pattern of an object is represented by the interrelation between primitives, forming small sequences of alphabet symbols.

In the same way, a single visual pattern cannot represent visual variations of objects. For representing a visual variation of an object, visual patterns must also be related in a compositional way. Thus, a visual variation of an object can be represented by a large sequence of symbols composed of a sum of smaller sequences, i.e., a sum of visual patterns. Consequently, the visual variations of the object, such as pose variations, deformations, occlusion, among others, can be represented by few variations in the sequence of symbols that represent it. This representation strategy might increase the generalization capacity and allows an unlimited number of visual variations of an object to be obtained from a few combinations of small sequences of symbols. This can allow the visual variations of an object to be extrapolated beyond the samples from the training set, giving better conditions to deal with pose variations, deformation, and different object perspectives.

This constructive strategy, in which primitives are represented by alphabet symbols, visual patterns by small sequences of symbols, and visual variations of objects by combining these small sequences of symbols, becomes the pattern recognition problem in images, usually performed in a discriminative way, a pattern recognition problem in symbol sequences guided by descriptive and composition rules. This allows this work to raise the following hypothesis: Natural Language Processing models generalize from a few samples in image classification tasks when trained with visual patterns of objects represented in a syntactic way.

Unlike Dosovitskiy et al. (2021), Touvron et al. (2021), Chen et al. (2020b), and Wu et al. (2020), which apply PLN directly to images to find patterns at the pixel level, this work applies PLN to find patterns in sequences of symbols used for representing high-level visual patterns of objects. Thus, PLN models are applied to image classification problems in the same way as they are applied to sentences.

1.1 Objective

The objective of this work is to represent visual patterns of objects in a syntactic way for applying natural language processing models to image classification problems.

For this, two strategies to represent visual patterns of objects in a syntactical way are adopted. In the first, keypoints are identified in images and associated with alphabet symbols. Then, the keypoints in each image are related to derive a sequence of symbols. In the second, component parts of objects are provided by means of bounding boxes in the images. The component parts of the objects in each image are associated with alphabet symbols and related to also derive a sequence of symbols. In both strategies, the symbol sequences derived from the images encapsulate the structural interrelation between the primitives that compose the visual pattern of the object in the image. After representing the visual patterns of the objects in images using a syntactic approach, natural language processing models can be applied to classify the images within their object classes.

1.2 Motivation

The visual representation of an object in an image is analogous to the composition of a sentence. An object is composed of visual patterns formed by the interrelation between primitives. A sentence is composed of words formed by alphabet symbols. This similarity motivates this work to try to replicate the success of natural language processing models in image classification problems.

1.3 Contributions

The main contributions of this thesis are as follows:

1. A survey of current studies in the area of syntactic pattern recognition in images and videos by means of a systematic literature review.

2. New strategies to represent visual patterns of objects in a compositional way by means of a syntactic approach.

3. A new approach for applying natural language processing to image classification problems.

1.4 Thesis Outline

The remainder of this thesis consists of five other chapters. Chapter 2 gives an overview of some concepts used in Chapters 4 and 5. In Chapter 3 the paper "Syntactic Pattern Recognition in Computer Vision: A Systematic Review" published in the ACM Computing Surveys journal is presented. This paper surveys the most relevant studies that use syntactic approaches for pattern recognition tasks in images and videos. Moreover, this survey has revealed gaps in the syntactic pattern recognition in images exposed by this work.

In Chapter 4 the paper "Combining Syntactic Methods with LSTM to Classify Soybean Aerial Images" published in the IEEE Geoscience and Remote Sensing Letters journal is presented. The paper introduces an approach that extracts keypoints of images and treats them as primitives. The keypoints are associated with alphabet symbols and connected into a graph to derive a sequence of symbols from each image. Then, a Long Short-Term Memory (LSTM) (Greff et al., 2017) is used as a classifier to learn the relationship between the symbols in sequences. An extensive experimental evaluation using aerial images from a soybean field captured by Unmanned Aerial Vehicles has been conducted to evaluate the approach on small datasets. The approach showed competitive results when there is only a limited number of samples for training.

In Chapter 5 the paper "A New Approach for Applying Natural Language Processing to Image Classification Problems" recently submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence is presented. The paper introduces a hybrid approach that handles component parts of objects as primitives. The objects are decomposed into component parts and share common parts with each other. The component parts of the object are associated with alphabet symbols and connected to derive a sequence of symbols from each image. Then, NLP models are trained using the sequence of symbols. The aim is to classify objects from the composition of their main parts encapsulate in the sequence of symbols. When using this strategy, we are extrapolating object appearance (partial occlusion, different poses, etc.) beyond the samples that were offered for training. This approach also showed competitive results when trained with a limited number of samples.

Finally, Chapter 6 presents the final considerations and future works. Also attached to this thesis is a list of technical and scientific production performed during the research period (see Appendices A). The production is about topics related to object detection and classification whose the achieved results supported the development of this thesis.

CHAPTER 2

Background

In this Chapter, the major concepts used in this work are presented. Section 2.1 introduces the syntactic pattern representation applied to computer vision. Section 2.2 introduces the SIFT, a feature extractor used to extract primitives of objects in the paper presented in Chapter 4. Section 2.3 introduces the Simple Linear Iterative Clustering (SLIC) Superpixels, a segmentation algorithm used to limit the object scope contained in images. SLIC is used in the paper presented in Chapter 4 to build the dataset. Sections 2.4 and 2.5 provide an overview of NLP models used in the papers in Chapter 4 and Chapter 5.

2.1 Syntactic Pattern Representation

Patterns are detectable regularities that repeat predictably and can be understood as the means by which the environment around us can be interpreted (Fu, 1982). Pattern recognition for humans is a trivial task, but by and large, it becomes a complex problem when we try to perform it artificially. The pattern recognition area focuses on discovering significant regularities in data in order to use it to describe or classify data in different categories (Bishop, 2006).

The regularities in the data are represented as measurable features, which can be numeric and/or non-numeric, and their relationships. The composition formed by these features defines the pattern from the dataset (Fu and Rosenfeld, 1976). However, it is not enough to define a representative standard for the data, a careful analysis of the pattern in order to represent it

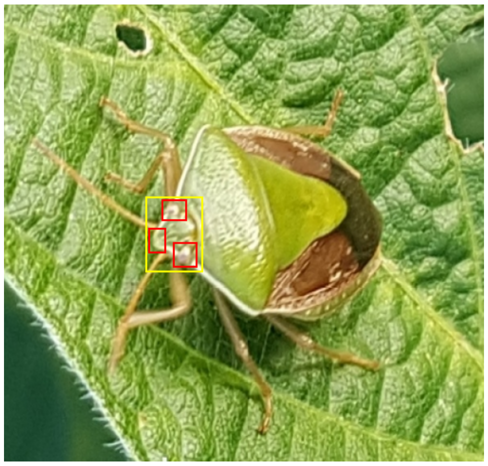

Figure 2.1: An illustration of the visual patterns composition in an insect image. The yellow rectangle illustrates the pattern/sub-pattern, and the red rectangles illustrate the primitives.

in a way interpretable to algorithms is necessary.

There are several approaches for representing patterns, such as constellations (Fergus et al., 2003), *quadre* structures (Pedro, 2013), and *And-Or* graphs (Zhu L. Chen, 2009). However, the representation generally used is feature vectors (Goodfellow et al., 2016), and the pattern recognition is usually done by similarity comparison (Fu and Rosenfeld, 1976; Bishop, 2006).

The syntactic approach to pattern representation deals with patterns from a compositional perspective. A complex pattern is composed of simpler sub-patterns, which are composed of simpler ones. At the lowest level of this compositionality are the basic, non-divisible patterns, which are called primitives (Fu and Rosenfeld, 1976). In Figure 2.1 , an example of visual pattern composition is presented. The yellow rectangles illustrate the patterns/sub-patterns, and the red rectangles illustrate the primitives. Primitives are usually used to represent corners, outlines, lines, or textures. On the other hand, patterns/sub-patterns are used to represent structures that shape perceivable visual patterns. This compositional model of pattern representation allows repeating primitives and sub-patterns in different visual patterns, enabling many visual patterns representation from a finite set of data. In addition, it describes how the visual pattern was generated.

Similar to a language, primitives are like letters in an alphabet, basic sub-patterns are like words, and complex patterns, which shape visual patterns, are like sentences. In this way, a letter can be associated with an image primitive in order to represent it, as well as a set of letters can be associated with basic sub-patterns and so on.

The syntactic approach to pattern representation allows the pattern recognition problem in images, usually performed by the similarity between feature vectors, to be treated as a pattern recognition problem in character sequences. Thus, a visual pattern of an object can be recognized as belonging to a category using natural language processing models.

2.2 Feature Descriptor SIFT

The central idea around the SIFT (Lowe, 2004) is based on detecting salient and stable points in an image. These points are called keypoints and provide features that describe a small region in the image. The region described by the keypoint is a small circular region with an orientation and invariant to rotation and scale. The key point is described using four parameters: the x and y coordinates of the keypoint center, the scale (the region radius), the orientation (an angle defined over the radius), and a feature descriptor. In Figure 2.2, an example of SIFT operation in a soybean disease image is shown. SIFT consists of two main parts: detector and feature descriptor.

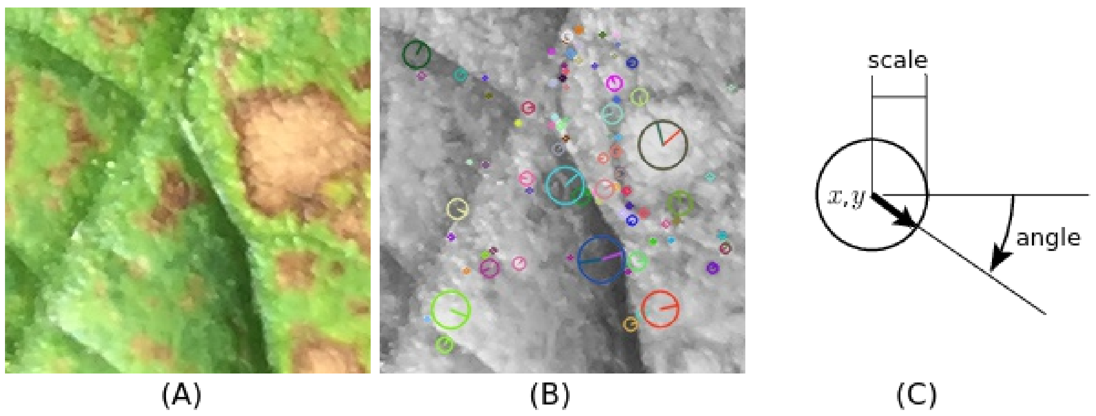

Figure 2.2: SIFT operation in a soybean disease image. (A) Soybean disease original image, (B) keypoints obtained by the SIFT, and (C) keypoints representation no description.

The keypoint detector is based on the Difference of Gaussian (DoG) calculations, and the descriptor uses gradient-oriented histograms to describe the neighborhood around the keypoint (Lowe, 2004). To detect and describe a keypoint, SIFT applies the following basic steps:

1. First, a scale-space using DoG is estimated. The aim is to identify keypoints candidates invariant to scale.
2. Second, each candidate keypoint is analyzed, and keypoints with low contrast are discarded (Oyallon and Rabin, 2015).
3. Third, an orientation is assigned to the selected keypoints based on the directions of

the image gradient at the keypoint location. This orientation is used later on to build descriptors invariant to rotation.

4. Finally, a 128-dimension local descriptor is calculated for each keypoint based on the orientation and gradient magnitude of the image in the keypoint region.

2.3 Simple Linear Iterative Clustering (SLIC) Superpixels

A superpixel is a region of an image formed by pixels that share similar color information or grayscale (Achanta et al., 2010). Generally, a superpixel provides a primitive from the image where local features can be obtained using feature extractors such as SIFT.

Superpixels are obtained by means of algorithms that aim to cluster similar pixels in atomic regions in the image. One of these algorithms is the Simple Linear Iterative Clustering (SLIC) Superpixels proposed by Achanta et al. (2010). SLIC clusters pixels based on the color similarities and the spatial proximity in the image. The SLIC receives a k value as a parameter, which corresponds to the amount of superpixel obtained from a given image with approximately equal sizes. To obtain the superpixels, SLIC performs the following steps (Achanta et al., 2010):

1. Initially, the input image is converted to the CIELAB color space;

2. After, a total of k^2 initial cluster centers $C_i = [l_i \;\; a_i \;\; b_i \;\; x_i \;\; y_i]^T$ are arranged in a regular mesh spaced at $S = \sqrt{N/k}$ separate pixels. N is the pixel number of the image;

3. Each pixel is associated with the nearest cluster center according to a distance measure D, considering only the centers whose region of $2S \times 2S$ pixels overlaps its location;

4. After, an update step adjusts the cluster centers to a mean vector $[l \;\; a \;\; b \;\; x \;\; y]^T$ of all pixels that belong to the cluster;

5. Steps 3 and 4 are repeated for a total of 10 iterations;

6. Finally, a post-processing step forces the connectivity between some disjoint pixels, which do not belong to the same connected component, to merge with an adjacent superpixel component.

The distance measurement D is performed as follow:

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \tag{2.1}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{2.2}$$

$$D_S = {d_{lab}}^2 + \left(\frac{d_{xy}}{S}\right)^2 m^2 \tag{2.3}$$

where m gives the relative importance between d_{lab} color dissimilarity and spatial distance d_{xy}. When m is a high value, the resulting superpixels are more compact. On the other hand, when m has a low value, the superpixels are irregular in size and shape. Figure 2.3 shows an example of applying the SLIC algorithm to a soybean disease image. In the left image, SLIC was executed with $k = 64$ and in the right image with $k = 256$.

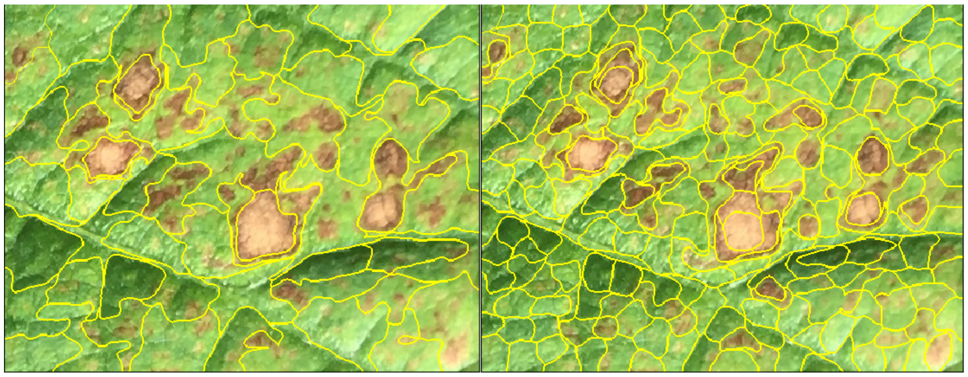

Figure 2.3: A soybean disease image segmented into 64 and 256 superpixels using the SLIC Superpixels algorithm.

2.4 Transformer

The sequence-to-sequence architecture was introduced by Sutskever et al. (2014) aiming to transform a data input sequence to a new one, respectively, source and target sequence. The sequence-to-sequence architecture is composed of an encoder-decoder model. Both the encoder and decoder are normally a Recurrent Neural Networks (RNN) (Chung et al., 2014) or a Transformer (Vaswani et al., 2017).

In Figure 2.4, an encoder-decoder model applied to a translation task is shown. The encoder receives sequential data as input, known as word embedding, and compresses them into a fixed length context vector. The decoder receives the context vector as input and emits as output the modified sequential data.

The Transformer (Vaswani et al., 2017) is a deep learning model that employs the mechanism of attention, whose objective is to give relevance to different parts of the sequential data input.

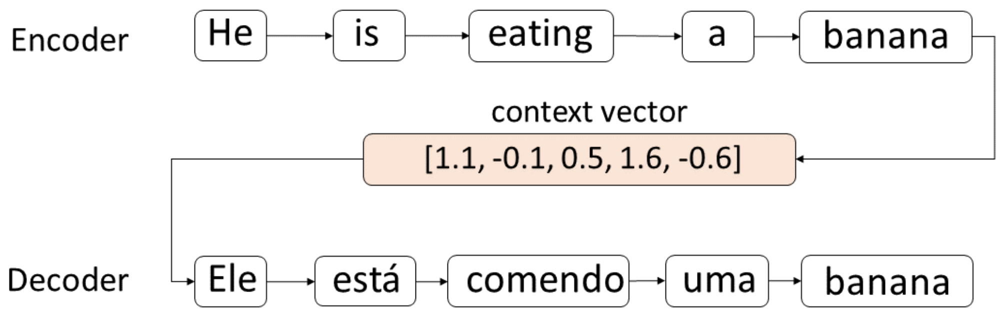

Figure 2.4: The encoder-decoder model, transforming a data input sequence “he is eating a banana” to a new one output sequence "ele está comendo uma banana".

Transformer architecture is based on the encoder-decoder model and is designed to deal with sequential data (see Figure 2.5). The encoder is composed of encoding layers that process data input through the layers iteratively. The decoder consists of decoding layers that receive the encoder output as input and produce a modified sequential data as output.

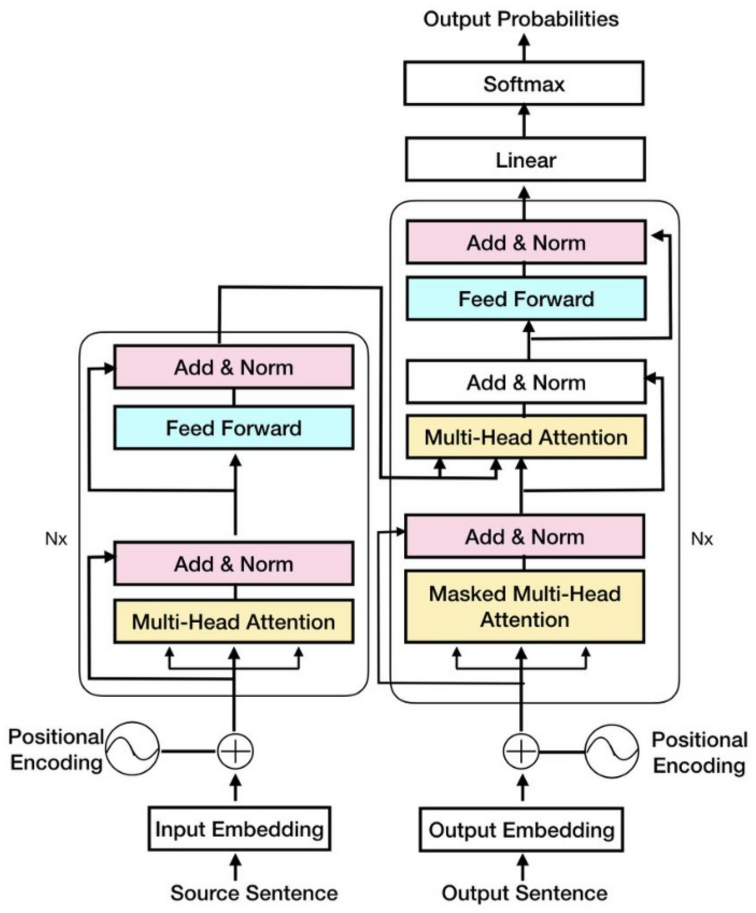

Figure 2.5: Transformer architecture by Vaswani et al. (2017). Left image is the encoder and right image is the decoder.

The encoder receives a data sequence as input processed by the Positional Encoding module. This module is in charge of adding within the vector that represents each entry in the sequence information about its position in the data sequence. Each encoder layer consists of two main components: the self-attention module and the feed-forward neural network. The self-attention module receives entries from the previous encoder layer. For each entry X, the self-attention layer generates a weight vector Z that represents the relevance for X in relation to all others entries in the data sequence. Z is generated by means of a scalar product of the vector that represents the entry X with the vectors that represent all others entry in the data sequence. The next step in the self-attention layer is to calculate the attention score corresponding to entry X. The attention score has a size of n, where n is the number of entries in the data sequence, and each element of this attention score is a value that tells how much a given entry in the data sequence is important (in terms of co-relation) to the current entry X. The attention score is calculated by multiplying the vector that represents each entry in the data sequence by the attention score and summing up the weight vector Z, resulting in attention/context vector for entry X. This process is made to all entries in the data sequence. The outputs of the self-attention layer are sent to a feed-forward neural network. The feed-forward neural network generates an output for each input attention/context vector, which is passed into the next self-attention layer of the encoder and so on.

The decoder has the same layers as the encoder, with an attention layer between them. The attention layer allows the decoder to focus on the relevant parts of the input sequence. The decoder receives the encoder output as input, i.e., the processed vectors that represent each entry from the data sequence. The last decoder layer is followed by a linear transformation and a softmax layer to produce the probability output over the vocabulary to produce a modified data sequence as output.

2.5 Recurrent Neural Networks

The Recurrent Neural Networks (RNN) (Chung et al., 2014) receive as input not only the samples of current data but also information observed previously in time. Time here refers to a logical time that denotes an order in a sequence (E.g.: word sequence). Thus, RNNs build their pattern recognition model by combining two input sources, the present and the recent past. This is possible because the RNNs have a *feedback loop* that connects to relevant information observed in the past. The *feedback loop* causes the network to receive its own output as one of its inputs, creating information feedback in order to build cycles. Each cycle stores relevant information over time in hidden states maintained by the network. In this way, at each time t, the network not only stores information from the data observed at t in its hidden state, but it also retrieves relevant information from the hidden state $t-1$ that happened in the past. This enables the network to create correlations between separate data over time, creating a dependency between them.

The hidden state represents a context vector. In general, the hidden state is passed from one stage to another in the network together with one new input from the sequence to produce one output and update the context vector, representing a new hidden state. The process continues until the last entry in the sequence is processed to produce a final hidden state, i.e., the final context vector. Figure 2.6 shows a simple RNN. In Figure 2.6 ("rolled network"), the neural network A receives X_t as input and processes the output h_t. The *feedback loop* allows information to pass from one cycle to the other in the network (see Figure 2.6, "unrolled network"). A standard RNN is like a chain of repeated neural network modules. These modules have a simple neural network layer structure, such as a single *tanh* layer. This chain architecture is intimately related to sequence processing, as words in a sentence or chain of symbols.

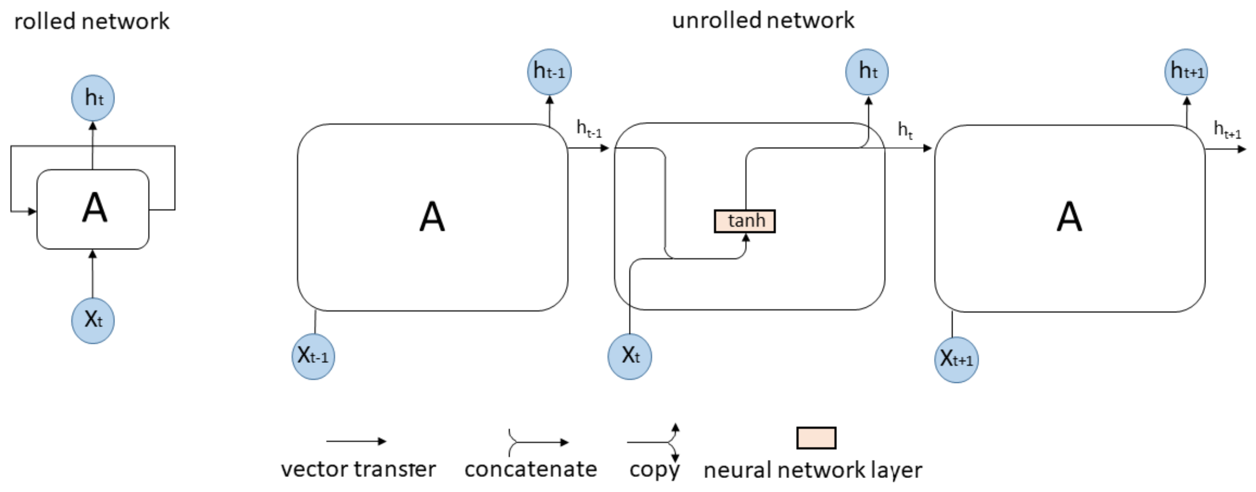

Figure 2.6: Summarized RNN (left) and detailed schematic of the RNN (right). An RNN is like a chain of repeated neural network modules. This simple RNN has a single *tanh* layer.

Although RNNs can create correlations between separate data over time, they have problems maintaining them over the long term. To solve this problem, the Long Short-Term Memory (LSTMs) (Greff et al., 2017) were proposed, introducing more complexity in the network module with four neural network layers (see Figure 2.7). LSTM is an RNN increased by a cell state at each time t, in which it is possible to write and read the information outside the normal flow created by the hidden states inherited from the RNN. This cell state is called short-term memory. With this, the LSTM has two different states passed between the cells: the cell state and the hidden state.

Access to writing and reading the cell state is controlled by units called gates, units composed of a *sigmoid* neural network layer and a pointwise multiplication operation. The decision of the gates to allow access to the cell state is made based on a set of weights computed by the network learning process.

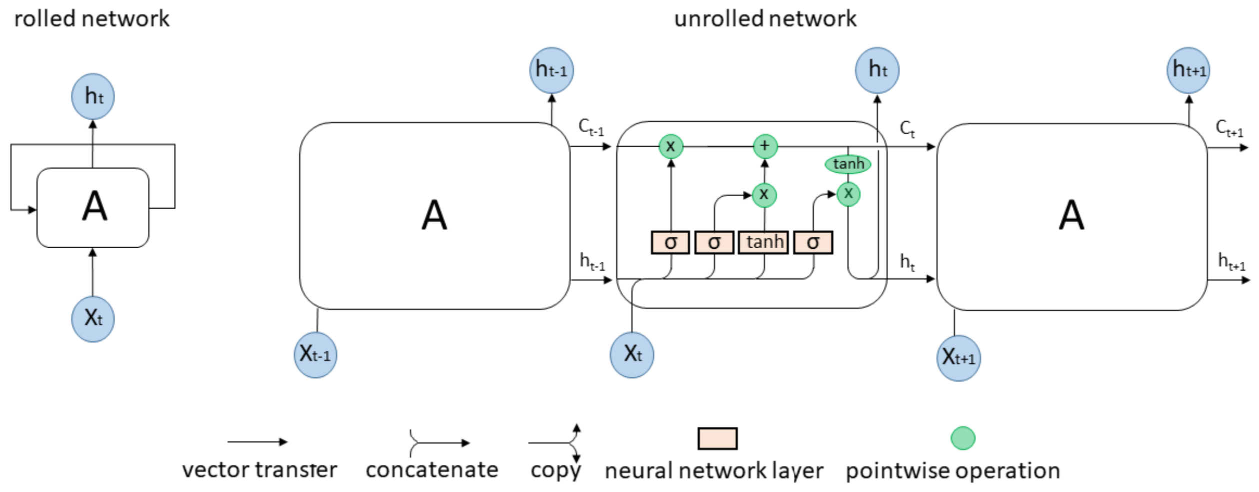

Figure 2.7: Summarized LSTM (left) and detailed schematic of the LSTM (right). An LSTM is like a chain of repeated neural network modules. In each LSTM module have four neural network layer.

The Gated Recurrent Unit (GRU) (Cho et al., 2014) is very similar to a LSTM. The difference is that in a GRU, the cell state is not maintained at each time *t*. Instead, the GRU uses the hidden state maintained by the network to transfer information from short-term memory through the network (see Figure 2.8). While the LSTM has two different states passed through cells, the cell state and the hidden state, which carry long and short term memory, respectively, GRUs have only one hidden state transferred between the steps of time. This hidden state is capable of maintaining long and short-term dependencies at the same time. This means that the GRU has few parameters to be trained compared to LSTM.

In addition to the standard LSTM, in this work, two others LSTM variations are used: Bidirectional Long Short-Term Memory (BLSTM) (Graves and Schmidhuber, 2005) and BLSTM with attention mechanism (Zhou et al., 2016). The BLSTM is structurally like an LSTM. The difference is that the BLSTM is not only connected to relevant information observed in the past but also in the future. The central idea of BLSTM is to process sequences forward and backward to two separate LSTMs, both of which are connected to the same output layer. This means that at each time *t*, the BLSTM stores information from past and future about the data observed

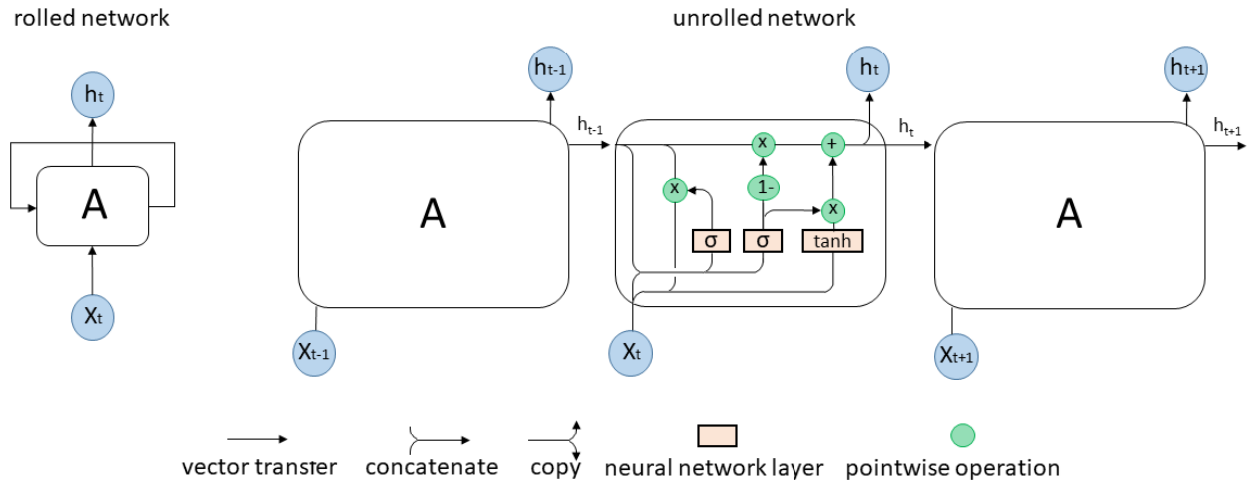

Figure 2.8: Summarized GRU (left) and detailed schematic of the GRU (right). A GRU is like a chain of repeated neural network modules. In each GRU module have three neural network layer.

in its hidden state and cell state to produce the final hidden state and represent the context vector. This enables the network to maintain long and short-term dependencies in both input directions. BLSTM with attention mechanism is a BLSTM increased by an attention block. In a BLSTM with an attention block, the context vector is represented by the final hidden state and by all hidden states produced at each time t. The attention block is in charge of calculating attention weights for each hidden state produced at each time t, indicating how much focus the data input at the time t should receive, i.e., how relevant it is in the input data sequence.

CHAPTER 3

Syntactic Pattern Recognition in Computer Vision: A Systematic Review

Authors[1]: Gilberto Astolfi, Fábio Prestes Rezende, João Vitor Porto, Edson Takashi Matsubara, and Hemerson Pistori.

Abstract: Using techniques derived from the syntactic methods for visual pattern recognition is not new and was much explored in the area called syntactical or structural pattern recognition. Syntactic methods have been useful because they are intuitively simple to understand, and have transparent, interpretable, and elegant representations. Their capacity to represent patterns in a semantic, hierarchical, compositional, spatial, and temporal way, have made them very popular in the research community. In this paper, we try to give an overview of how syntactic methods have been employed for computer vision tasks. We conduct a systematic literature review to survey the most relevant studies that use syntactic methods for pattern recognition tasks in images and videos. Our search returned 597 papers, of which 71 papers were selected for analysis. The results indicated that in most of the studies surveyed, the syntactic methods were used as a high-level structure that makes the hierarchical or semantic relationship among objects or actions to perform the most diverse tasks.

[1]Astolfi, G., Rezende, F. P. C., Porto, J. V. D. A., Matsubara, E. T., and Pistori, H. (2021). Syntactic pattern recognition in computer vision: A systematic review. *ACM Comput. Surv.*, 54(3). 13

3.1 Introduction

Computer vision aims to reproduce the human capacity to identify, interpret, and establish relationships between objects in images and scenes (Forsyth and Ponce, 2002). The interest in computer vision is due to a variety of applications in the real-world, from precision agriculture (Tetila et al., 2020) and environmental conservation (Jalal et al., 2020) to urban mapping (Santos et al., 2020) and surveillance (Tian et al., 2015). Its main tasks include image classification, in which the aim is to predict whether an object is in an image, object detection, which consists of predicting and locating objects in images, scene understanding, which elaborating an interpretation for a scene, and object tracking, whose aim is to detect and track objects in video.

Over the last years, computer vision has been dominate by Convolutional Neural Networks (CNNs) based approaches. Although these paradigms have achieved excellent results, some research efforts focus on developing approaches that address issues that remain open in the CNN paradigm, for instance, the hierarchical compositional representations of objects. Syntactic pattern recognition is usually applied to computer vision to address these issues, as they naturally are hierarchical and compositional approaches.

Syntactic pattern recognition have been widely used by the natural language processing community. However, from the 1960s, approaches that use syntactic methods to recognize patterns in images began to emerge (Eden, 1961; Narasimhan, 1962). For example, Kirsch (1964) introduced a formal descriptive mechanism, called Picture Description Language, to be an image description language. Since then, many approaches have been using syntactic methods to deal with pattern recognition in computer vision problems. Some approaches explore only syntactic methods (Pistori et al., 2013), others use hybrid models to combine syntactic methods and deep neural network architecture (Li et al., 2017) or combine syntactic methods with shallow learning algorithms (Zarchi et al., 2016). Regardless of how syntactic methods are being used, they are helpful in providing structural descriptions of objects in images, scenes, and videos. For this reason, several studies have used syntactic methods in recent years with promising results in computer vision and pattern recognition areas.

In this paper, we report the results of a comprehensive systematic literature review about syntactic pattern recognition in computer vision. Our review adopts a set of well-planned steps, defined by a previously established and documented protocol, taking into account the criteria set out by a systematic literature review (Kitchenham and Charters, 2007). Some literature reviews on syntactic methods in the computer vision area have been carried out by other researchers. Pedro et al. (2013) reported a systematic review of about 52 papers of which were analyzed the methods employed in each one. This review, which extends to 2013, had a greater emphasis on the use of grammars for pattern recognition in images. In a review that surveyed published papers prior to 2010, Chanda and Dellaert (2004) tried to show the most important grammatical methods applied to computer vision and pattern recognition, as well as their practical applications. Andreopoulos and Tsotsos (2013) dedicated part of her extensive literature review to analyze papers that use syntactic methods and graph representations on passive and active object recognition. Our systematic literature review surveyed 71 papers with publication date higher than the year 2012. Unlike Pedro et al. (2013), Chanda and Dellaert (2004), and Andreopoulos and Tsotsos (2013), we analyzed approaches that use syntactic methods on videos, since the video analysis has received tremendous attention in the computer vision community during the last years, mainly to represent spatio-temporal relation between current and previous video frames.

The remainder of this paper is organized as follows. In Section 3.2, we give an overview of the syntactic methods, and right after, in Section 3.3, we describe the methodology used in our systematic review. In Section 3.4, we presented and analyzed the syntactic method used by each surveyed paper. In Section 3.5, we discuss the results and point out possible future directions. Finally, in Section 3.6, we present conclusions about our systematic review.

3.2 A Brief Introduction of Syntactic Methods

The main idea behind syntactic pattern recognition in computer vision is to represent visual patterns in a structured way using, for example, strings, graphs, trees, or formal languages, and also recognize them in a structured way using, for example, formal language or automata. When the patterns are complex, they usually are composed of simpler subpatterns of which are composed of even simpler subpatterns. At the lowest level of this compositionality are the basic, non-divisible patterns that are called primitives, which usually are represented by an alphabet symbol (Fu and Rosenfeld, 1976). By making an analogy between syntactic pattern recognition and languages, we can say that the primitives are like the letters of an alphabet, the basic subpatterns are like words, and complex patterns, which represent the visual patterns, are like the sentences. Besides, grammar can be used to define the composition rules for a visual pattern in the same way as a language grammar defines the rules for building a sentence. The grammar can be designed manually when the structure of the pattern is well-defined or can be inferred from training data. In both cases, the grammar has the role of exclusively representing a class of patterns or objects (Jain et al., 2000).

When patterns are better represented using elementary features of the data instead of feature vectors, syntactic approaches, categorized as generative pattern recognition models, take precedence over discriminative models, especially when there are few examples available for training (Ng and Jordan, 2001). Therefore, for structurally-oriented pattern recognition tasks, whose generative models are the most appropriate, such as scene recognition, temporal video analysis, object tracking, spatial and structural object recognition, the syntactic recognition approaches tend to be better. Besides, in these types of tasks, the syntactic approaches not only classify patterns but also represent them in an interpretable and transparent way (Flasiński and Jurek, 2014).

One of the most important issues in the syntactic pattern recognition is the representation of primitives(Chanda and Dellaert, 2004). Some approaches, as we will see in the course of this review, represent primitives using the classic representation form that is by means of symbols of an alphabet, others use graph nodes, however, the choice of the representation is determined by the features extracted from the data. Another important issue is the representation and analysis of the patterns(Chanda and Dellaert, 2004). When the spatial relation between patterns is an important factor, shape grammars and split grammars are most commonly used. On the other hand, when the temporal relationship between the patterns is more important, hybrid models having stochastic characteristics are the most appropriate because they can get a probabilistic distribution over a set of valid compositions of the pattern over time. However, there is no specific syntactic approach for each type of problem, this makes the choice of approach independent. The key to syntactic pattern recognition is to follow a general set of steps (Flasiński and Jurek, 2014): identifying and representing primitives; represent the pattern in a structural and/or compositional way; infer a grammar or model an approach to pattern analysis. In this our systematic literature review, we observed that most of the surveyed studies employ hybrid models consisting of generative and discriminative approaches in any of these basic steps of pattern recognition.

3.3 Methodology for the Systematic Literature Review

The systematic review process has performed into three steps: planning, conduction, and data extraction (Kitchenham and Charters, 2007). In the planning step, we defined the research guidelines based on a protocol. First, we defined the research questions: "What methods are used to represent and recognize patterns in images using syntactic approaches?" and "What methods are used to infer grammars from identified patterns in images?". To compose the search strings to answer both questions, we used the keywords *image*, *grammar*, *computer vision*, *parsing*, *syntactic*, *inference*, *syntax analysis*, *grammar learning*, and *grammar model*. We defined the research databases IEEEXplore, ACM Digital Library, Web of Science, and Science Direct to carry out the searches. After this step, we defined the criterion for the inclusion and exclusion of papers. For a paper to be included in the review, it must use some syntactic method to represent or recognize patterns in images or videos. The papers returned by the search which do not meet this criterion are excluded from the review. We analyzed the title, abstract, and conclusion of each paper to verify if it meets the established inclusion criterion. Finally, as the last step in planning, we established that the papers that would be analyzed must have the publication date higher than the year 2012.

The first search by papers was carried out in August 2017. In December 2018 and January 2020, we carried out two other searches. In three searches, we used the same planning. From August 2017 to January 2020, we analyzed the papers surveyed to bring to light the main features of each study. Based on this analysis, we suggest possible future works that can be made using syntactic methods applied to computer vision.

3.4 Results and Analyses

3.4.1 Global Analysis

The searches returned 597 papers, of which 147 were outside the scope of the systematic review. In other words, they did not use syntactic methods in their approaches. These papers were returned in the searches because they cite other works that use syntactic methods or, in some ways, they mention syntactic methods in their text. Also, 383 papers published before the year 2013 were discarded. In this group, some papers used syntactic methods. However, they were not included in the review due to the protocol established in the planning phase. After applying the inclusion and exclusion criteria, we selected 67 of 597 papers for analysis. Moreover, four other papers were found and added to the review after the two initial searches, totaling 71 surveyed papers. In the systematic review performed by Pedro et al. (2013), which is similar to that presented by this paper, were selected 50 papers published over a period of twenty-seven years. It is evident from this that the syntactic methods are still explored by computer vision, even with the increasing use of techniques such as deep learning. In Figure 3.1, the number of publications per year can be observed among the surveyed and analyzed papers.

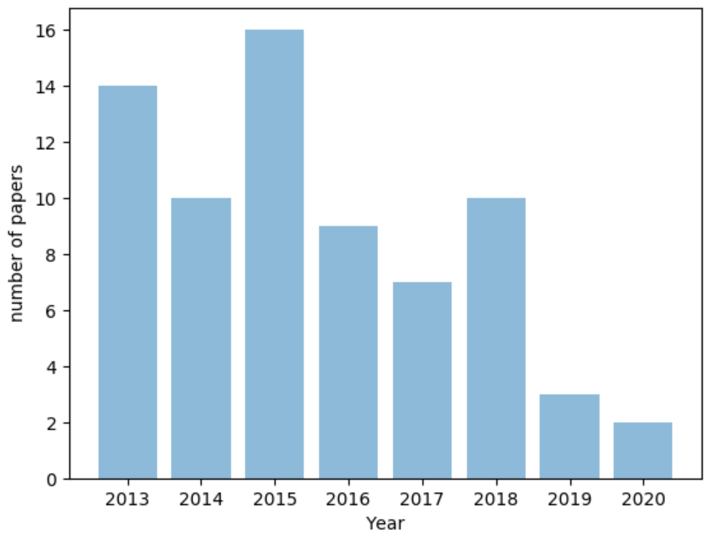

Figure 3.1: Number of papers published per year surveyed by our systematic review.

In the Figure 3.2 the syntactic methods used by the analyzed papers are showed. Each paper is categorized into only one syntactic method. However, in some cases, more than one syntactic method is addressed by the paper, such as Liu et al. (2018a) that combines Attribute grammar and Stochastic Context-free Grammar to track people in videos. In this case, the paper is categorized into the syntactic method that the authors put more emphasis on. It is observed in Figure 3.2 that the most explored syntactic method among all others was Stochastic Context-Free Grammar (28.2%). Context-free Grammar came in second (14.1%), followed by Shape Grammar (9.9%), Attribute Grammar (9.9%), and Markov Models (9.9%). In the appendix 3.7 is shown the summary of the 71 papers analyzed.

3.4.2 Methods for Syntactic Pattern Recognition

Syntactic pattern recognition applied to computer vision is used to decompose the appearances of objects or actions in videos into decomposable components. For example, a face can be decomposed into parts like eyes, nose, and mouth; and a video with human actions into subactions like standing, walking, and running. Usually, these components tend to occur in objects or videos more often and not in a random way, providing a basis for encoding formal models based on hierarchy and compositionality rules.

The syntactic pattern recognition is commonly performed by a grammar, which usually consists of four-tuple $G = (V_N, V_T, S_0, P)$, where V_N and V_T are terminal and non-terminal symbols respectively, and P are the production rules: $A \to \alpha$ where, $A \in V_N$ and $\alpha \in (V_N \cup V_T)^+$ (Wu, 2013). A language from the grammar G denotes the set of all strings that can be generated by produc-

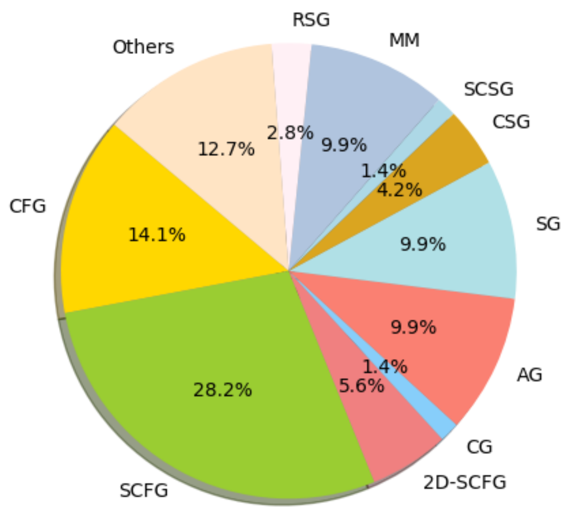

Figure 3.2: Syntactic methods used by the surveyed papers: Context-free Grammar (CFG), Stochastic Context-free Grammar (SCFG), Two-dimensional Stochastic Context-free Grammar (2D-SCFG), Coordinate Grammar (CG), Attribute grammar (AG), Shape Grammar (SG), Context-sensitive Grammar (CSG), Stochastic Context-sensitive Grammar (SCSG), Markov Models (MM), Relational String Graph (RSG), and Others.

tion rules from this grammar. Similarly, within the context of the computer vision, this grammar could generate all possible visual representations for an object or all possible actions for a video category, and finally, all the possible configurations for a scene. This makes the syntactic pattern recognition methods to be characterized by providing models based on parts, hierarchical and semantical decomposition, and that exhibit large amounts of structural variations. We list below some common computer vision problems that can be approached by syntactic methods, and in Figure 3.3, we illustrate three of these problems.

- Pose estimation: the model is trained with the object or person parts using the grammar to relate those parts. In this way, an unlimited number of poses can be generated by the grammar production rules from a finite set of training data. Another application of syntactic methods on the pose estimation is to infer occluded object parts from a given known sequence of object parts. Besides, a poses sequence obtained from a video can be used to predict, for example, falls or human actions.

- Object tracking: the model is built to estimate the state of the target object or person in a video from the previous state of the target. The person or object trajectory can be modeled by a graph controlled by production rules or a Markov chain (here as a syntactic method). On the object tracking, the syntactic methods can be applied in traffic surveillance, tracking of players actions, interaction between people, human actions, etc.

- Scene understanding: it is the process of analyzing and elaborating an interpretation for a scene. The model can be built using a graph to relate objects and persons in the context of a scene. The relationship between scene components can be controlled by probabilistic rules and spatial locations that define valid scenes. The scene understanding is relevant for applications that operate in the real world.

- Object detection: this task deals with the detection of object instances of a certain category in an image or video. Usually, the built model relates object parts in a compositional and hierarchical way guided by grammatical rules. This model is useful to infer occluded object parts from a given known sequence of object parts.

- Facade parsing: the facade parsing is a similar problem to scene understanding. However, the main applications are mainly aimed at urbanism and 3D building model generation tasks. The syntactic model for facade parsing is based on a segmenting image process of facades into semantic categories corresponding to architectural components such as roof, windows, and walls. The built model relates those components in a spatial and compositional way guided by grammatical rules. This model is useful to infer occluded architectural components by trees, cars, or buses from a given known sequence of other components.

In this section, the 71 surveyed papers will have their methods analyzed. Although all the works analyzed use syntactic methods, most of them combined syntactic methods with other approaches. Generally, the syntactic method plays a hierarchical representation of objects or scenes, when applied in static images. In papers that focus on the tracking of objects in videos, the syntactic methods are used to represent a probabilistic model of possible sequences of actions and possible positioning of objects in space and time. Most of the time, the papers use deep learning techniques, shallow algorithms, or appearance descriptors to recognize and detect objects or detect actions. In these papers, the syntactic methods were used as a high-level structure to relate objects or actions into a hierarchical or semantic way.

The studies set out to perform the most diverse tasks. Some papers have focused on detecting objects in scenes. Others have focused on tracking objects in videos and predicting sequences of actions; many described scenes or analyzed facades of buildings. We analyze each paper focusing on identifying the purpose of the study, the method adopted to perform the task and presenting the results achieved.

The papers were grouped into sections, representing each syntactic method (see Figure 3.2). At the beginning of each section, we briefly describe the specific characteristics of each syntactic

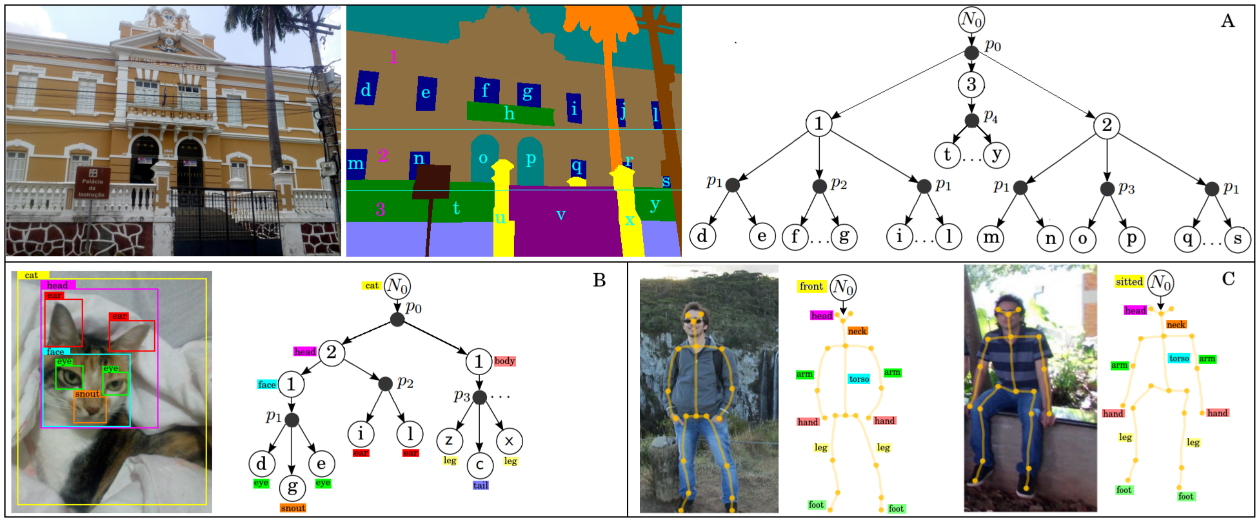
Figure 3.3: A: On the left, a facade. On the middle: segmented facade into semantic categories. On the right: Hierarchy of patterns corresponding to the segments (components). The circle with letters correspond to classes (window, door, etc.). Note that the hierarchical model infers some occluded components (*j, l, n, q, r,* and *s*). Black circles are production rules. B: On the left: a detected object based on part-level. On the right, the graph represents the part (letters) configurations guided by production rules represented by black circles. C: On the left, a human pose is represented by a skeleton graph format. On the right, the graph represents the dependencies between body parts, whose configurations are controlled by grammar production rules.

method to contribute to the understanding of the analyzed papers.

Context-free Grammar

A Context-free Grammar (CFG) is a four-element tuple $G = (V_N, V_T, S_0, P)$, where V_N are the non-terminal symbols, V_T are the terminal symbols, S_0 is the initial non-terminal symbol, and P are the production rules: $A \to \alpha$ where, $A \in V_N$ and $\alpha \in (V_N \cup V_T)^+$ (Wu, 2013). It can prove quite useful in decomposing videos into actions and object into parts both in a hierarchical and compositional way (see example in Figure 3.4).

The methods CFG-based are able to recognize human actions and interactions as well as relate them to formally represent complex activities. Feng et al. (2014) have proposed a method that obtains human poses from video sequences and relates them to predict falls of elderly people. The method uses a background subtraction technique to obtain human silhouettes from videos and represent them using ellipse fitting. Each ellipse has a set of features that are used to define the body's pose in a given video frame. These poses are learned by a Support Vector Machine (SVM) and afterwards converted into four rules of a CFG. The silhouette's motion through video is modeled using an Integrated Normalized Motion Energy (INME) image, whose

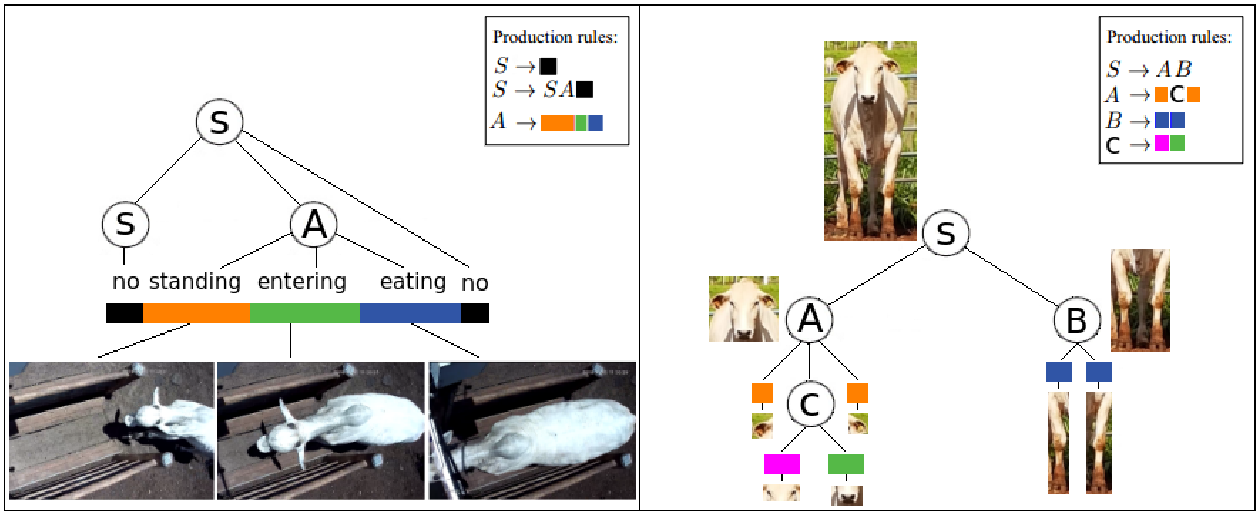

Figure 3.4: On the left, we present a hierarchical parse of a video decomposed into sub-actions. The action "bull is going to eat" is decomposed into sub-actions: *standing* at the door, *entering* in the feed bunk, and *eating*. The *no* symbol is the "no bull". On the right, we show the object appearance decomposed into parts-level. The decompositions are guided by the production rules of specialized CFGs.

poses are concatenated to try and detect a possible fall. The method achieved high detection accuracy and no false positives.

Pirsiavash and Ramanan (2014) have used a Latent Hierarchical model and a Segmental CFG model, respectively, to decompose actions from video into a hierarchical way and capture it's temporal structure. The method computes a bag-of-features descriptor for each video segment (frame of action) and assigns for each one of them a visual word, for example, pull, raise, pause, and background. The visual word sequences extracted from each video are used as the basis for building a CFG, whose non-terminal symbols represent actions and terminal symbols represent sub-action and background segments. The method achieved 62% accuracy at frame labeling and 22% for segment detection. Kuehne et al. (2016) also have parsed videos to recognize human activities. The method represents each video frame using a Fisher Vector (Jaakkola and Haussler, 1999) obtained by means of Dense Trajectory Features (DTF) (Wang et al., 2013) fitted by Gaussian mixture. The action units found in the video are modeled using an Hidden Markov Model (HMM) (see HMM in section 3.4.2) and sequenced into a CFG. The recognition of a probable sequence of actions is determined by combining HMMs and learned CFG. The authors proposed a new method version in (Kuehne et al., 2017). In this work, they introduced a weakly supervised learning approach from video transcripts to identify the sequence in which human actions happen in the video. They used a speech recognition technique on the video transcripts to identify each action and model them in the form of an HMM. The captured action units of the video are sequenced using a concatenation of HMMs. In the experiments, the method achieved an accuracy of 62% in aligning transcripts with the video data. However, on the segment detection, the method achieved an accuracy of 43%. Tian et al. (2015) have proposed a vehicle detection method in traffic surveillance videos that handles partial occlusion. The vehicle is decomposed into two-layer; the first layer contains the semantic parts and the second the decomposition of them. The parameters and the production rules of the CFG are extracted using an SVM classifier, which analyzes two or more semantic parts to indicates whether they belong to the same vehicle, and the appearance information of the parts, which are obtained using Histogram Oriented Gradients (HOG), Deformable Part Models (DPM) (Bourdev et al., 2010) and Latent SVM. The authors referred to approach as *pairwise SVM* Grammars. In experiments performed on real urban scenarios, the approach adapted to partial occlusion problems. Le et al. (2017) have proposed a method for monitoring drivers. A CNN is trained using driver body parts (face, torso and seat belt), and the probability map generated by it is inputted into a Semi-Supervised Normalized Cuts segmentation algorithm to build driver representation. From this representation, it is extracted the features of the driver region bounding box and foreground using an Region with Convolutional Neural Networks (R-CNN) to precisely segment driver body parts. The segments of each body parts are classified by an SVM and represented by a directed acyclic graph that embodies CFG rules. The decomposed parts of the driver are modeled as $\Gamma = \{\gamma_k\}_{k=1}^{K}$, where $\gamma = (x_k, y_k, \vartheta_k, s_k)$ denotes the position (x_k, y_k), orientation ϑ_k and scale s_k of part k of the driver, respectively. The head, body and seat belt are the main parts, however, the head is modeled into the sub-parts mouth, eyes, and nose. The approach has achieved results better than to state-of-art.

Two papers surveyed have used DPM, which is essentially a CFG, to deals with the object appearances at object parts-level instead of the object as a whole. Jiang and Ma (2015) proposed a model to person detection. The method detects the human body appearance using HOG III that is defined by a combination of bar-shape features (HoB), color features (HoC) and HOG, and decomposes it into six movable parts and, in some instances, combined with an occluder. The human body parts combination is done by a grammar defined by hand, while the learning of the HOG III template appearances, grammar deformation parameters and scores of productions is through Latent SVM. The method achieved 57% in the experiment on PASCAL VOC dataset for person-class. Gonfaus et al. (2015) also dealt with the object appearances at parts-level using a DPM, however, the proposed method by them can be applied to any object class. The authors illustrate the method by mean of a car composition. They use two different wheel styles for the same car instead of using two identical cars with different wheel styles. The method uses an

SVM to learn the objects parts appearance from a grid-like structure and relates the parts using an And-Or graph (AOG). In the experiments, the method achieved results similar to state-of-art.

Usually, Shape Grammar (see section 3.4.2), which is a particular type of CFG, is an established grammatical model to express hierarchical spatial relationships and therefore is very used for building facade interpretation. However, Demir et al. (2015) proposed a method that uses a CFG to represent urban structures. The proposed method aims to identify models matching to detect repeated structures. The method receives as input the point clouds that represent the urban structures and segments them to discover dominants planes through distance-based grouping and detection of point repetitions. The extracted segments are converted into a tree representation, whose nodes correspond to segments or the subset of segments. The approach is limited to generate few rules to a CFG if the points clouds have few repetition patterns.

A Graph Grammar G is labeled graph with labels in its vertices and edges. It is denoted by a pair (V_G, E_G), where V_G represents the vertices of G and E_G represents the edges of G. Moreover, G has labels in its vertices and edges. The labels are defined by a function f that assigns symbol labels to vertices and relation labels to edges (Pfaltz and Rosenfeld, 1969). A Graph Grammar is a CFG used to model two-dimensional (2D) data as labeled graphs (Pfaltz and Rosenfeld, 1969). Julca-Aguilar et al. (2017) have proposed a method for recognizing online mathematical expression using Graph Grammar. They model a handwritten mathematical expression like a labeled graph created from a graph grammar. Non-terminal nodes represent sub-components of the mathematical expression, terminal nodes represent the symbols (for example, +), and the edges represent the relation between symbols and sub-components. The recognition problem is modeled as a graph parsing; given an input stroke set, the algorithm obtains a parse tree that describes the best interpretation of the input. The method generates multiple interpretations of the mathematical expression consistent with the CFG, and then it extracts an optimal interpretation according to a cost function that takes into consideration the probability of symbols and structures.

Stochastic Context-free Grammar

A Stochastic Context-free Grammar (SCFG) is a CFG in which each production is increased with rules of probability. It is defined as a tuple $G = (V_N, V_T, S_0, R, P)$ (Wu, 2013), where V_N are the non-terminal symbols, V_T are the terminal symbols, S_0 is the initial non-terminal symbol, R are the production rules, and P are the probabilities on production rules. So every rule is associated with a probability: $P(A \rightarrow \alpha) \in [0, 1]$ and $\Sigma_{\forall \alpha} P(A \rightarrow \alpha) = 1$ (Wu, 2013). The SCFG is widely used in computer vision due to its stochastic process, whose model, usually represented by an *And-Or* graph (AOG) or its variants, defines a probabilistic distribution over a set of valid compositions. This characteristic allows modeling temporal and causal events from video input as well as modeling contextual relationships between objects and their parts in scenes. Figure 3.5 shows application examples of SCFG on the decomposition of video and object based on a probability distribution.

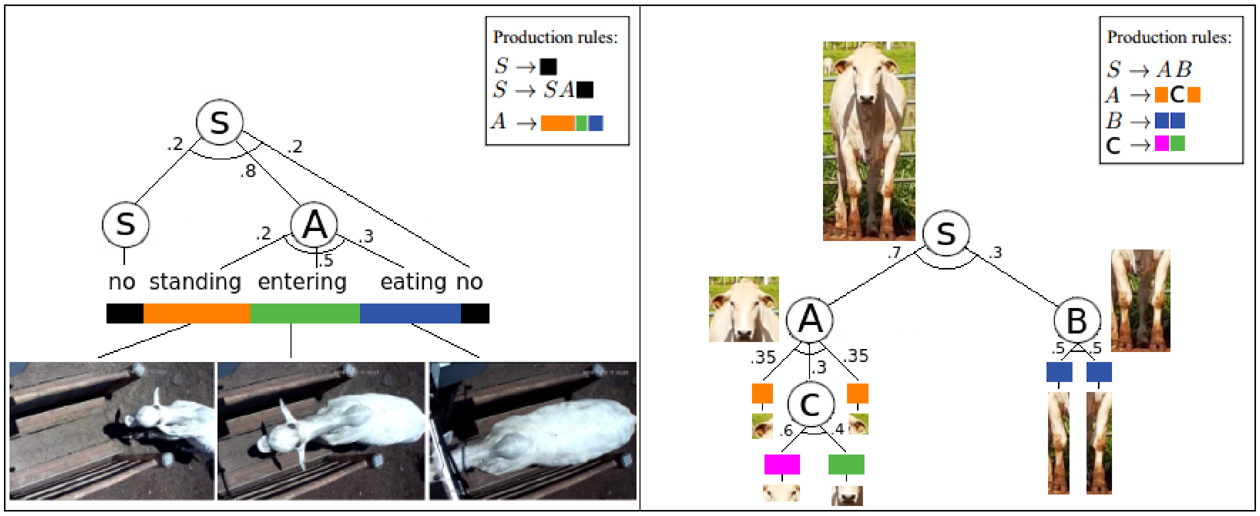

Figure 3.5: On the left, we present a hierarchical parse of a video decomposed into sub-actions based on a probability distribution. The action "bull is going to eat" is decomposed into sub-actions: *standing* at the door, *entering* in the feed bunk, and *eating*. The *no* symbol is the "no bull". On the right, we show the object appearance decomposed into parts-level. The decompositions are guided by rules preceded by a probability that indicates the relative frequency with which it occurs.

On the human activities modeling, Vo and Bobick (2014) have used an SCFG to classify body poses in complex activities. The authors modeled an AOG to represent temporal structures of complex human activities, whose *or-nodes* model the variation in the actions progress, and *and-nodes* concatenate sub-actions of a complex activity. From action detection, it is created a Bayesian Network to represent smaller actions sequence, whose observed nodes are primitive actions, hidden nodes are action timings and the edges are dependencies between actions. The approach aims to represent activities structure probabilistically, allowing to evaluate the probability of occurrence of any action at any time. The method achieved 58% segmentation accuracy and an offline mode segmentation accuracy of 91.8%. Subsequently, Vo and Bobick (2016) evolved the approach using a different action primitives detector and performed new experiments using a new dataset. The model proposed by Lee et al. (2015), beyond encapsulates the temporal structure of activities, keeps a history of the observed actions. The authors used HOG to extract visual features from video frames, a Random Forest classifier to recognize elementary human actions that result in a likelihood distribution of actions, and an SCFG to represent temporal structures of human activities. The model receives as input a sequence of action for predicting the next action or successive actions. The approach was tested on a dataset with concurrent activities performed by multiple humans and achieved good results. Tayyub et al. (2018) annotated in the videos each human activity using labels to build a hierarchical activities model. The proposed method by authors infers a "part-of" hierarchical activity model occurring within each video from the semantic similarity between the labels. Then, the method clusters the hierarchical activity models derived from each video to generate a unified hierarchical and probabilistic model embodied in an SCFG, which captures the variation in activities through *Or* rules on a generated AOG. In video inference and interpretation, first the primitive actions in the video are automatically detected using a state-of-the-art action recognition approach. Then, based on the SCFG, the method infers the most likely activity hierarchy to the set of primitive actions detected in the video. Qi et al. (2017) have integrated objects and human actions to predict future human activities from partially observed videos. The compositional structure of events is modeled by an SCFG in a Spatial-Temporal *And-Or* graph (ST-AOG), whose and-nodes represents a decomposition of an action (e.g., take to the oven) into its constituents (sub-actions such as opening oven, putting in bread), the or-nodes represents the variation in actions (put a bread or put a cake into the oven) and the terminal nodes represent observation (human and objects in a video frame). The method uses an Earley parser (grammatical parser algorithm) to predict sub-actions and all the learned cues (parsed graph and sub-actions) to predict human activity. The authors reported that the method achieved excellent results. Fire and Zhu (2017) embodied an SCFG in a Causal *And-Or* Graph (C-AOG) to infer the causal-effect relationship between object and actions agent, for example, a person pushing the door. The C-AOG is used to connect the actions in a sequential model. The model allows connecting triggering agents to actions and actions to their effects, allowing long-term inference of actions and agents. The authors modeled or-nodes like an alternative cause (e.g., a printer can start printing by someone using a computer), and-nodes like connectors of sub-actions and conditions (e.g., the sub-actions performed to detect the keyboard use), terminal nodes like complete actions or flow change in the video, and horizontal relationships between nodes like temporal relationships (e.g., a person nears the door before opening it). All rules were designed manually based on video sequences produced in the laboratory.

On the detection and track moving objects in video, Xu et al. (2018) have proposed an approach that uses a C-AOG for tracking and modeling causal-effect relations on the interaction between people and people with the environment. The method tracks people like a fluent variable that changing visibility status (visible, occluded, or contained) by interacting with the environment, for example, getting into a car, entering a house, etc. In generated C-AOG, or-nodes represent the visibility status, the edges indicate how people transit among visibility status, the terminal nodes represent actions or sub-event describe by concatenation of and-nodes. For modeling the transitions of the people, indicating in the video his position and visibility status, the method uses a model based on a Markov Chain (see section 3.4.2 about Markov Model), creating a dependency between the states of the people. Moreover, the method uses an Integer Linear Programming model to search the optimal states of the people over time. Thus, given a video, the proposed method can predict the visibility status of the people and recover their complete trajectories. The proposed approach by Li et al. (2015) aims to show the location of pedestrians in complex traffic surveillance environments. The method captures expressive keypoints in video frames using HOG features at multiple scales to obtain the parts of the human body (head, arm, etc.) and relates these parts into an AOG. The related parts of each frame are clustered and submitted to the HOG-SVM detector for training. To estimate the pedestrian localization and the body parts configurations is employed convolution and a bottom-up inference on OAG. The approach proposed by Lu et al. (2014) infers hidden states of objects in video frames. In the first frame of the video, the method uses HOG, local binary pattern features (LBP) (Ojala et al., 1994), and RGB color histograms (for color videos) to obtain the appearance features of the objects. After, the objects are divided into a small cell-based grid (3×3) and organized into an AOG, whose terminal nodes represent the parts of the objects, the nonterminal nodes the part decomposition, and or-nodes the alternative parts decomposition. The method uses a Latent SVM to learn the AOGs and a temporal dynamic programming algorithm based on HMM (see section 3.4.2) to formulate the object tracking in the video. The method has outperformed state-of-the-art tracking algorithms, including convolutional neural network-based deep learning techniques.

SCFG also was used by some of the surveyed papers to detect objects in images. Song et al. (2013) used an *And-Or* Tree (AOT) to relate into a compositional way primitives shapes of objects, whose appearances were obtained using HOG. The AOT represents different subcategories and perspectives of objects; for example, a front horse image, side horse image, and back horse image. These subcategories are learned using a clustering algorithm in an unsupervised way while the configuration of object parts are learned in a weakly-supervised way, and the generated parameters of both learning process are learned together by an SVM. The method achieved better performance than the baseline methods on a dataset of 20 object categories. Li et al. (2014) have addressed the occluded object detection problem. The proposed method uses an *And-Or* Directed

Acyclic Graph (AODAG) to model X-to-X occluded objects, for example, car-to-car or person-to-person in a hierarchical relationship way. The appearance features of occluded objects pairs (X-to-X) and the corresponding two separate objects (X and X) were obtained using HOG, and then these appearance features were decomposed into an AODAG, whose or-nodes represent a mixture of various types of occluding objects pairs, and-nodes both occluded objects pairs or occluded objects single, and terminal nodes the appearance for each object pair. The final model embeds an occluding object pair detection SCFG that is learned by the Latent Structural SVM (LSSVM). The method was tested only on car-to-car or person-to-person, although according to authors, it can be used to recognize other object types. Rothrock et al. (2013) have used an SCFG embodied in an OAG to represent human pose variation. The method handles the appearance variation of people by substituting their parts by variants. The obtained human parts by means of segmentation and Dense HOG are related in an OAG, whose and-nodes represent the distinct part appearance, or-nodes the appearance variations, and the edges determine the context relation between variant parts of the human pose. In experiments, the authors demonstrated the effectiveness of the model for human pose estimation.

In the context of scene understanding, the researchers use SCFG to represent objects in a hierarchical and compositional way or to relate objects in scenes. Chua and Felzenszwalb (2016), for example, have explored contextual evidence provided by the compositional rules of an SCFG to identify objects and infer parts of them when they are missing in the scene. The method uses implicit blocks for defining the scenes that represent a pair of type and pose, respectively, an alphabet symbol and the type position in space. Factor graphs (bipartite graph representing the factorization of a function of several variables (Forney, 2001)) and loopy belief propagation are used to represent a probability distribution about possible scenes. The authors have shown experimental results with two different applications. The first application involves the reconstruction of binary contour maps, and another detects human faces in the image. In both experiments, the method handles with robust inference algorithms that can effectively combine local information to reason about a scene. Zhao and Zhu (2013) have analyzed indoor scenes to infer the object functions. They assumed that an indoor environment is designed to meet human needs like a bed to sleep on or sofa to sit on. To identify the objects and their functionality, the method first groups detected line segments in 3D primitives to build shapes; following it assigns labels to primitive shapes to represent its functionality, forming a parsing tree that represents the appearance hierarchical and function of the objects. In the inference process, the method, guided by an SCFG, uses the Metropolis-Hastings algorithm (Robert and Casella, 1999) to accept or reject the parse tree. The proposed method by Walton et al. (2017)

was applied in the Naval Tactical domain. The authors aim to represent semantic and physical relationships, such as the spatial, temporal, and semantic context between objects and events in a scene. The scenes are segmented and decomposed into an AOG, whose nodes represent entities, the internal edges specify spatial and functional relations between objects, and the lateral edges correspond to relations that allow the graph to encode contextual information between entities at all hierarchy levels of a subgraph. Although the authors have considered that an SCFG is suitable for analyzing tasks in scenes, they concluded that inferring the context of events and objects in the Naval tactical command and control domain is a very critical task that must be done by largely trained and experienced human operators.

There are works as Jiang et al. (2018) and Qi et al. (2018) that use SCFG to learn the layout of the indoor scenes in order to automatically generate a large-scale 3D indoor scene dataset. These approaches receive as input 2D images with pixel-wise ground-truth or set of image segments, and by means of the rules learned from grammar, combine components of a given indoor scene to generate new layouts to it. The generated scene datasets are very useful for serve as training data for object detection and scene understanding tasks.

In order to introduce an explainable generative model, Xing et al. (2019) propose a method that uses the first convolutional layers of a generator network (Goodfellow et al., 2014) as a feature descriptor and an AOG to represent a learning model. The method applies sparse operation on the feature maps obtained from convolutional operations on image to identify primitives such as edges, colors, and object parts. These primitives are connected using an AOG in order to represent visual patterns in a hierarchical and compositional way. The experiments showed that the method can learn visual patterns in different convolutional layers of a generator network. Wang et al. (2018) also use CNN architecture for getting image features and relate them using grammar with similar characteristics to a SCFG. The authors introduce Fashion Grammar to classify fashion images. The grammar is composed of two other grammars: symmetry grammar that models the bilateral symmetric properties of clothes, and kinematics grammar that is responsible for describing kinematic relations between clothing landmarks. The method starts by producing clothing landmarks on the fashion items, such as the corners of the neckline, hemline, and cuff, using a heatmap obtained by means of the conv4-3 layer of the VGG-16 Architecture (Simonyan and Zisserman, 2015). These landmarks are related according to specific rules of kinematics and symmetry grammars to build global constraints in fashion grammar. The approach uses Bidirectional Convolutional Recurrent Neural Networks (BCRNNs) units to message passing over the fashion grammar for iteratively update and refine landmarks. The final classifying clothes process is performed by two types of attention mechanisms, the first concentrating on the functional clothing regions and the second on the categories. The method was tested on two datasets and achieved the state-of-the-art at classifying fashion images. Xing et al. (2020) proposed an approach that produces an AOG model by sparsifying generator network (Goodfellow et al., 2014) for generating image synthesis with explicit hierarchical and compositional representations. The proposed model converts dense connections between layers in a generator network to a sparsely activated and connected AOG model. The AOG is generated from sparsity-inducing constraints introduction in training. The aim is to learn a hierarchical compositional of scene-objects-parts-subparts primitives for AOG-guided image synthesis generation from scratch. The experiments performed on four benchmark datasets showed that the approach achieved good results.

Two-dimensional Stochastic Context-free Grammar

The Two-dimensional Stochastic Context-free Grammar (2D-SCFG) is a extension of SCFG and introduces mainly two differences. The terminal and non-terminal symbols characterize two-dimensional regions and are associated with 2D coordinates, and production rules have a parameter (*spr*) that relates terminal and non-terminal symbols spatially. The rules in 2D-SCFG are defined as $A \xrightarrow{spr} \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup V_T)^*$ and *spr* defines the spatial relationship between nodes, whose possible labels can be: up, bottom, left, right, superscript, subscript and inside (Wu, 2013). Figure 3.6 shows an example of 2D-SCFG on the decomposition of an object. Note that, besides the probability distribution, non-terminal symbols are related by a parameter (*spr*), and terminal symbols are associated with 2D coordinates.

We surveyed four papers that use 2D-SCFG and mainly explore the spatial relationship provided by it. Álvaro et al. (2014) have proposed a model to recognize handwritten mathematical expressions using 2D-SCFG and HMM. The HMM recognizes mathematical symbols and the 2D-SCFG models the relationship between them. The rules to grammar 2D-SCFG were defined manually in order to map the horizontal (AB), superscript (A^B), subscript (A_B), vertical ($\frac{A}{B}$), and inside $\sqrt{A}$ relation between symbols. The complete mathematical expressions are learned by an SVM from samples of spatial relation types between mathematical symbols. This model is improved in Álvaro et al. (2016), in which the authors propose modeling the structural relationships between symbols of a mathematical expression by means of a statistical grammatical model analogous to a 2D-SCFG. The mathematical expression is divided into a sequence of strokes and represented by a parse tree, whose mathematical symbols are classified by means of a Bidirectional Long Short-Term Memory RNNs (BLSTM-RNN) (Schuster and Paliwal, 1997). The mathematical expression recognition is carried out by comparing parse trees. The method

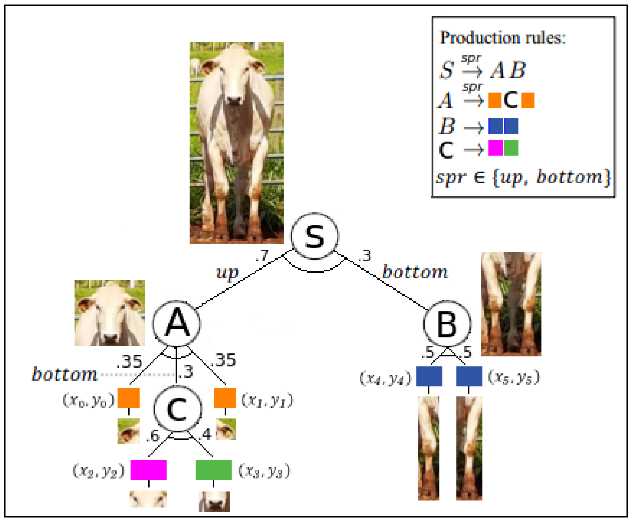

Figure 3.6: The object appearance is decomposed into parts-level. The decomposition is guided by rules preceded by a probability that indicates the relative frequency with which it occurs. Besides, non-terminal symbols are related by a parameter (*spr*), and terminal symbols are associated with 2D coordinates. Non-terminal symbols could also be associated with 2D coordinates.

achieved 82.2% of precision on the task of symbols classification and segmentation, which is a significantly superior result to state-of-the-art methods.

Martinovic and Van Gool (2013) used a 2D-SCFG version that includes attributes (see section 3.4.2 about attribute grammar) to represent building facades. The grammar called Two-Dimensional Attributed Stochastic Context-Free Grammar (2D-ASCFG) is defined as a tuple $G = (V_N, V_T, S_0, R, P, A)$, where V_N are the non-terminal symbols, V_T are the terminal symbols, S_0 is the starting symbol, R are the production rules, $\{P(r),\ r \in R\}$ are the probability rules and $\{A(r),\ r \in R\}$ are the attribute rules (Martinovic and Gool, 2013). A facade image is semantically segmented into classes such as roof, windows, and walls to generate rules for a Split grammar. As a result, it is obtained grammar rules for each facade image. Then, all rules are merged, following a Bayesian model-merging technique, to build a single 2D-ASCFG. The method achieved an accuracy of 74.82% in the facade analysis on a dataset of 30 images. The authors only used a limit of 30 images due to the cost of processing the method pipeline.

Fang et al. (2018) proposed a pose grammar model that uses three kinds of grammar, which together are analogous to a 2D-SCFG. The model maps a human pose from 2D to 3D, relating the dependencies between body parts, in order to 3D human pose estimation. The model has a deep neural network constituted by two basic blocks used for extracting pose-aligned feature, and a pose grammar network composed by kinematic, symmetry, and motor coordination grammars that encode human body parts dependencies and relations which are learned by an Long Short Term Memory (LSTM) (Greff et al., 2017). The model basically extends three types of human pose grammar into deep neural networks to model high-level knowledge of 3D human pose. The approach was evaluated on three popular 3D pose estimation datasets and compared with 16 state-of-the-art methods in a way quantitative and qualitative. According to the authors, the model obtained superior performance over the 16 state-of-the-art methods.

Coordinate Grammar

A Coordinate Grammar (CG) is a CFG whose symbols are located at given coordinates which are computed by functions associated with the production rules (Ferber, 1986). A CG is an eight-element tuple: $G = (V_T, V_M, V_N, L, f, n, \Phi, P)$, where V_T are terminal symbols, V_M are intermediate non-terminal symbols, V_N are non-terminal symbol, L are the coordinates, f a special symbol, such that $f \notin \{V_T, V_M, V_N, L\}$, n a natural number, the order of the grammar, Φ are functions associated with the production rules, and P are production rules (Ferber, 1986).

Our search returned only the work of Ayeb et al. (2015) that use CG. The proposed method aims to recognize Arabic mathematical formulas extracted from scanned images. The symbols of the formulas are recognized singly using a combination of the features descriptors Zernike moments, Hu moments, run-length, white pixel portion, and bilevel co-occurrence and the classifiers Multilayer Perceptron, K-Nearest Neighbors, Naive Bayes, K* and Decision Tree. The symbols of the formula are related using a CG and the recognizing of the complete formula is performed by a top-down and bottom-up parsing scheme based on operator dominance. The method was tested on a dataset with 5000 mathematical symbols, and the best recognition rate, 91%, was achieved by K*.

Attribute Grammar

An Attribute Grammar (AG) is a CFG increased with attributes, semantic rules, and conditions. The attributes have values that are used by semantic rules associated with the production rules (Slonneger and Kurtz, 1995). These characteristics entitle AG to be used in scene understanding tasks and analyze objects trajectories in videos, whose models usually require a semantic and structural relationship between primitive patterns. Figure 3.7 shows application examples of AG on the decomposition of video and object based on attributes and semantic rules.

Our research returned four papers that used GA at some step in their pipelines to interpret scenes. The proposed method by de las Heras et al. (2015) interprets a scene from the structure of floor plans. The authors used annotated floor plans (i.e., walls, doors, and rooms) as input to the method to model the structure of a floor plan in a hierarchical composition of contextually constrained parts. A first step is to detect primitives, such as lines, wall or doors. The walls are

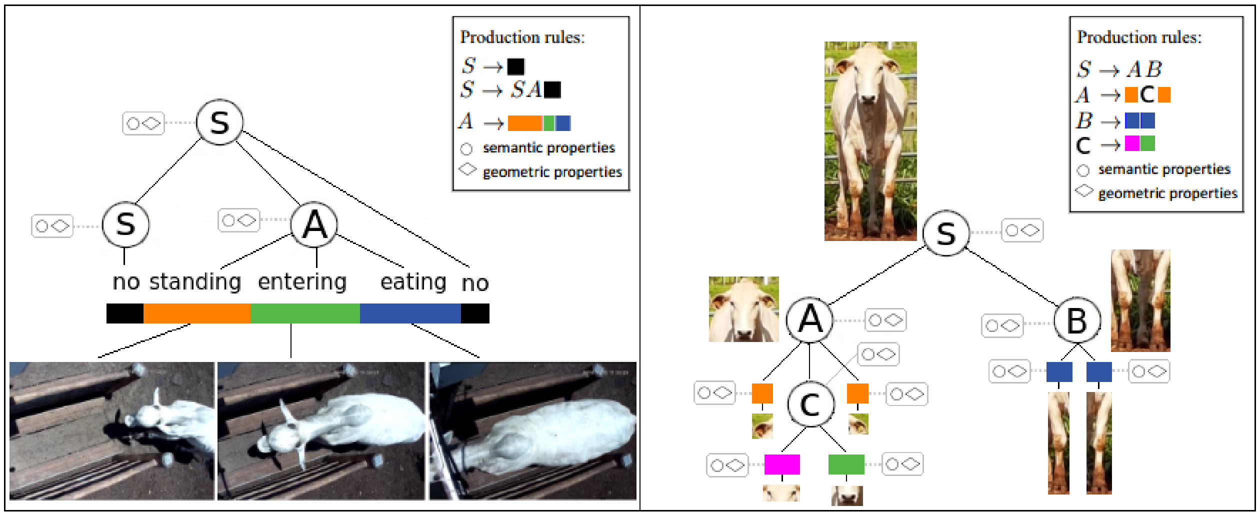

Figure 3.7: On the left, we present a hierarchical parse of a video decomposed into sub-actions. On the right, we show the object appearance decomposed into parts-level. The decompositions are guided by attribute evaluation rules called semantic functions.

detected using watershed transformation and the doors are discovered by finding arcs employing the Hough transform. After, the image is segmented into different domains for constructing the floor plan graph representation using the AG rules. They performed experiments using four datasets and show that the method is better than most recent floor plan interpretation techniques. In the same way, Boulch et al. (2013) have used AG to interpret semantic information on CAD building models. The method transforms the CAD model in graph nodes, whose terminals nodes are the 3D primitives, i.e., geometric forms. The objects in the scene are decomposed in a hierarchical way, and object parts are related by the AG. The structure of the scene is reflected in an analysis tree where the 3D primitives have semantic labels and relations. The authors consider that the method obtained excellent results. Despite this, they say new tests must be performed in real environments to deal with noise and occlusion. Liu et al. (2018b) and Liu et al. (2018b) have proposed methods for parsing outdoor scenes. The authors assume the scenes are composed of parallel lines that may form one cartesian coordinate system. In both methods, the buildings parts are decomposed in a hierarchical way and related semantically in a graph by an AG. The proposed method by Liu et al. (2018b) decomposes the scene into superpixels that are represented by graph nodes. Each node has attributes to represent the scene-level superpixel global geometry as well as local geometry. These attributes impose constraints between the graph nodes employing five grammar rules that are used to decompose the scene. To infer the optimal parse graph for a scene, the method employs recursively the five grammar rules. The proposed method by Liu et al. (2018b) differs from Liu et al. (2018b) only in the step of inference from the optimal parse graph for a given scene, Liu et al. (2018b) employ the five grammar rules using a probabilistic framework based on Markov Chain Monte Carlo (MCMC) (Brooks et al., 2011). In this new method version, the results are comparable to state-of-the-art 2D semantic region segmentation and single-view 3D scene reconstruction.

Usually, as shown in section 3.4.2, CFG is employed for video parsing, however, due to GA inherits basic features from CFG, we surveyed two papers that use GA to analyze objects trajectories in video. The proposed method by Choe et al. (2013) aims to retrieve videos from a large video dataset having similar activities to a given query video. The method divides the video into frames and each frame is segmented into superpixels which are clustered by a model based on Markov Random Field (MRF)(see section 3.4.2). The clustered superpixels are examined by analyzing the Spatio-temporal trajectory to detect the target (human, vehicle, etc) and basic actions of the target (appear, disappear, move, stationary, stop, etc), including context information. This information is reflected in an AOG, whose semantic rules are provided by productions of the AG, to model the activity in the video. In one of the experiments, the method achieved an accuracy rate of 80% on videos correctly retrieved. The method proposed by Liu et al. (2018a) tracks humans in the video. The person and its movement trajectory are described using a graph, whose nodes nonterminal represent sequences of person bounding boxes over a period of time. These nodes nonterminal are decomposed into children terminal nodes that represent only a person bounding box detected in a certain video frame. The terminal nodes, which are detected using deep learning architecture CaffeNet (Jia et al., 2014), have attributes like geometry (moving speed, direction), activities (walking, running), and/or accessories (bags). The method parses a given video employing both top-down and bottom-up parsing to infer for it an optimal parse graph together with its attributes. The graph construction is guided by semantic rules provided by an AG. The method outperformed state-of-the-art tracking methods using public video datasets.

We surveyed one paper that uses AG for the pose-estimation task. The method proposed by Park and Zhu (2015), aside from estimating poses, finds parts jointly and recognizes part attributes (i.e., long hair, glasses, hat, etc). The human body is decomposed into its constituent parts which are related in a hierarchical way by an AOG. The parts appearance is described using approaches based on CNN, HOG and color features, and the geometry state of them is represented by (x, y, s), i.e., position and scale. The AG rules define the relations of adjacency and semantic between the parts to infer an optimal parse tree for a given image. The method achieved better results against state-of-the-art methods on pose-estimation, part localization and attribute recognition tasks.

Shape Grammar

Shape Grammar (SG) is a type of CFG whose rules generate geometric shapes. The SG has two main rules which define how a shape can be transformed in others (parts of shapes). A rule deals with the recognition of a particular shape. The other deals with its possible replacement by a different shape (Stiny and Gips, 1971). Figure 3.8 shows an example of AG on the decomposition of an object. Note that complex shapes are generated from primitive geometric shapes.

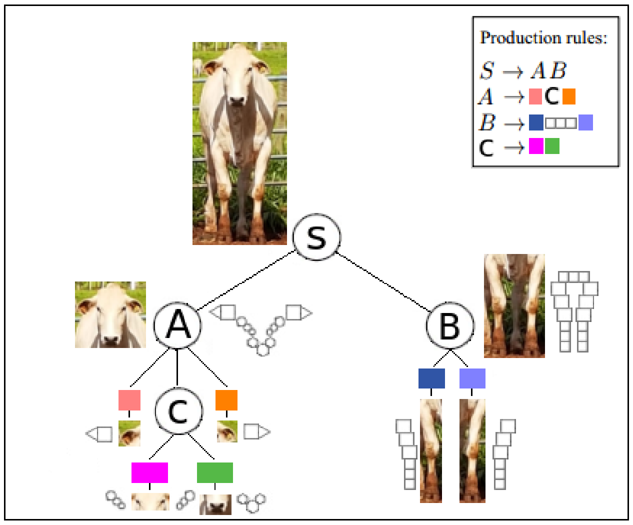

Figure 3.8: Example of a decomposed object by a Shape Grammar. The object appearance decomposition is guided by rules that generate complex shapes from primitive geometric shapes.

SG is an established method to represent hierarchical spatial relationships and is therefore suited to represent and interpret the semantic relationship between components of building facades. Our search returned five papers that use SG in your approaches to facade analyzes. The method proposed by Weissenberg et al. (2013) receives as input facade segments like door, window or piece of wall and provides as output the production rules and the parameters learned through the facade, in which both are combined to construct a parse tree representing the given facade. After, the method merges all parse trees obtained from each facade to generate a single grammar for the facade style. The method contribution is to eliminate the need for manual expert work to describe a facade model. Koziński et al. (2015a) have analyzed facades based on semantic people-defined rules in which stipulate the relationships vertically or horizontally between class pairs (windows and wall, roof and sky, etc.). The class pairs are obtained using a hierarchical partitioning of the facade into grids, and the semantic rules are modeled by an SG. The model is trained with class pairs to discover the structure of the occluded facade when is performed the facade parsing. In experiments, on several facade datasets, the method demonstrated state-of-the-art results. In previous work (Koziński et al., 2015b) to this, the authors decomposed the facades into rectangular regions reflected in a tree, whose terminal nodes correspond to segments as such windows, roof, etc. Another paper (Koziński and Marlet, 2014) by these authors, which will be shown in section 3.4.2, employs MRF to analyze facades. Teboul et al. (2013) also decomposed facade images in rectangles into predefined semantic classes such as a wall, window, balcony, roof, etc. However, the method employs a reinforcement learning-based approach to train an SG model. The authors adopted reinforcement learning due to the computational complexity of the problem. The results demonstrated that the method achieved state-of-the-art results in less time than similar approaches. Gadde et al. (2016) analyzed facades with regular architectural features like Haussmannian and Art deco to generate rules for a Split Grammars, which is a particular kind of SG where basic shapes are split into spatial regions. The method parses the training dataset facades using a generic grammar to generate various parse trees. These trees are merged to defines the rules of an observed architectural typology.

Zieliński et al. (2015) defined an SG to detect erosions and osteophytes in bone contours. The SG language contains an alphabet with two types of letters: arcs and angles. The arcs are primitives with information about the length, start and end of the angle. When concatenated, the arcs define a contour which is represented by a word. The hand radiographs are segmented to detect metacarpal bones and obtain information about outlines and joints of finger bones, aside from locations and borders of joint surfaces. The hand radiographs outlines are described employing a word built from the SG language alphabet (arcs and angles). The locations of erosions and osteophytes on hand radiographs are detected by parsing the words that represent them. The method located joints in 98.3% of cases. The sensitivity and specificity of detecting lesions were 70%.

An SG was used by Ikehata et al. (2015) to represent the environment structure and reconstruct an indoor scene like a structured model. The SG has eight rules for representing and constructing an environment as a whole. The rules represent the structure of the environment as the relationship between the rooms and the relationship between objects within the room. The environment structure is reflected in a graph, whose nodes correspond to elements such as rooms and walls. The node of a given room connects to a subgraph that represents the inner objects of it. Both the environment structure segmentation and the rooms internal segmentation were performed by algorithms proposed by the authors.

Context-sensitive Grammar

A Context-sensitive grammar (CSG) is a four-element tuple $G = (V_N, V_T, S_0, P)$, where V_N are non-terminal symbols, V_T are terminal symbols, S_0 the starting non-terminal symbol, and P are the production rules $\alpha A \beta \rightarrow \alpha \gamma \beta$ where $\gamma \neq \in$ (Özkural, 2014). During derivation non-terminal A will be changed to γ only when it is present in context of α and β (Özkural, 2014). Figure 3.9 presents a simple CSG instantiated on the decomposition of an object. Note that the properties and constraints are used to impose the context in object decomposition.

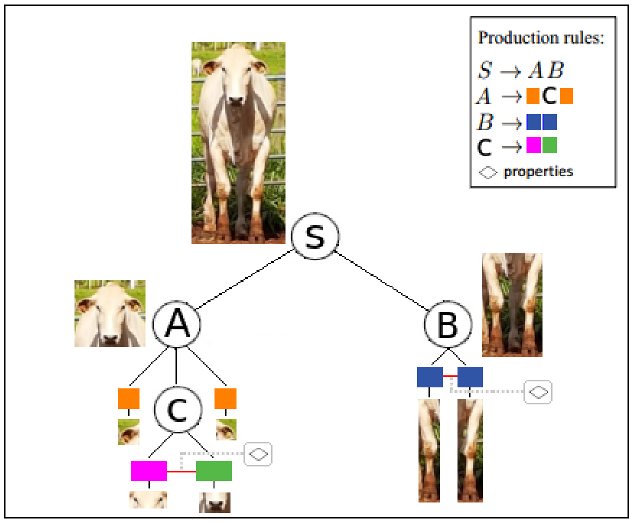

Figure 3.9: The CSG decomposes the object into part level by the terminal and non-terminal nodes. The horizontal links (red) in the graph, based on properties, impose the context and constraints between the nodes and guide the object decomposition in a probabilistic way.

Our search returned only three papers that used CSG. Park et al. (2018) used a CSG to represent human poses and attributes in a compositional way. The authors combined the grammatical models Phrase Structure Grammar, Dependency Grammar and Attribute Grammar to define the called Attribute And-Or Grammar (A-AOG) model which is essentially a CSG. Phrase Structure Grammar is used to represent the body parts such as head, trunk, arm, etc, and the part attributes such as the hairstyle (short, long, etc) are modeled by an Attribute Grammar. These parts, whose appearance is obtained by a CNN under different points of view and poses, are terminal nodes and are associated with their spatial locations (x, y). The Dependency Grammar is used to model body pose by a kinematic graph. The probability model of the A-AOG is formulated by a bayesian network that calculates the part of the body as a product of an earlier probability. The learning is divided into two stages. In the first, the model learns parts and poses. In the second, it learns the relationship between parts and attributes. The method achieved state-of-the-art in precision and performance. Li et al. (2019) also combined the grammatical models Phrase Structure Grammar and Dependency Grammar to build an AND-OR Grammar (AOG) which has CSG characteristics. The AOG is integrated into deep learning architectures and is called of AND-OR Grammar networks (AOGNets). An AOGNet consists of AOG blocks that unify the best practices developed by various deep learning architectures, i.e. the AOG guides deep learning architectures generators. The method achieved state-of-the-art performance on three benchmarks on tasks of objects classification and recognition. Zarchi et al. (2016) used a CSG to point out high-level contextual relationships between objects in a scene in order to understand it. The authors introduced the concept of *Visual Term*, which is a term used to refer to the co-occurrence of objects in scenes. For example, if a person riding a horse occurs in multiple scenes, the co-occurrence of these two objects is considered a *Visual Term*. The scene is decomposed into parts that are represented by an AOG, whose concatenated and-nodes represent the Visual Terms and or-nodes indicate the variations on them. The scene parts (objects) appearance is obtained by HOG, DPM and Latent SVM, at different perspectives (front, side, etc.). When there is the occlusion problem, the objects that co-occur are first determined and then the occluded objects are trained separately. The method outperforms other methods on *Visual Term* detection and, in most cases, on object detection.

Stochastic Context-sensitive Grammar

A Stochastic Context-sensitive Grammar (SCSG) is a CSG increased with $Pr(p_i)$, in which $Pr(p_i)$ assigns a probability to each production rule p_i (Özkural, 2014).

Pei et al. (2013) used an SCFG to model interactions between agents and objects in videos, like the interaction of a person with a laptop. The method models the interactions using a Temporal And-Or Graph (T-AOG), whose relations are defined base on positions of A and B in the form of $r(A,B)$, for example, $touch(people, laptop)$. These relations are grouped to build atomic actions, which are observed in the video when all its relations are identified with a high probability, and atomic actions are concatented to denote a completed action. The concatenated and-nodes of the T-AOG represent completed actions and or-nodes their alternative ways. To distinguish different actions with similar structures, the child nodes of a given and-node have their temporal relations modeled. When parsing the T-OAG, the method infers the goal of a given person and predicts their intents. The method achieved an accuracy of 90% in predicting of the goal of the person and 87% in predicting intention.

Markov Models

The Markov chain and its variations are stochastic models widely used in syntactic pattern recognition in computer vision. Basically, these models emit a sequence of states that satisfying a property called Markov property from which can be derived a string or a sequence of events. A process satisfies the Markov property if its state at the time step t_1 depends only on the state at time step *t* and not on the states previous (Ghahramani, 2001). In other words, the movement to the next state depends exclusively on the current state. In this review, we surveyed seven papers that use variations of Markov chains, four of which track objects in videos and two analyze images. Figure 3.10 presents generic Markov models, one for tracking objects and another for interpreting.

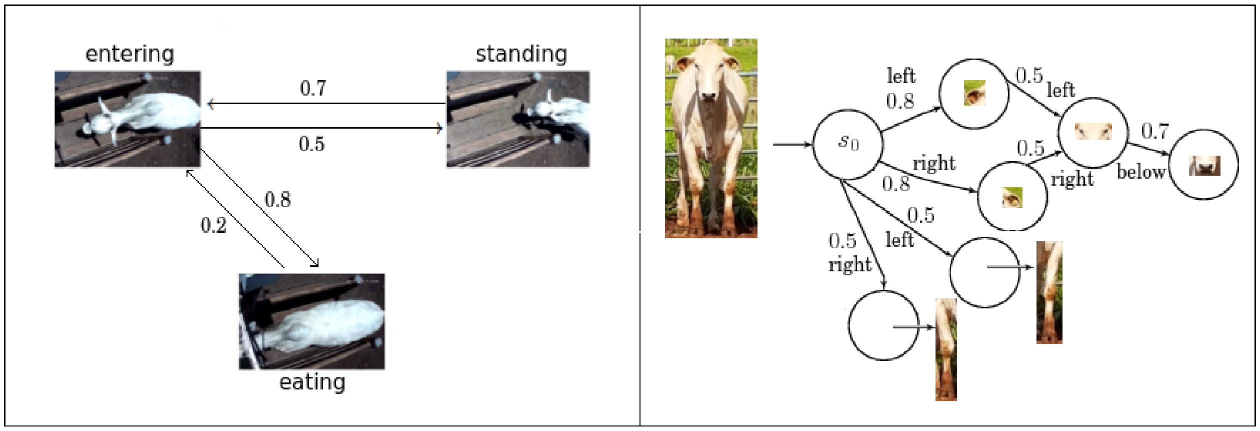

Figure 3.10: On the left, the model tracks the bull and identify their actions based on a chain guided by a probability distribution. The words (standing, entering, eating) represent states. On the right, the object is built by a chain also guided by a probability distribution. The circles represent parts of the object and words (left, right, below) represent model constraints.

Krüger and Herzog (2013) tracked people in the video to recognize their actions using Parametric Hidden Markov Models (PHMM), which is a type of Markov chain that represents a probability distribution on a sequence of states increased with parameters (Wilson and Bobick, 1999). Action recognition and body tracking are done from an object-driven perspective, e.g., instead of the method considers body poses, it considers the actions practiced to objects. The primitive actions are captured by a 3D tracking device like the Kinect and modeled by a PHMM to represent parametric movements. The complex actions are modeled by concatenating primitive actions, whose control is done by a grammar. Finally, body tracking is performed using particle filtering based on the parameters of the primitive actions. Windridge et al. (2015) used another variation of Markov chain called Markov Logic Network (MLN) to track tennis ball and annotate tennis game videos. According to (Mihalkova et al., 2007), a MLN consists of first-order formulas with weights. A formula represents a relational rule, and its weight represents the importance of that rule. The state of the network are formulas and connections between states are logical connections. The formulas are associated with functions that use the weight to perform a calculation and build the Markov chain. The method learns game-rule from video frames employing a second-order meta-grammar for MLN construction. However, to be able to learn the game-rules it is necessary to identify its state in each frame. This is done based on the ball position and the players' state. Information from these two objects is obtained by identifying the tennis court lines using Hough transformation, by tracking player using background subtraction and particle filter, by recognizing player actions using K-Nearest Neighbors and HOG 3D, and by tracking ball using background subtraction and SVM. The method achieved event prediction accuracy of 68% on a simulated tennis game dataset. Liu et al. (2016) used MLN to track people and identify their daily tasks. The method obtains tasks information using sensors and categorizes them into three types: locomotion, left-hand and right-hand action. Classifiers based on C4.5, SVM, Decision Tree, and K-Nearest Neighbors are used to recognize atomic actions within these three categories and annotate them using labels, for example, action "Alice cleans table using a right hand" can be labeled as follows [clean, table, Alice, right_hand, atomic]. Then, intervals that represent sets of atomic activities, are created and related to each other by means of temporal and hierarchical relations to construct atomic activity sequences. The patterns are extracted from these sequences for generating rules set. Each rule in the set is converted to clauses to form an MLN for inferring daily tasks. Kong and Ranganath (2014) used the semi-Markov chain model, which is a variation of a Markov chain where each state persists for a certain time before transitioning to another state (Sarawagi and Cohen, 2004), to recognize continuous gestures of American Sign Language (ASL). The method uses Cyberglove and magnetic trackers to obtain the gestures sequence. The sequences are segmented and each segment is labeled using SIGN or ME (movement epenthesis) and then they are connected to build a Bayesian network. The segments labeled like ME are removed from the network and the labeled like SIGN are recognized by a classifier based on Conditional Random Field (CRF) and SVM. Finally, the recognized signs are modeled like a full sentence, whose translation is done by an approach based on semi-Markov chain model. The method was able to interpret 89% of the sentences used in the tests.

Concerning analyzing images, Liu et al. (2015) proposed a method that aims at understanding image structure. The method detects the edge segments of the object and the scale from the image to build a Hierarchical Edge Tree in a top-down way. In parallel, an Appearance Tree also is built from segments extracted from the image. Then, both trees are merged, using a structural appearance pooling operation based on a Hidden Markov Model (HMM) process (Ghahramani, 2001), to build a single shape graph that represents the image. Koziński and Marlet (2014) used Markov Random Field (MRF) and Factor Graph to analyze the facade of buildings. According to Blake et al. (2011), MRF is a set of random variables described by a probabilistic model over undirected graphs that having Markov chain properties. The authors derived a Factor Graph from facade image using a ready grammar for Haussmannian buildings (Teboul et al., 2011), whose variable-nodes correspond to geometric primitives and their position in space, and factor-nodes correspond to constraints on the composition of the objects, for example, the composition of a window or wall. The Factor Graph that represents the facade image is reflected in an MRF to infer the position of the parts. The method achieved state-of-the-art performance on tasks of inferring the architectural position of components. The framework proposed by Kortylewski et al. (2019) for learning hierarchical compositional models decomposes objects into strokes, obtained by Gabor filters, which together form the object sketches. The model extends an Active Basis Model (ABM) (Wu et al., 2009), whose aim is to decompose an object into a set of Gabor elements at certain locations and orientations. These Gabor elements are linearly connected to create the observed object sketch. The ABM generalization for an object class is provided by a Compositional Active Basis Model (CABM) (Wu et al., 2009) in a probabilistic way, which has as base an MRF. The framework formulates the object sketch learning task as a compositional clustering process guided by a tree-like MRF that allows learning object small parts first, before moving to compose the object on the whole. The experiments showed that the framework outperforms other generative object models at object classification task.

Relational String Graph

Relational String Graph (RSG) is a fully connected graph $G = (V,R)$, where $V = \{v_1, v_2, \ldots, v_n\}$ are the nodes, and $R = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ are directed edges. Each node v_i denotes an edge line, which is formed from line segments $v_i = (l_1, l_2, \ldots, l_{|v_i|})$. The line segments l_j has the geometric properties: mean coordinate $mid(l_j)$, $angle\theta(l_j)$, and length $len(l_j)$, which are read from the LEM (Line Edge Map) data (Dahm et al., 2013). LEM data are lines of information that can be extracted by standard border detector like Canny (Dahm et al., 2013).

RSG is used by Dahm et al. (2013) for face alignment and recognition. They define an RSG matching algorithm that makes face alignment and recognition tasks simultaneously. Basically, the method generates an RSG from face images, whose nodes represent primitives, and analyzes possible relation correspondence between two graphs calculating the translation, rotation and scale parameters needed to convert a relationship into its equivalent. The experiments were performed using 50 face images at different rotations and scales, 25 being male and 25 female. The algorithm was able to achieve 84% accuracy. When the rotation and scale of the images were replaced it reached 64%. Zhu et al. (2013) were based on RSG to introduce the String of Feature Graphs (SFG) model to represent and infer human activities. The method models a video V of duration T like a set of feature points $V = \{f^t_{x,y} | t \in [1,T]|\}$ where $f^t_{x,y}$ represent a feature point at spatial location x, y and time index t. The video V is divided into N intervals in time

$t_0, t_1 \ldots, t_N$ and the features included in each time interval, which are extracted using Spatio-Temporal Interest Point (STIP)(Sethi and Roy-Chowdhury, 2010), are represented as F. Thus, the video is represented as $V = \{F_1, F_2, \ldots, F_N\}$ where $F_1 = \{f^t_{x,y} | t \in [t_0, t_1]\}$, $F_2 = \{f^t_{x,y} | t \in [t_1, t_2]\}$, etc. The representation of the video is done by concatenating its features F like an SFG. The matching between two videos $V^{(1)}$ divided into N_1 intervals and $V^{(2)}$ divided into N_2 intervals is performed by matching their single feature sets $\{F^{(1)}_i | i = 1 \ldots N_1\}$ and $\{F^{(2)}_i | i = 2 \ldots N_2\}$ in a spatial and temporal order. The method recognizes and localizes complex activities, even when various people are interacting.

Others

In this section, we present papers not categorized in any of the syntactic methods presented earlier, but which employ approaches to represent or recognize patterns in a syntactic way.

Lemus et al. (2015) proposed a method for representing three-dimensional (3D) objects surfaces using symbol chains. The authors digitalize voxelized surfaces and assign each face a symbol of an alphabet that has nine symbols. From this, the method obtains a graph of the image surface, whose nodes represent a face, consequently, each graph node is associated with face symbol. Each alphabet symbol represents a direction. Thus, a chain that represents a given surface is obtained by traversing the graph changing of direction when the nodes are visited to form one Hamiltonian circuit over the surface. Pistori et al. (2013) also obtained character chain from images. The proposed method detects keypoints in the images using the Speeded-Up Robust Features (SURF) algorithm (Bay et al., 2008) and clusters them using the K-means algorithm for building an alphabet, whose symbols represent each resulting cluster centers. Each keypoint is mapped to the alphabet symbol that represents the cluster center closest to it. The keypoints from the image are traversed and a character chain is derived by concatenating the symbols assigned to each point. The character chains of each training image are used to infer a grammar, employing the K-Testable algorithm, to each object class. A given image is associated with a class when the grammar of the class produces the smallest error counter by parsing its character chain. Astolfi et al. (2020) extended the model proposed by Pistori et al. (2013), but rather than infer a grammar using the character chains derived from the images, the authors used an LSTM to learn the relationship between the symbols in character chains. The model maintained stable results even when trained with few examples. Abid et al. (2018) also adopt an approach based on sequences to represent visual patterns. The method proposed aims to identify the named entity on a non-standardized postal address. The textual address is obtained from the input images and divided into parts as words and characters. The parts are submitted to a tokenization process based on a given pattern and afterward, they are embedded and inserted into a BLSTM classifier to learn the sequential pattern. The method achieved 90.44% of accuracy, a similar result to state-of-art.

Kembhavi et al. (2016) proposed a method to interpret scientific diagrams. The diagram components are detected using a Random Forest classifier from features generated by non-parametric kernel density estimates in RGB, texture and entropy spaces. The diagram components are related using a graph called Diagram Parse Graphs (DPG) to parse the diagram in a syntactic and semantic way. The syntactic analyze learns to infer the DPG which best explains a given diagram. The semantic analysis tries to respond to questions on a given diagram. For example, "From the diagram, what will lead to an increase in the population of deer? a) increase in lion b); decrease in plants; c) decrease in lion; d) increase in plants". The method uses a LSTM to learn the relationship sequence between diagram components to answer the questions. The method achieved an accuracy of 51.45% in syntactic parsing and 38.47% question answering. Deufemia et al. (2014) also proposed a method for analyzing diagrams, but in the electric circuit diagrams domain. The method detects feature points like strokes, curvature, a corner point, an end point or an overlapping point, and represents them as o_i. The feature points have information as orientation, distance to another point, direction, etc. A stroke segment which connects o_{i-1} and o_i is represented with S_{o_i}. A S_{o_i} is represented like a feature points sequence $[o_i, \ldots, o_m]$ in which each S_{o_i} is assigned a symbol label w_i (W = Wire, C = Capacitor, V = Voltage, and I = Inductor), and the whole sequence is defined like an Latent-Dynamic Conditional Random Field (LDCRF) (Morency et al., 2007). The labels from the LDCRF are clustered taking into account the geometric relationships between strokes, to obtain a symbols sequence. The method achieved an accuracy of 91.0% on an electric circuit diagrams dataset.

Isola and Liu (2013) proposed a method to parse scenes. The method receives as input an image and uses it to retrieve object segments from a similars image dictionary for use in collages. The segments are selected using the K-Nearest-Neighbors classifier and the descriptors Spatial Pyramid Matching (SPM) (Lazebnik et al., 2006) and Bag-of-Visual-Words (BoVW) (Hentschel and Sack, 2014) with HOG. The selected segments are compared with the segments of the input image to identify those that best fit semantically and explain the appearance of the input image. After, they are organized using a scene graph, which is controlled by grammar with defined rules to valid scenes. Finally, is done a random search over productions from the grammar to find for collages that cover the input image.

Rodríguez et al. (2017) proposed a conceptual analysis of three major language grammar properties: meaning, synonym, and polysemy; and how they can be adapted to the computer vision and image understanding domain. Initially, relevant topics are identified using the BoVW approach. The topic is modeled based on the meaning of the visual word within the set, in which a weight is assigned to define the importance of the visual word. This approach is similar to the importance of nouns and adjectives in a sentence. Topics are formed with a higher level of generalization, modeling partially shared meanings among words. The relationships among the topics are analyzed based on the concepts of synonymy and polysemy. Visual words are synonymous when they share similar characteristics. The synonyms are modeled according to a set of criteria that allows defining relations of characteristics among the visual words. The relationships of the synonyms are expressed using a graph, which forms a group of visual words with the same meaning. In addition to grouping similar visual words, the synonymy relations provide additional information to each visual word. Visual words are polysemous when they belong to at least two different topics. By linking visual words to topics and in terms of synonyms and polysemy, the authors conceptually derive BoVW for a model called Visual Grammar, bringing visual words closer to language concepts that are easily understood by humans (meaning, synonymy, and polysemy). Finally, they used the cosine similarity measure as the way of measuring the similarity of two bags of words vectors obtained from two images.

Recently, the hierarchical feature learning using the CNNs learning structure has emerged in the design of models. Tabernik et al. (2016) introduces the notion of hierarchical compositionality, provide usually by compositional models, into the CNNs learning structure. The authors inserted basic units into the CNN in order to regularize the convolution filters to be sparse and, consequently, to expose an explicit structure of compositions provided by spatial clustering of the feature activations in the network layers. The model has similar performance to standard CNNs on classification tasks but has the advantage of reducing the number of features and improve training times.

3.5 Discussion

CFGs can represent a large number of structural visual patterns using compositional rules of high-level. They can build dependencies between primitive patterns in nested structures that allow future predictions on sequence data. Such features allow that CFG to be a natural choice to predict actions in video sequences (Tian et al., 2015; Le et al., 2017; Feng et al., 2014; Kuehne et al., 2016; Pirsiavash and Ramanan, 2014; Kuehne et al., 2017). However, the CFG has a limited semantic as its formalism expresses syntax, i.e., it is composed of a set of rules that govern a linear structure of dependence between primitive patterns. On the other hand, an AG, which is a

CFG increased with attributes and attribute evaluation rules, allows easily to describe primitive patterns and the relationships between them, promoting more semantic in the composition rules of the model (Slonneger and Kurtz, 1995); for this reason, GAs are suitable for scene understanding tasks (de las Heras et al., 2015; Boulch et al., 2013; Liu et al., 2018b,b). The disadvantage is that the AG models are computationally more demanding than CFG models because it can require one or more passes to evaluate the attributes.

Some studies used SCFGs motivated by the fact that the stochastic approaches provide a way to assign a probability to a particular sequence of data. This non-deterministic process allows deriving a much longer and elaborate sequence of primitive data for compact representation of well-understood data. This probabilistic compositional model allows predicting action sequences (Vo and Bobick, 2014, 2016; Lee et al., 2015; Tayyub et al., 2018; Qi et al., 2017; Fire and Zhu, 2017), tracking objects (Xu et al., 2018; Li et al., 2015; Lu et al., 2014), scene understanding (Chua and Felzenszwalb, 2016; Zhao and Zhu, 2013; Walton et al., 2017), and dealing with partial occlusion of objects (Li et al., 2014), besides being able to be totally incorporated into an *And-Or* graph. The SCFGs can be extended to 2D-SCFG by adding in each production rule the spatial relationship between the primitive data, which makes them more accurate in tasks whose spatial relationship between the primitive data is a determining factor, such as mathematical expression analysis (Álvaro et al., 2014, 2016). The disadvantage is that the SCFG models are computationally more demanding than CFG models (Moore and Essa, 2002). However, in most cases, they are superior because the probability assigned to each production rule offers a quantitative basis for ranking and pruning parses (Moore and Essa, 2002).

SGs have properties suitable to operate on facade analyzes (Weissenberg et al., 2013; Koziński et al., 2015a,b; Teboul et al., 2013; Gadde et al., 2016). The rule components are shapes like points, lines, circles, squares, or triangles, whose hierarchical composition naturally build facade parts like windows, walls, roofs, or doors. Besides, the production rules deal with the shapes as nonatomic components, i.e., they aren't predefined in grammar, which allows shapes to be freely decomposed and recomposed to form new shapes (Stiny and Gips, 1971).

Another method well explored from a syntactic perspective were those based on Markov Models, mainly to map spatial and temporal relationships, respectively, on images (Liu et al., 2015; Koziński and Marlet, 2014) and videos (Krüger and Herzog, 2013; Windridge et al., 2015; Liu et al., 2016; Kong and Ranganath, 2014). In most studies, Markov Models were used to represent probabilities distribution about sequences of objects parts or videos. Typically, the approaches detect primitives using shallow algorithms and relationships them spatially or temporally using symbols chain. The relationship among primitives is performed using probability distribution or heuristic rules. They are related to inferring something, such as the next action in a video or part of an object. However, Markov Models are susceptible to some limitations compared to CFGs and SCFGs. Markov Models, which are essentially finite state machines, are suitable for modeling a single hypothesis or a set of them in parallel, but as the variations increase in the model, it becomes more complex to manage additional hypotheses using only a finite state model (Moore and Essa, 2002).

3.5.1 Advantages and Disadvantages of Using Syntactic Methods in Computer Vision

The main advantage of syntactic methods applied to computer vision is their high degree of modularity. Syntactic methods are natively hierarchical and compositional, i.e., primitives as parts of objects and atomic actions of videos can serve as reusable basic units to build models based on hierarchical and semantic decomposition that exhibit large amounts of structural variations. Such characteristics impose an efficient representation of the learning model that allows faster inference, feature sharing, and spatial and contextual reasoning. These models demonstrate high generalization capabilities for a wide range of applications, such as pose estimation, object tracking, scene understanding, object parsing, actions recognizing, and image classification.

Concerning the disadvantages, the most important is the need to segment the target object from the background. Besides, the syntactic methods are dependent upon image preprocessing techniques, such as local feature extractors, segmenters, edge detectors, corners, or lines, required to extract reusable primitives from the image. Any failure to obtain information from the image can affect the representation and pattern recognition. Another disadvantage is that usually hierarchical and compositional models can only be learned if their hierarchical structure is provided previously, i.e., they are very domain dependent.

Models that adopt syntactic pattern recognition can be classified as generative models. The inference in these models is via inverted indexing, whose aim is to compute a parsing for a given sample as its interpretation or one of its closer interpretations, which allows a faster inference (Tabernik et al., 2016). These models also offer decoupled data representations providing interpretability and straightforward reconstruction from observations of features (Kingma et al., 2014). Also, in these models, the learning is performed by co-occurrence needing fewer data for training (Ng and Jordan, 2001). On the other hand, CNNs, which are discriminative models, need more data for training to be more accurate than generative models (Ng and Jordan, 2001). The CNNs also fail to expose spatial relationships between high-level parts (Hinton et al., 2011) that hinder it from dealing with partial occlusion and object missing parts (Tabernik et al., 2015). Although these approaches are the computer vision state-of-the-art, they are seen as black-boxes that only receive inputs and produce learning models hard to interpret because of their complexity.

3.5.2 Directions for Future Research

Although the CNNs have become dominant in many computer vision tasks, they still present some issues that need to be addressed. For instance, they have limited capacity to represent orientational and relative spatial relationships between parts at the object level (Hinton et al., 2011), have difficulty learning semantic relationships between objects at the scene level, can not adequately deal with partial occlusion of objects (Tabernik et al., 2015). These issues can be addressed by taking into account hierarchical compositional representations of features, which remains an open issue for models based on CNNs. There are some attempts to model hierarchical compositional relationships inside of the internal knowledge representation of the CNNs (Hinton et al., 2018; Sabour et al., 2017). However, they do not explicitly represent hierarchical relationships between features as well as their relative spatial relationships in a syntactic way. On the order hand, some studies propose to regularize the convolution filters of the CNNs and connect them sparsely to model hierarchical relationships between features, embedding syntactic approaches into its structure(Xing et al., 2020, 2019; Tabernik et al., 2016). To incorporate syntactic approaches into the learning structure of the CNNs is one of the main directions for future research, as this can pave ways for building models that learn visual features from a hierarchical and compositional perspective. These models could allow feature sharing between objects as well as spatial and contextual reasoning between subparts, parts, and the whole.

3.6 Conclusions

The papers surveyed by our systematic review provided a comprehensive overview about the use of syntactic pattern recognition in computer vision. Even with the increasing use of techniques such as deep learning, syntactic methods are still widely explored in computer vision. Most studies use syntactic methods to represent a high-level structure that relates object parts in images, objects in scenes, and actions in videos. In these studies, objects or parts of them, as well as actions or sub-actions in videos were identified using deep learning or shallow algorithms and related by syntactic methods. On the other hand, few studies have used only syntactic methods to represent and recognize patterns, which can be viewed like a gap in this research area. We note that syntactic methods naturally to represent object appearances in a hierarchical and compositional way, considering logical or probabilistic rules of composition. This suggests that from a finite set of primitive structures, multiple appearances can be achieved for a given object through few configurations, what gives the possibility to extrapolate the object appearances beyond the examples that were offered by the training set. Consequently, it may allow the object recognition method sensitivity not to be limited to the training set.

3.7 Paper appendix: Summary of the Papers Analyzed

In the list below is shown a summary of the 71 analyzed papers grouped by syntactic methods. We highlight the objective and the result achieved for each study.

- **Context-free Grammar (CFG)**
  - Feng et al. (2014): Fall detection, monitoring elderly people in a house care environment. It achieved high detection accuracy and no false positives.
  - Pirsiavash and Ramanan (2014): Models the video's temporal structure for detecting human activities. Accuracy of 62% at labeling frames and segment detection 22%.
  - Tian et al. (2015): Detecting vehicles for complex traffic surveillance. The method adapts to partial occlusion.
  - Jiang and Ma (2015): Combine models based on deformable parts for human detection.Accuracy of 57% in the experiment on PASCAL VOC dataset.
  - Gonfaus et al. (2015): Extending the DPM for pose estimation to general object class detection. Outperforms the state-of-the-art results for several object categories.
  - Demir et al. (2015): Procedural modeling of the 3D point cloud of urban structures. It didn't show results.
  - Kuehne et al. (2016): Recognizing human activities in video based on the sequence of actions. Outperforms the state-of-the-art approaches for larger datasets.
  - Kuehne et al. (2017): Recognizing human activities from video transcripts, derived from Kuehne et al. (2016). Accuracy of 62% at labeling frames and segment detection 43%.
  - Le et al. (2017): Framework applied to the driver monitoring tasks. Have better performance than other state-of-the-art detection and segmentation methods.

  - Julca-Aguilar et al. (2017): Method for the recognition of online mathematical expression. Obtain state-of-the-art accuracy in recognition of mathematical expressions and flowcharts.

- **Stochastic Context-free Grammar (SCFG)**
  - Song et al. (2013): Presenting a method of learning from weakly annotated data for object detection. Tested on detection benchmarks of 20 object classes and outperformed state-of-the-art.
  - Rothrock et al. (2013): Model to represent the appearance and geometry variation of human pose into modular parts. Better performance than state-of-the-art, the accuracy of 79%.
  - Zhao and Zhu (2013): Model to hierarchical decompositions of objects for analyzing indoor functional objects in scenes. The approach obtained good performance for the functional object recognition task.
  - Vo and Bobick (2014): Body pose classification in complex activity. Segmentation accuracy of 58% and in segmentation in offline mode, the accuracy is 91.8%.
  - Lu et al. (2014): The model simultaneously tracks, learn and parse objects in video sequences. It outperforms the state-of-the-art tracking algorithms including two trackers based on CNN.
  - Li et al. (2014): Approach to learn a hierarchical model for X-to-X occluded objects detection (e.g., car-to-car and person-to-person). The method obtains comparable or better detection performance to state-of-the-art deformable part-based methods.
  - Lee et al. (2015): Representing the temporal structure of activities and encode the history of the observed actions. The method achieved a good activity detection performance on a structured activity dataset of concurrent multiple human objects of high-resolution video.
  - Li et al. (2015): Presenting a part-based pedestrian detection algorithm for complex traffic surveillance environments. The method outperforms other successful approaches with high reliability and robustness in complex environments.
  - Vo and Bobick (2016): Body pose classification, derived from segmentation. Accuracy of 60.2% and in segmentation in offline mode, the accuracy is 91.8%.

  - Chua and Felzenszwalb (2016): Method that represented objects in scenes using a hierarchical structure defined by composition rules. The resulting algorithm is efficient, robust, and broadly applicable.
  - Qi et al. (2017): Method to predict future human activities. The effectiveness of the model on both detection and anticipation human activity is respectively 77% and 56.5% of average precision.
  - Fire and Zhu (2017): Infer the causal-effect relationship between status object. The results demonstrate the effectiveness of the method to infer the causal-effect relationship between objects over time.
  - Walton et al. (2017): A method to represent and learn contextual relationships with applications to thes Naval Tactical Domain. Not showing results.
  - Tayyub et al. (2018): Approach to represent complex long-term human activities. In one of the datasets achieved 93% accuracy.
  - Xu et al. (2018): Method to describe the transitions of the subject, indicating in the video his the position and visibility status. Outperforms the alternative trackers and can recover complete trajectories of humans in complicated scenarios with frequent human interactions.
  - Jiang et al. (2018): Method to learn the layout of the indoor scenes. Comparable to state-of-the-art.
  - Qi et al. (2018): Method to learn the layout of the indoor scenes. Similar to state-of-the-art.
  - Wang et al. (2018): Method to classify fashion images. It overcomes the state-of-the-art.
  - Xing et al. (2019): A method that uses the first convolutional layers of a CNN as a feature descriptor and an AOG to represent a learning model.
  - Xing et al. (2020): An approach that induces an AOG model by sparsifying generator network for generating image synthesis. The approach achieved good results in image synthesis and reconstruction.

- **Two-dimensional Stochastic Context-free Grammar (2D-SCFG)**
  - Martinovic and Van Gool (2013): Representing building facades. The method achieved an accuracy of 74.82% in the facade analysis.

  - Álvaro et al. (2014): A formal model for the recognition of on-line handwritten mathematical expressions.In a contest of mathematical expression recognition, and it obtained the best results at different levels.
  - Álvaro et al. (2016): Mathematical expression recognition. Performance of 82.2% of precision, outperforming to the state-of-the-art.
  - Fang et al. (2018): Approach to 3D human pose estimation. Superior performance over the 16 state-of-the-art methods.

- **Coordinate Grammar (CG)**
  - Ayeb et al. (2015): Arabic mathematical formula recognition. The method achieved 91% of accuracy.

- **Attribute Grammar (AG)**
  - Boulch et al. (2013): A method to interpret the semantic information of a 3D model. The authors consider that the method obtained excellent results.
  - Choe et al. (2013): Retrieving videos containing similar complex activities with the query video rather than finding visually similar videos. One of the experiments achieved 80% of videos correctly retrieved.
  - Liu et al. (2014): Parsing outdoor scenes images into semantic surfaces, and recovering its 3D model simultaneously. The best result archived was 79.53% precision.
  - de las Heras et al. (2015): Representing the structure of a floor plan. In one of the datasets achieved 96.37% accuracy.
  - Park and Zhu (2015): Model to tackle the tasks of attribute recognition, pose estimation and part localization jointly. The method achieved better results against the state-of-the-art.
  - Liu et al. (2018b): Scene understanding. The method achieved state-of-the-art.
  - Liu et al. (2018a): Model to track humans in the video. Outperforming state-of-the-art tracking methods.

- **Shape Grammar (SG)**
  - Weissenberg et al. (2013): Facade structure understanding. Accuracy of 87%.
  - Teboul et al. (2013): Facade structure understanding. The method achieved the state-of-the-art results in a fraction of the time required by other methods.

- Koziński et al. (2015a): Facade structure understanding. Outperforming state-of-the-art on a number of facade segmentation datasets.
- Koziński et al. (2015b): Segmentation of Building Facades. It yields state-of-the-art performance on standard datasets.
- Zieliński et al. (2015): Detecting erosions and osteophytes in bone contours. The sensitivity and the specificity of detecting lesions are around 70%.
- Ikehata et al. (2015): Indoor scene representation. Outperforming state-of-the-art on room segmentation algorithm.
- Gadde et al. (2016): Facade segmentation. Outperforming state-of-the-art on four different datasets segmentation algorithm.

• **Context-sensitive Grammar (CSG)**

  - Zarchi et al. (2016): Scene understanding. Outperforming the state-of-the-art in recognizing image concepts.
  - Park et al. (2018): Model for explicitly representing human poses. The method achieved state-of-the-art in precision and performance.
  - Li et al. (2019): Objects classification and recognition. Obtaining better performance than ResNet and its variants on three benchmarks.

• **Stochastic Context-sensitive Grammar (SCSG)**

  - Pei et al. (2013): Model to learn the interactions between people and objects in the scene. Goal inference accuracy was 90%, and the intent prediction accuracy was 87%.

• **Markov Models (MM)**

  - Krüger and Herzog (2013): Tracking humans and recognizing human action. Effective on synthetic and on real image sequences using human-upper body single arm actions that involve objects.
  - Kong and Ranganath (2014): Model for recognizing the continuous gestures of American Sign Language (ASL). The model achieved an average accuracy of 89% in decoding continuously signed sentences.
  - Koziński and Marlet (2014): Facade analyze. Results are comparable to the state-of-the-art.

  - Windridge et al. (2015): Model for tennis video annotation. Event prediction accuracy of 68%.
  - Liu et al. (2015): Hierarchical shape parsing for understanding image structure. Performance compared to the state-of-the-art.
  - Liu et al. (2016): Model to recognize complex activities. One of the tests achieved an F1 score of 80%.
  - Kortylewski et al. (2019): Generative object model to classify objects. The approach outperforms other models at object classification task.

- **Relational String Graph (RSG)**
  - Dahm et al. (2013): Face alignment and recognition. The method achieved 84% accuracy.
  - Zhu et al. (2013): Representing human activity. The method is effective to recognize and localize complex activities.

- **Others**
  - Pistori et al. (2013): A strategy for applying grammatical inference to image classification problems.
  - Isola and Liu (2013): An approach to parsing scenes. Accuracy of 70%.
  - Deufemia et al. (2014): Recognition of hand-drawn electric circuit. Accuracy varying between 81.3% and 91.0%.
  - Lemus et al. (2015): Method for representing three-dimensional (3D) simple objects surfaces using chain code. Not showing results.
  - Kembhavi et al. (2016): Method for interpreting and reasoning in the context of science diagrams. On the syntactic parsing, the method achieved 51.45% and on question answering 38.47% of accuracy.
  - Rodríguez et al. (2017): Method for representing visual words as meaning, synonym, and polysemy, and how they can be adapted to computer vision. The method achieved similar accuracy to the state-of-the-art.
  - Abid et al. (2018): Method to identify the named entity on a non-standardized postal address. The method achieved 90.44% of accuracy.

- Tabernik et al. (2016): A Hierarchical feature learning model based on CNN learning structure. The model improved training times.
- Astolfi et al. (2020): The model derives character chains from the images and uses an LSTM to learn the relationship between the symbols in chains. The model achieved good results on small data sets.

Chapter 4

Combining Syntactic Methods with LSTM to Classify Soybean Aerial Images

Authors[2]: Gilberto Astolfi, Marcio Carneiro Brito Pache, Geazy Vilharva Menezes, Adair da Silva Oliveira Junior, Gabriel Kirsten Menezes, Vanessa Aparecida de Moares Weber, Everton Castelão Tetila, Nícolas Alessandro de Souza Belete, Edson Takashi Matsubara and Hemerson Pistori.

Abstract: Syntactic methods in computer vision represent visual patterns in a hierarchical and compositional perspective, which is converted to strings. Long Short-Term Memory (LSTM) is able to learn patterns in sequences. In this paper, we propose a syntactic approach to represent visual patterns as sequences of symbols and we use an LSTM as a classifier to learn the relationship between the symbols in sequences. An extensive experimental evaluation using aerial images from a soybean field captured by Unmanned Aerial Vehicles has been conducted to compare our method with two deep learning architectures, one syntactic method, and one shallow learning algorithm. The results achieved by the proposed method maintain stability even when trained on small data sets, suggesting that representing visual patterns in a compositional way, repeating primitives, may be a viable alternative when there is only a limited number of samples for training.

[2]Astolfi, G., Pache, M. C. B., Menezes, G. V., Junior, A. d. S. O., Menezes, G. K., de Weber, V. A. M., Tetila, E. C., de Souza Belete, N. A., Matsubara, E. T., and Pistori, H. (2020). Combining syntactic methods with lstm to classify soybean aerial images. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5.

4.1 Introduction

Unmanned Aerial Vehicles (UAVs) are important tools for precision agriculture, since they can be helpful in various areas from planning up to harvest. One of the advantages of using UAVs is the possibility of covering large areas quicker and cheaper, allowing early identification of crop problems, such as pest attacks, diseases and weed infestation.

Some researchers have used UAVs and computer vision techniques to identify problems in crops. Tetila et al. (2017) used six shallow learning algorithms to identify soybean leaf diseases in images captured by a UAV in different heights, including 1, 2, 4, 8, and 16 m. Bah et al. (2018) proposed a method for detecting weeds but they used Convolutional Neuronal Networks (CNNs) and unsupervised training instead of shallow learning algorithms. Ferreira et al. (2017) also adopted a supervised training approach to detect weed infestation. Amorim et al. (2019) combined CNNs and a semi-supervised training approach for detecting and identifying herbivorous pests on soybean leaves. Potena et al. (2017) exploited two different CNNs to process RGB and near-infrared (NIR) images from winter wheat fields with heavy leaf occlusion.

The reported results achieved high accuracy but used a large number of training examples. Dyrmann et al. (2017) also used large training sets; the authors had to annotate 17,000 images from winter wheat fields to achieve an accuracy rate of 86.6% on weed detection. However, these methods can produce poor results when few examples are available for training by having discriminative characteristics (Ng and Jordan, 2001). In contrast, descriptive methods tend to be better when fewer training examples are available (Ng and Jordan, 2001). The methods that use simple primitives to represent patterns in a hierarchical and compositional manner, giving us the possibility to represent various patterns from a small set of examples, have descriptive characteristics. The syntactic approaches (Chanda and Dellaert, 2004) are included in this method category since they can represent patterns from a compositional viewpoint, making the pattern representation more transparent and interpretable. They often represent primitives using alphabet symbols and complex patterns organizing these primitives through well-defined composition rules.

In this paper, we propose a classification model, which we call Syntactic LSTM (SLSTM), that uses a syntactic approach to represent visual patterns of objects as strings and an LSTM (Greff et al., 2017) to classify them. The SLSTM identifies primitives related to object parts in the image and represents them using alphabet symbols. These symbols are related in a compositional way to form strings that are inputted into an LSTM classifier. The SLSTM has a descriptive characteristic, since it discovers primitives in the structure of the objects and repeats them in different visual patterns, allowing us to deal with situations in which building large datasets is a time-consuming and costly process. This situation is observed in soybean crops, since it is necessary to cover large extensions of land to obtain a single example of disease outbreaks, due to the intensive application of pesticides.

We evaluated the SLSTM on the classification task of problems that occur in soybean crops, such as weed infestation and disease outbreaks, using aerial images captured by a UAV. We compared the SLSTM with Support Vector Machines trained on a Bag-of-Visual-Words (BOVW+SVM) (Hentschel and Sack, 2014), a syntactic method that combines keypoint detection and grammatical inference (Pistori et al., 2013), ResNet-50 (He et al., 2016), and Xception (Chollet, 2017) using small training sets. Our goal is to investigate whether the classification models maintain the performance when trained with increasingly smaller datasets. The precision rate achieved by SLSTM maintains stability even when the training set decreases significantly.

This paper makes the following contributions: a) provide a new approach for representing visual patterns of objects as strings in a compositional way; b) show that the visual pattern representation in a compositional way, repeating primitives, can be a viable alternative when there is only a limited number of samples for training.

4.2 Proposed Method Overview

The SLSTM consists of some stages. First, we use Scale Invariant Feature Transform (SIFT) (Lowe, 2004) to detect keypoints from each object (superpixel image) in the training set. We consider these keypoints as primitives. Then, we use k-means to cluster similar primitives and define an alphabet, whose symbols are used to represent each clustering of primitives. Now, each primitive is associated with the alphabet symbol that represents the cluster to which it belongs (see section 4.2.1). The second stage consists of connecting primitives contained in each object to derive a string from each one (see section 4.2.2). Finally, the strings, whose composition represents the visual pattern of the object, are inputted in the LSTM for training (see section 4.2.3) or classification (see section 4.2.4). We employ LSTM because it has special units called memory cells that can keep information in memory for long periods of time, which makes it efficient in processing very long sequences (Greff et al., 2017). Figure 4.1 shows the SLSTM structure and training process.

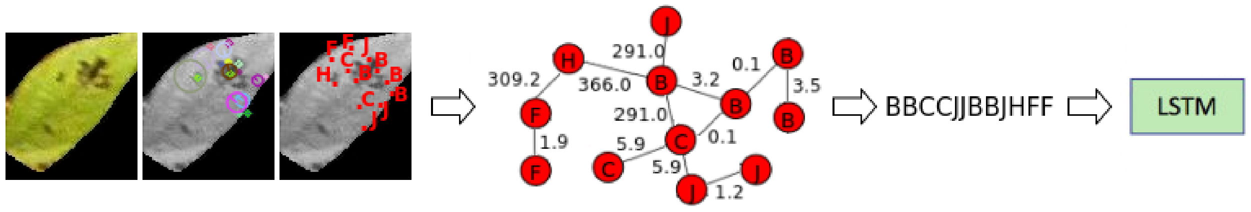

Figure 4.1: Overview of SLSTM structure and training process. In the first stage, the SLSTM detects keypoints and represents them syntactically as primitives using alphabet symbols. After this, the SLSTM semantically and structurally relates the primitives in a graph from which a string is derived to represent the visual pattern of the object. Finally, the string with the visual pattern of the object encapsulated is inserted into an LSTM classifier for training.

4.2.1 Identifying and Representing Primitives

This stage of the SLSTM aims to identify and represent primitives syntactically, i.e., to associate primitives of the objects to alphabet symbols. According to Fu and Rosenfeld (1976), primitives are non-divisible basic patterns of objects. Based on this definition, we consider as primitives the keypoints of the objects detected by SIFT. The SIFT detects keypoints from each object (superpixel) of the training dataset and generates for each detected keypoint a feature vector containing 128 values and an xy coordinate system.

All feature vectors extracted from objects of the training dataset are clustered using k-means, and the resulting k cluster centers form an alphabet of size k. For example, when the feature vectors of the keypoints are clustered using $k = 10$, the resulting alphabet is the set of symbols $\Sigma = \{A, B, C, D, E, F, G, H, I, J\}$, where each cluster center is represented by an alphabet symbol. Now, each keypoint can be represented by an alphabet symbol corresponding to its nearest cluster center. Since we are treating keypoints as primitives, this strategy allows us to represent object primitives in a syntactic way.

4.2.2 Mapping Objects to Strings

This stage of the SLSTM aims to represent the visual pattern of the object syntactically, deriving a string from each object of the training dataset. The strategy is to connect the primitives of a given object to build a graph and then traverse it to derive a string.

We define the Visual Weighted Graph (VWG) to encapsulate structural and semantic relations among the object keypoints. A VWG is a two-element tuple $G = (V(G), E(G))$, where $V(G)$ is a four-element tuple $V(G) = (\lambda, \vec{\alpha}, P, c)$ that represent objects keypoints, i.e., primitives. λ is the label of the node represented by an alphabet symbol, $\vec{\alpha}$ is a feature vector containing 128 values correspondent to the keypoint, P is the keypoint center coordinates x and y, and c is the nearest cluster center to the keypoint. $E(G)$ is the edges set, where each edge $e_i \in E(G)$ is defined by a three-element tuple $e_i = (v_j, v_k, w)$, such that $v_j, v_k \in V(G)$ and $w \in \mathbb{R}$ denotes the weight of e_i.

In VWG two nodes v_i and $v_j \in V(G)$ are adjacent if v_i is the node nearest to v_j or v_j is the node nearest to v_i, so a node only connects to its nearest neighbor. The distance between any two nodes is calculated based on spatial and semantic distance between them. The distance measurement $D_S \in \mathbb{R}$ between v_i and v_j is carried out as follows:

$$(v_i^c, v_j^c) = \left\| v_i^c - v_j^c \right\|_2 \tag{4.1}$$

$$d(v_i^P, v_j^P) = \left\| v_i^P - v_j^P \right\|_2 \tag{4.2}$$

$$D_S = (\delta \cdot d(v_i^P, v_j^P)) + ((1-\delta) \cdot d(v_i^c, v_j^c)), \tag{4.3}$$

where $\delta \in \mathbb{R}$ gives the relative importance between the semantic distance $d(v_i^c, v_j^c)$ and the spatial distance $d(v_i^P, v_j^P)$. When δ is a high value, more relevance is given to the spatial distance between nodes. Now, D_S becomes the edge weight w between v_i and v_j. In order to get a graph from an object, the algorithm selects a node v and calculates the distances D_S from v to all other nodes $V(G) - \{v_i\}$, connecting v to the node with the shortest distance D_S. A new node of $V(G) - \{v_i\}$ is selected and the process is repeated until each node is linked to its nearest neighbor. After this, the VWG is traversed from the node that is closest to the object center to derive a string. The algorithm always chooses a lower weight edge to move from one node to another. The purpose is to visit all VWG nodes and concatenate λ of each node to build a string. With this strategy, we can represent visual patterns of objects syntactically, i.e., deriving a string from an object through composition rules.

4.2.3 Learning Visual Patterns of Objects

At this stage, the SLSTM has already syntactically represented the visual pattern of each object of the training set. The next step is to train an LSTM with the strings derived from each object. To that end, the strings are inserted into the LSTM as a tuple $(class, s)$, where $class$ represents a category and s, a string.

4.2.4 Objects Classification

In this section, we show how to recognize visual patterns of objects encapsulated in strings. Given an object (superpixel) for classification, the first step is to detect its keypoints using SIFT. The second step is to associate each keypoint extracted from the object with an alphabet symbol built with the feature vectors of the keypoints collected during the training phase (see section 4.2.1). As the alphabet is represented by a k-means model and each cluster center is associated with an alphabet symbol, the feature vectors are used to predict the alphabet symbols of their respective keypoints. Then, the object keypoints are converted into a VWG from which a string is derived, as shown in section 4.2.2. Finally, the string is inputted into the LSTM for classification.

4.3 Experimental Setup

In this section, we evaluate the SLSTM on the classification task of problems that occur in soybean crops, such as weed infestation and disease outbreaks. Our goal is to compare the performance of the SLSTM with a syntactic method that combines keypoint detection and grammatical inference proposed by Pistori et al. (2013) that we call GI, ResNet-50, Xception, and BOVW+SVM using small training sets.

For capturing the images, we use a DJI Phantom 3 Professional model with integrated Sony EXMOR 1/2.3" camera in manual mode at an average altitude of 4 meters above ground level. The captured images were segmented using the Simple Linear Iterative Clustering algorithm (SLIC) (Achanta et al., 2010) (see Figure 4.2) to build a dataset with 400 superpixel images, arranged in a stratified way into four classes (see samples in Figure 4.3): *healthy soybean*, *diseased soybean*, *soil*, and *weeds*. We conducted an experiment on the dataset in which we maintained 70% of the images for testing and gradually reduced the training set size from 30% to a minimum of 5%, totaling six tests for each method. For instance, in the first test, we randomly selected 70% of the images for testing and 30% for training (30 images per class); in the second test, we randomly selected 70% of the images for testing and 25% for training (25 images per class), and so on until selecting 70% of the images for testing and 5% for training (5 images per class). Note that we did not vary the number of images in the testing set, i.e., we always used 280 images for testing (70 per class). Moreover, we carry out ten repetitions for each of the tests and produced averages for accuracy, precision, recall, and F-measure.

For the ResNet-50 and Xception architectures, we used a fine-tuning strategy with all layers initialized with Imagenet weights, as well as the Adagrad optimizer (Duchi et al., 2011) at their default values, dropout rate set at 0.5 and early-stopping to prevent overfitting (Srivastava et al.,

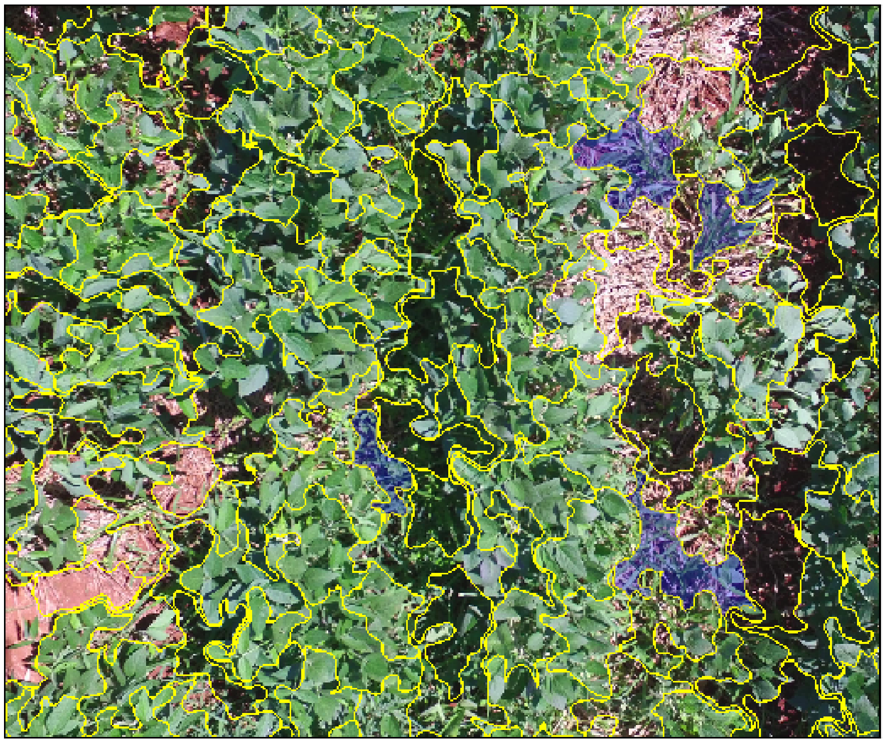

Figure 4.2: Image captured by the UAV DJI Phantom 3 Professional model, blue superpixels are weeds class samples.

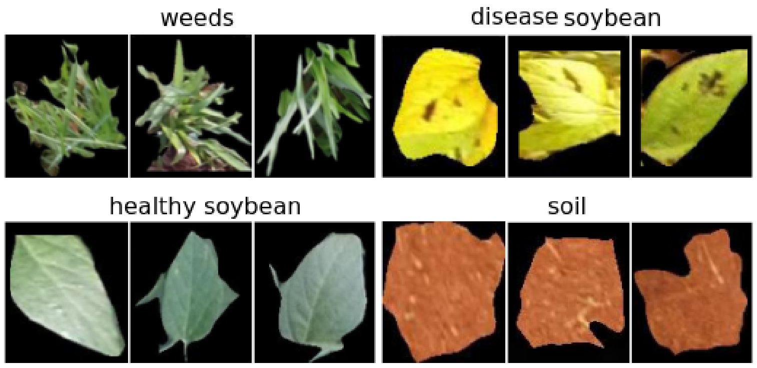

Figure 4.3: The image presents three superpixel samples from each class that compose the soybean dataset.

2014), and the learning rate at 0.001. Additionally, we varied the batch size at 4, 8, 12 and 16 in both CNNs and trained them with 50 epochs. In each training set arrangement, i.e., after reducing the training set by 5%, we performed the data augmentation. Augmentation includes horizontal flipping, random rotation by +30/-30°, rescaling factor set to $3.92 \cdot 10^{-3}$, zoom between 70% and 130%, and percentage of the image size to width and height shift at 30%.

For the SLSTM, we varied the alphabet size from 32 to 64 with an increment of 4, totaling nine experiments with different alphabet sizes (32, 36, 40, 44, 48, 52, 56, 60, and 64), i.e., nine different

Table 4.1: Metrics according to the average of 10 repetitions for Xception (1), ResNet-50 (2), BOVW+SVM (3), GI (4), and SLSTM (5). SLSTM, BOVW+SVM, and GI with alphabet size set at 44 and SIFT contrast at 0.04. ResNet-50 and Xception with batch size set at 16.

Image per class		Precision					Recall					F1-score					Accuracy				
train	test	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
5	70	41.6	11.9	75.4	43.9	**82.0**	38.6	22.9	71.9	33.7	**79.9**	35.4	12.6	72.9	27.7	**79.6**	38.8	23.1	72.6	33.8	**80.2**
10	70	66.3	11.2	79.8	41.5	**81.9**	61.9	25.1	78.3	36.1	**80.0**	60.3	12.1	78.4	30.7	**79.4**	62.4	25.3	78.4	35.4	**80.0**
15	70	74.4	15.8	80.9	42.0	**81.6**	71.3	25.9	80.2	36.3	**80.9**	70.3	13.4	80.0	33.1	**81.0**	71.1	26.4	80.0	36.1	**81.4**
20	70	80.7	11.8	81.5	39.6	**83.3**	77.9	25.7	81.2	33.9	**82.9**	78.2	13.7	80.8	30.8	**83.0**	78.6	25.9	80.6	34.2	**83.1**
25	70	80.2	16.9	84.0	42.6	**84.5**	78.2	27.9	82.9	36.3	**84.1**	77.6	14.9	83.4	33.7	**84.2**	78.2	27.5	83.2	36.0	**84.3**
30	70	82.9	18.2	83.9	42.6	**85.3**	81.1	25.8	83.2	35.5	**83.7**	80.8	14.1	83.5	32.4	**84.3**	80.8	26.1	83.4	34.9	**84.6**

cluster sizes. Before defining this value range, we carried out preliminary tests by varying the alphabet size from 8 to 256, doubling its size in each test (8, 16, 32, 64, 128, and 256), and the range from 32 to 64 showed the best results. We also used the values 0.04 and 0.05 for the SIFT contrast and the values 0.9 and 1 for the relative importance. The SIFT contrast determines the keypoints number detected by SIFT, and the relative importance determines the importance of spatial distance for connecting two nodes in the generated VWG from each superpixel image. The architecture of the LSTM model consists of three layers, including: embedding layer that receives sequences with a length that can vary from 40 to about 1200 (average length of 243) as input and has 32 output dimensions; an LSTM layer with 100 units and dropout rate set to 0.5 and; a dense layer with four nodes with a softmax activation as the output. Moreover, we used Adam optimizer (Kingma and Ba, 2015) at their default values, set the batch size at 64, trained the model with a total of 50 epochs, used sparse categorical cross-entropy for the loss function, and finally, we also used early-stopping to prevent overfitting. Like in SLSTM, we configured BOVW+SVM and GI models with alphabet size from 32 to 64, with an increment of 4, and values to SIFT contrast of 0.04 and 0.05.

4.4 Results and Discussion

Table 4.1 presents the best results obtained by the models on the different data arrangements. The SLSTM, BOVW+SVM and GI achieved the best result with alphabet size set at 44 and the SIFT contrast at 0.04, and the ResNet-50 and Xception architectures with batch size set at 16. The models achieved the best results when trained with the largest training sets. The comparison with the results of the other models shows that the SLSTM presents a better performance, especially in precision, i.e., in the proportion of true positives concerning the total of predicted positives.

Figure 4.4 shows the comparison of the results achieved by the models using the precision metric presented in Table 4.1. We performed an analysis of variance with Kruskal-Wallis on a 0.05 level of significance using the precision as metric to determine if the models maintain the average performance between the different training sets. We adopted the Kruskal-Wallis test because, in general, the average precisions didn't present normality/homogeneity of variance. The test for SLSTM (alphabet size = 44) resulted in a p-value of 0.156, therefore, we have no evidence that there is a statistically significant difference in its average performance. On the other hand, the test for BOW+SVM resulted in a p-value of 0.0001, which indicates a statistically significant difference in its average performance. There is evidence that SLSTM has a small advantage over BOW+SVM when compared to precision. This may be related to the way in which the models process the features obtained from the images. BOW+SVM only counts the primitives (visual words) that occur in the images, and SLSTM in addition to observing the occurrence of the primitives in the images (an implicit count) also analyzes the relation between them.

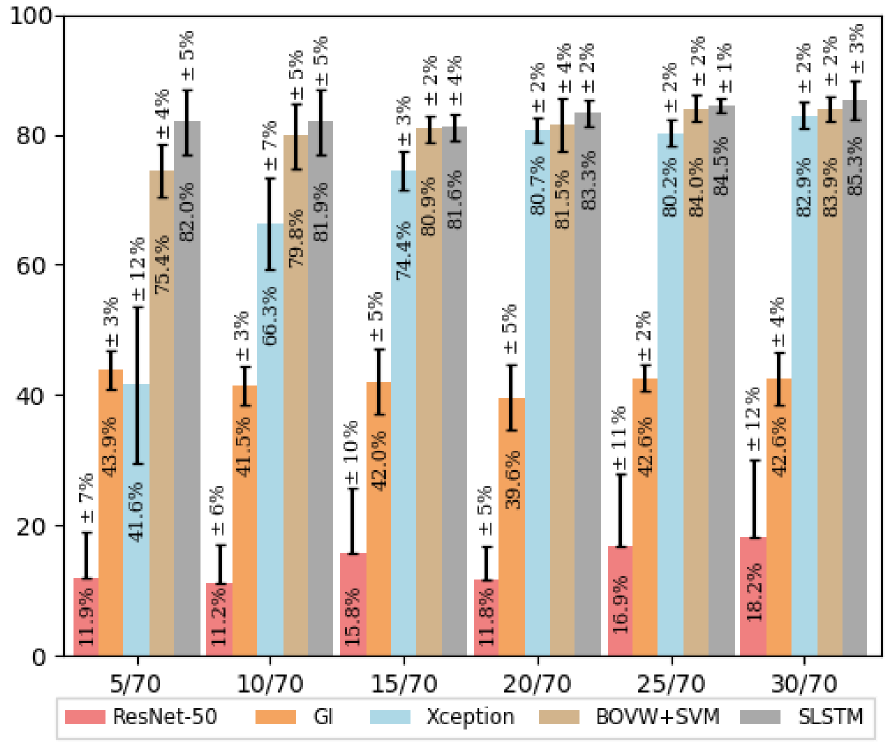

Figure 4.4: Average precision and standard deviation of 10 repetitions on different training sets for ResNet-50, GI, Xception, BOW+SVM, and SLSTM.

Figure 4.4 shows that Xception improves the precision when it is trained with larger image sets. On the other hand, Resnet-50 maintained a constant, but poor precision compared to Xception, BOW+SVM, and SLSTM. We believe that due to its higher number of trainable parameters, around 10.6% more than Xception, the Resnet-50 may respond less favorably to a number of limited data for training. The Kruskal-Wallis test for Xception resulted in a p-value < 0.0001, which indicates a statistically significant difference in its average performance on the different training sets. Although GI presents a poor precision, the Kruskal-Wallis test resulted in a p-value of 0.3591, i.e., we also have no evidence that there is a statistically significant difference in its average performance.

We also performed the Kruskal-Wallis test to compare the models using the accuracy as metric. The test on the data arrangement 5/70 resulted in a p-value < 0.05, indicating a statistically significant difference between the models. The test on the other data arrangements indicated no evidence of a statistically significant difference between SLSTM and BOW+SVM, as well as between Xception and BOW+SVM on data arrangement 20/70. The SLSTM differed statistically from Xception on all the data arrangements.

Table 4.2 shows a comparison of the precision metric achieved by SLSTM with different alphabet sizes and SIFT contrast set at 0.04. We performed the Kruskal-Wallis test to analyze the differences between the results achieved by SLSTM on the different alphabet sizes. The test resulted in a p-value < 0.05 for all alphabet sizes, except when the alphabet size is 44 (p-value of 0.156), which indicates a statistically significant difference in the average performance of the SLSTM between the different alphabet sizes and training sets.

Table 4.2: Comparison of average precision of 10 repetitions for SLSTM.

Image per class		Precision for different alphabet sizes								
train	test	32	36	40	44	48	52	56	60	64
5	70	71.5	73.6	79.4	**82.0**	76.8	73.3	72.1	73.2	71.8
10	70	76.1	77.2	77.5	**81.9**	79.9	78.4	79.1	77.4	78.7
15	70	78.8	78.7	81.0	**81.6**	78.9	79.7	79.3	79.1	80.1
20	70	82.2	82.0	82.4	**83.3**	80.6	82.1	82.5	83.1	82.6
25	70	83.1	82.5	83.2	**84.5**	83.8	82.4	82.8	82.7	83.6
30	70	84.2	83.5	83.9	**85.3**	85.0	83.7	84.5	83.9	83.8

4.5 Conclusion

We present a descriptive model that infers relations between decomposable components of objects to represent their visual patterns. The method connects these components, represented by alphabet symbols, to form strings that represent the visual patterns of the objects. These strings are inserted in an LSTM classifier to find the relation between their symbols and decide which target class this relation belongs to. An extensive experimental evaluation using aerial soybean crop images captured by a UAV was carried out to compare our method with other approaches. The results suggest that representing visual patterns in a compositional way, repeating primitives, can be a viable alternative when there is only a limited number of samples for training. As future work, we plan to explore Gated Recurring Units (GRU) in our model in order to reduce time and the storage of parameters.

CHAPTER 5

A New Approach for Applying Natural Language Processing to Image Classification Problem

Authors[3]: Gilberto Astolfi, Diego André Sant'Ana, Jõao Vitor de Andrade Porto, Fábio Prestes Cesar Rezende, Everton Castelão Tetila, Edson Takashi Matsubara, Hemerson Pistori.

Abstract: A growing interest in applying Natural Language Processing (NLP) models in computer vision problems has recently emerged. This interest is motivated by the success of NLP models in tasks such as translation and text summarization. In this paper, we propose a new method for applying NLP to image classification problems. Our aim is to represent the visual patterns of objects using a sequence of alphabet symbols and then train some form of Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), or Transformer using these sequences to classify objects. An extensive experimental evaluation using a limited number of samples for training has been conducted to compare our method with ResNet-50 deep learning architecture. The results achieved by the proposed method overcome ResNet-50 in all test scenarios. In one test, the method presents an average accuracy of 95.3% against 89.9% of the ResNet-50.

[3]Paper submitted in June 2021 to IEEE Transactions on Pattern Analysis and Machine Intelligence

5.1 Introduction

Convolutional neural networks (CNNs) have been the standard paradigm for image classification tasks in computer vision. One of the key elements for its success is the availability of large images set for training, as ImageNet (Deng et al., 2009). In parallel, Natural Language Processing (NLP) also has made significant progress, in particular Self-attention-based architectures (Vaswani et al., 2017), which is currently the standard paradigm in NPL.

Motivated by the success of NLP models, some works try to combine NLP and CNN models for building hybrid architectures to solve computer vision tasks (Carion et al., 2020; Chen et al., 2020a; Touvron et al., 2021; Dosovitskiy et al., 2021). However, even presenting good results, these hybrid models need large amounts of data for training for having a good generalization (Dosovitskiy et al., 2021). The same is observed in CNN-based models since this type of model has discriminative characteristics (Gu et al., 2018).

In this work, we present a new method that combines object detection models, syntactic pattern representation, and NLP models to classify images when there is only a limited number of samples for training. In the method, we assume that an object can be recognized from the composition of its main parts and that a small set of object parts can be combined to describe the visual pattern of different objects. These parts are regions in an object that attract our attention to identify it and that, together, form its visual pattern. The method receives as input an annotated image with bounding boxes around parts of the object classes. The bounding boxes in the image are associated with alphabet symbols and related using a graph to derive a sequence. The resulting sequence, which encapsulates the structural relationship between object parts, is sent to an encoder (E.g .: some form of Gated Recurrent Units (GRU) (Cho et al., 2014), Long Short-Term Memory (LSTM) (Greff et al., 2017), or Transformer (Vaswani et al., 2017)) to train a sequence classifier. In parallel, we train a model to detect object parts using the same images with bounding boxes. During the tests, we used the object parts detector model to identify the parts in the image and then relate them using a graph to derive a sequence. Finally, the sequence is sent to the sequence classifier to classify it and determine the object class. We experimented as encoder the well-explored NPL models: Gated Recurrent Unit (GRU) (Cho et al., 2014), Long Short-Term Memory (LSTM) (Greff et al., 2017), Bidirectional Long Short-Term Memory (BLSTM) (Graves and Schmidhuber, 2005), BLSTM with attention mechanism (Zhou et al., 2016), and Transformer (Vaswani et al., 2017), and used a Faster R-CNN model (Ren et al., 2017) with ResNet-50 backbone (He et al., 2016) trained from scratch to build a model to detect object parts.

We evaluated the method on the insect classification task. The dataset has a small set of images annotated with bounding boxes around the insect parts (head, body, paw, etc.) and classified based on the insect species (Nezara, Edessa meditabunda, etc.). We compared the method with ResNet-50 (He et al., 2016). We aim to investigate whether our method and ResNet-50 have good classification performance when trained on small datasets. The results achieved by the proposed method overcome ResNet-50 in all test scenarios. In one test, the method presents an average accuracy of 95.3% against 89.9% of the ResNet-50.

The contributions of this paper are:

- We propose a new method that combines object detection models, syntactic pattern representation, and NLP models to classify objects from the composition of its main parts.

- We show a new way to train classification models, in which objects are decomposed into main parts sharing common parts with each other. This allows us to extrapolate object appearance (partial occlusion, different poses, etc.) beyond the samples that were offered for training.

The rest of the paper is organized as follows. In Section 5.2, we show some works that apply natural language processing concepts to computer vision problems. In Section 5.3, we introduce our method. In Section 5.4 present the experimental setup, followed by the results and analysis in Section 5.5. Finally, in Section 5.6 we present the conclusions.

5.2 Related Work

This is not the first attempt to investigate the possibility of applying natural language processing to computer vision problems. Pistori et al. (2013) proposed a method to represent an image as a string. The method infers a grammar for an object class from the set of strings derived from images of this class. An image is associated with a class when the grammar for this class produces fewer errors by parsing the string derived of it. Astolfi et al. (2020) extended the model proposed by Pistori et al. (2013) by exchanging the grammatical inference by an LSTM classifier. The proposed method maintains stable results even when trained with few samples. Abid et al. (2018) also adopted an approach based on sequences to represent visual patterns. In this work, the authors split postal address images into parts, as words and characters, and then tokenize them to be embedded and inserted into a Bidirectional LSTM classifier. Álvaro et al. (2016) used a Bidirectional LSTM to classify sequences derived from images that represent handwritten mathematical expressions. For building the sequences, the authors used a tree to connect the mathematical expression strokes.

More recently, inspired by Self-attention-based architectures success in NLP, some convolution-free models try to apply a standard Transformer (Vaswani et al., 2017) directly to images to computer vision problems, positioning it as a possible alternative to the convolutional neural network. Dosovitskiy et al. (2021) proposed the Vision Transformer, a simple model that split an image into 16×16 patches for providing a sequence and inputting it into a standard transformer. Touvron et al. (2021) improved the proposed model by Dosovitskiy et al. (2021) by adding Knowledge Distillation (Hinton et al., 2015) and class tokens along with the Vision Transformer patches. Chen et al. (2020b) proposed a model trained in an unsupervised way that pre-processes images to reduce resolution and color space for reshaping it into a 1-dimensional pixel sequence for then applies standard Transformer to pixel sequence. Wu et al. (2020) applied convolutional layers to obtain image features and build tokens. However, before sending the tokens to standard Transformer, they are clustered into small visual tokens that express semantic concepts in the image and then are related for embedding spatial relationship between them.

Our work is different from all mentioned since we propose a method that relates component parts of a given object in an image using a graph from where we derive a sequence. The symbols of the sequence, which encapsulate structural relations between component parts of the object, are treated the same way as tokens in NLP problems.

5.3 Proposed Method Overview

Our idea here is that an image can be summarized by a sequence of alphabet symbols and can be interpreted as a natural language processing problem. To instantiate this idea, we introduce a method that derives a sequence of symbols from an image for classifying it as if it was some sentence of a language. The symbols of the sequence are handled the same way as tokens (words) in NLP problems.

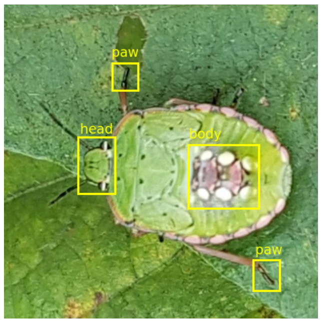

Figure 5.1: Example of component parts of an object class. The head, body and paws are component parts of the object class insect.

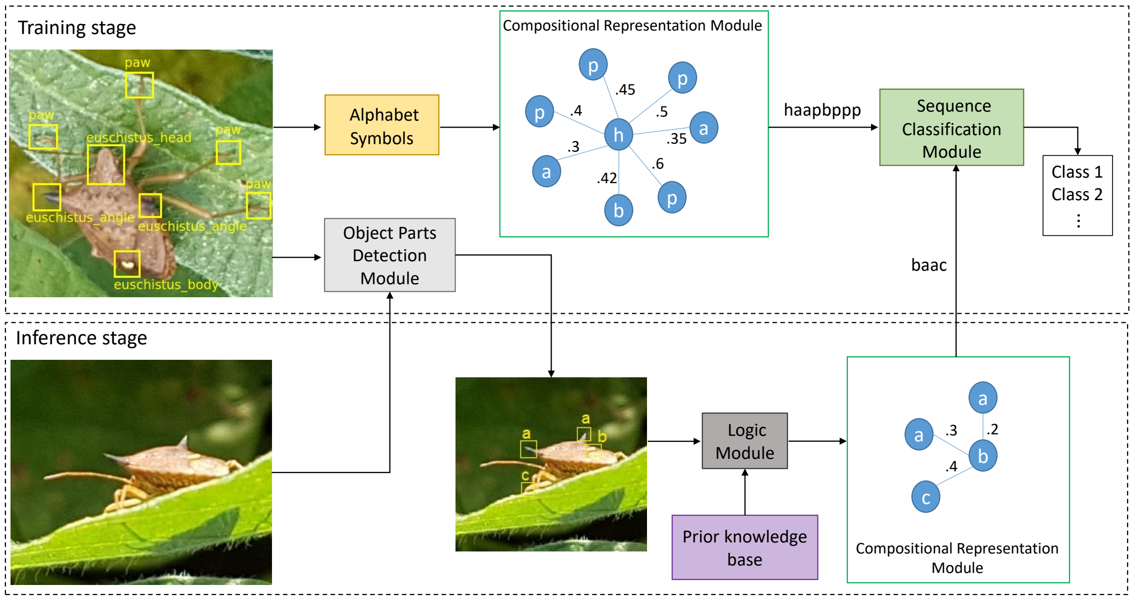

Figure 5.2: Method overview: In the training stage, the method receives as input an annotated image with bounding boxes. The bounding boxes in the image are associated with alphabet symbols and related using a graph in Compositional Representation Module to derive a sequence. The resulting sequence is sent to the Sequence Classification Module for training. In parallel, the Object Parts Detection Module is trained using the same annotated image with bounding boxes. In the inference stage, the Object Parts Detection Module detects the component parts of the object and sends them to the Logic Model that transfers the filtred component parts for Compositional Representation Module for deriving a sequence. The resulting sequence is sent to the Sequence Classification Module to determine the image class.

Before introducing the proposed method, we need to define the component part of an object class. In this paper, component part is used to refer to the parts of a given object we look at to identify it, i.e., object parts that attract attention. In Figure 5.1 examples of component parts of an object class are shown.

The proposed method has two main stages: training and inference, both employing a hybrid architecture. Figure 5.2 shows an overview of the proposed method.

In the training stage, the method receives as input an annotated image with bounding boxes around component parts of a given object class. Each component part is associated with an alphabet symbol and the central coordinate of its bounding box (see section 5.3.1). The annotated image is inputted into the Compositional Representation Module to derive a sequence from it. To that end, a graph is built based on a distance measurement between the central coordinates of each bounding box. Nodes in the resulting graph represent component parts of the object class, whose labels are alphabet symbols. After, the graph is traversed using a predefined order, and a sequence is derived by concatenating the labels of each node (see section 5.3.2). Finally, the sequence is inputted in a Sequence Classification Module for training (see section 5.3.4). In parallel, an Object Parts Detection Module is trained using the same annotated image with bounding boxes (see section 5.3.3).

In the inference stage, the method receives as input an image for classification. The image is sent for the Object Parts Detection Module to detect the component parts (section 5.3.3). These component parts are inputted into the Logic Module to perform filtering based on a provided prior knowledge base (see section 5.3.5). The selected component parts are connected into a graph and a sequence is derived as in training (section 5.3.2). Finally, the Sequence Classification Module classifies the sequence and associates it with an object class (section 5.3.4).

In the following, we describe the modules of the proposed method in detail.

5.3.1 Defining Alphabet Symbols

The alphabet symbols definition consists of making direct association between a given symbol (E.g.: ASCII character) with a component part of an object class. However, its definition depends on the number of component parts of each class in the dataset. An object class can have 2 to n component parts. Besides, two object classes can share common component parts between them. For example, the insects in Figure 5.3 are two different object classes with distinct component parts. However, they have the head and the paw as component parts in common. This is possible because their heads and paws have a similar visual pattern since they are of the same species at different life stages.

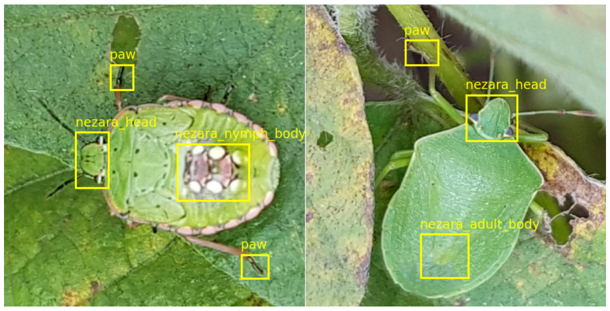

Figure 5.3: Two different object classes with component parts in common. Object classes can share component parts when these parts have a similar visual pattern.

Based on the two object classes used as an example in Figure 5.3, we could make the following direct association between alphabet symbols with the component parts of the object classes: $nezara\_adult\_body \rightarrow A$, $nezara\_nymph\_body \rightarrow B$, $nezara\_head \rightarrow C$, $paw \rightarrow D$. The Resulting alphabet could be $\Sigma = \{A, B, C, D\}$.

5.3.2 Compositional Representation Module

The Compositional Representation Module receives as input an image with bounding boxes around component parts of an object. The first step consists of associating an alphabet symbol with each bounding box. Next, the bounding boxes are processed to build a graph and derive a sequence for representing the structural relations among the component parts of the object.

We define the Visual Weighted Graph (VWG) to encapsulate structural relations among the component parts of an object. A VWG is a two-element tuple $G = (V(G), E(G))$, where $V(G)$ is a two-element tuple $V(G) = (\lambda, P)$ that represent the component parts of the object. λ is the label of the node represented by an alphabet symbol and P is the bounding box center coordinates x and y. $E(G)$ is the edges set, where each edge $e_i \in E(G)$ is defined by a three-element tuple $e_i = (v_j, v_k, w)$, such that $v_j, v_k \in V(G)$ and $w \in \mathbb{R}$ denotes the weight of e_i.

The first step to build VWG is to transform each bounding box into a node in the graph. This task consists in obtaining the central coordinates of each bounding box and associating them with an alphabet symbol. Next, we identify which coordinate is closest to the center of the image to start the graph build from it. This central coordinate becomes the graph center node v_1, connected to all other nodes. The edge weight between the central node v_1 and a given node v_i is calculated based on the spatial distance between them. The distance measurement $w_{1i} \in \mathbb{R}$ between central v_1 and v_i is carried out as follow:

$$w_{1j} = \left\| v_1^P - v_i^P \right\|_2 \tag{5.1}$$

The resulting VWG is a complete weighted bipartite graph $K_{1,k}$, i.e., a tree with one internal node and k leaves. Based on the bounding boxes of the two object classes presented in Figure 5.3, we could build the VWGs shown in Figure 5.4.

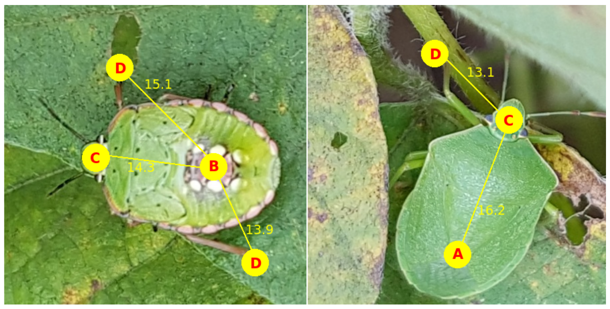

Figure 5.4: Generated VWGs from bounding boxes.

Finally, the VWG is traversed from the central node, visiting the nodes with the lower values first. The aim is to visit all VWG nodes and concatenate λ of each node to build a sequence. For example, the graphs in Figure 5.3 generate the sequences *BDCD* and *CDA*. To traverse the VWG from the central node guarantee that alphabet symbols keep their position in the sequence, even when there is a rotation on the image.

5.3.3 Object Parts Detection Module

The Object Parts Detection Module is a Faster R-CNN model (Ren et al., 2017) with ResNet-50 backbone (He et al., 2016) trained from scratch. The Faster R-CNN model receives as input annotated images with bounding boxes around component parts of the object classes to build an object parts detection model.

We choose the Faster R-CNN because it has good accuracy when dealing with small objects (Nguyen et al., 2020) , and component parts of the object classes have this characteristic. In this work, we adopt the definition of Zhu et al. (2016) for small objects. Zhu et al. (2016) define that small objects are objects whose sizes fill up to 20% of an image.

5.3.4 Sequence Classification Module

Typically a standard Natural Language Processing (NLP) model maps a fixed-length input to a fixed-length output, in which the length of both input and output may differ. In this model type, commonly known as Sequence to Sequence (Sutskever et al., 2014), the encoder captures the context of the input sequence as a hidden state vector and transfers it to the decoder, which then provides the output sequence. Generally, both the encoder and decoder use LSTM (Greff et al., 2017), GRU (Cho et al., 2014), or Transformers (Vaswani et al., 2017) to process sequences.

Our Sequence Classification Module uses only the encoder of a Sequence to Sequence model. It captures structural relations among the component parts of an object class encapsulated in a sequence and transfers it to an encoder. Then, it takes the produced output by an encoder and uses a feedforward network on top to classify the sequence, i.e., the object class.

First, we truncate and pad the input sequences so that they are all in the same length. Next, we convert alphabet symbols to numbers and transfers them to an embedded layer that uses fixed length vectors to represent each word. The embedded layer sends the output to an encoder layer (E.g.: some form of GRU, LSTM, or Transformer) for training. Finally, we use a Dense output layer with a number of neurons equal to the number of object classes, and a softmax activation function to make object class classification. In Figure 5.5, the Sequence Classification Module is shown.

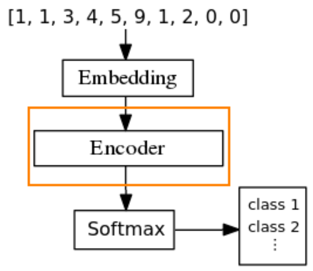

Figure 5.5: A Sequence Classification Module with a generic encoder, whose input is a sequence, and the output is an object class.

5.3.5 Logic Module

The Logic Module receives the output of the Object Parts Detection Model as input, i.e., a set of component parts of object classes detected (bounding boxes) in a given image. Then, a search is executed in a prior knowledge base to discard surplus component parts.

The prior knowledge base provides information about the composition of each object class in the dataset, maintaining the number of component parts of each one. For instance, if the dataset has only the two object classes shown in Figure 5.3, the prior knowledge base would be formed by the following dictionary containing keys and values: $\{nezara\_adult\_body : 1, nezara\_nymph\_body : 1, nezara\_head : 1, paw : 8\}$; where the key represents the component part and the value the number of occurrences permitted for the component part in an object class. In this example, the *paw* can occur only eight times in an object class, as it is a component part shared between the two insects, and each of them has a maximum of eight paws. The *nezara_head* can occur once because it also is a component part shared between the two insects. Finally, *nezara_adult_body* and *nezara_nymph_body* can occur once in both object classes. Thus, an image inputted for classification can have a maximum of eight *paws*, one *nezara_head*, one *nezara_adult_body*, and one *nezara_nymph_body*. The component parts selected are those with the highest confidence score.

The Logic Module output is a set of component parts of object classes detected (bounding boxes) and filtered by a prior knowledge base. This output is sent to the Compositional Representation Module.

5.4 Experimental Setup

5.4.1 Building the Dataset

Our dataset consists of 400 images 600x600 size from four insect classes, 100 images per class. The images are annotated with bounding boxes around the component parts of the insects, as well as classified based on the insect species.

To build the dataset, we first collected images of four insect classes from the iNat2017 dataset (Horn et al., 2018) and then completed with images from our dataset (Tetila et al., 2020) to obtain a set containing 100 images per class. The selected images were then manually cropped and resized to 600x600. Once the images were captured in real-world conditions, the dataset has insect images with partial occlusion and different poses. See Figure 5.6 for image examples per class.

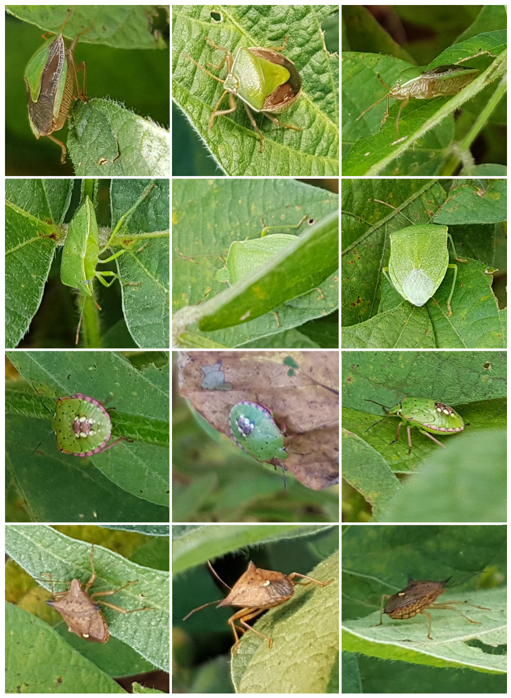

Figure 5.6: Sample images for each of the four insect classes. Row 1: Edessa meditabunda; row 2: Nezara adult; row 3: Nezara nymph; row 4: Euschistus heros.

After constructing the dataset in a balanced way, we use bounding boxes to annotate component parts for each of the four insect classes. Thus, the full dataset has four classes based on the insect names, whose images are annotated with bounding boxes around the component parts of each of the insects. See Figure 5.7 for examples for each of the four insect classes with bounding boxes around its component parts.

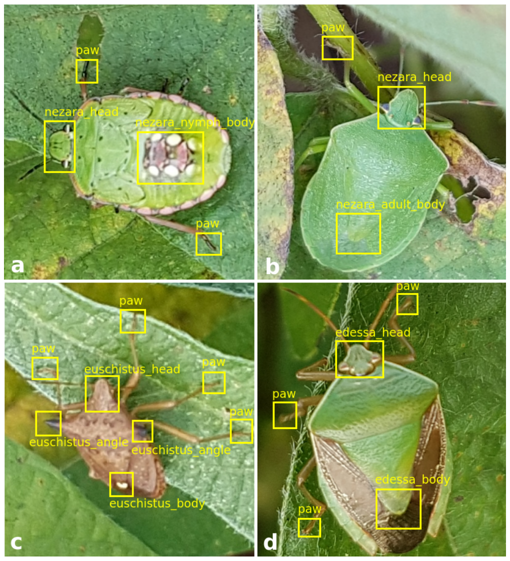

Figure 5.7: Sample images for each of the four insect classes with bounding boxes around its component parts. a) Component parts Nezara nymph: nezara_nymph_body, paw, and nezara_head; b) Component parts Nezara adult: nezara_adult_body, paw, and nezara_head; c) Component parts Euschistus heros: euschistus_body, euschistus_head, euschistus_angle, and paw; d) Component parts Edessa meditabunda: edessa_body, edessa_head, and paw. Component parts were annotated using LabelImg.

In Table 5.1, we present the component parts for each of the 4 insect classes. Note that, all insect classes share the component part *paw*, and Nezara nymph and Nezara adult classes share the component part *nezara_head*.

Table 5.1: Component parts for each of the four insect classes.

	Nezara nymph	Nezara adult	Euschistus heros	Edessa meditabunda
nezara_nymph_body	x			
nezara_adult_body		x		
nezara_head	x	x		
euschistus_body			x	
euschistus_head			x	
euschistus_angle			x	
edessa_body				x
edessa_head				x
paw	x	x	x	x

5.4.2 Sequence Classification Module Configuration

We conduct experiments on the insect dataset using five different encoder layers: GRU, LSTM, Bidirectional LSTM, Bidirectional LSTM with attention mechanism and Transformer.

In Figure 5.8, the architectures used in the experiment are shown. All layer stacking between the embedding layer and softmax layer are called encoder layer. During the experiment, we trade one encoder layer for another in order to compare them in the classification task. We set as default 100 units to LSTM layers, a rate of 0.2 in Dropout layers to prevent overfitting, and used the Adam optimizer (Kingma and Ba, 2015) with a learning rate set at 0.001 and a weight decay of 0.0001. We also used early-stopping to prevent overfitting with patience set at 5. We use 500 epochs for training.

Before all encoder layers, we first embedded the input sequences into a 32, 64, or 128-dimensional vector, then send them through the encoder. The vector dimension for embedding the input sequences is treated in this work as a hyperparameter. We want to analyze the effects of the variation of the input length in the embedding layer on encoder performance.

5.4.3 Object Parts Detection Module Configuration

In our Object Parts Detection module, we train the Faster R-CNN for the ResNet-50 as a backbone from scratch on the public MMDetection platform (Chen et al., 2019) with the provided training setup, except that we used Adam optimizer with a learning rate set at 0.001 and a weight decay of 0.0001. Besides, we used 500 epochs in training.

5.4.4 Counting Component Parts for Comparison

We carry out an insect classification based only on the component parts detection provided by Object Parts Detection Module. The classification consists of taking the output of the detector for a given image and counting the detected parts for each insect class. The insect class that produces the highest numbers of component parts detected is associated to the image. We carry out this counting to investigate whether the results achieved by our method are not only mere component parts count provided by the Object Parts Detection Module. This classifier is called "Count".

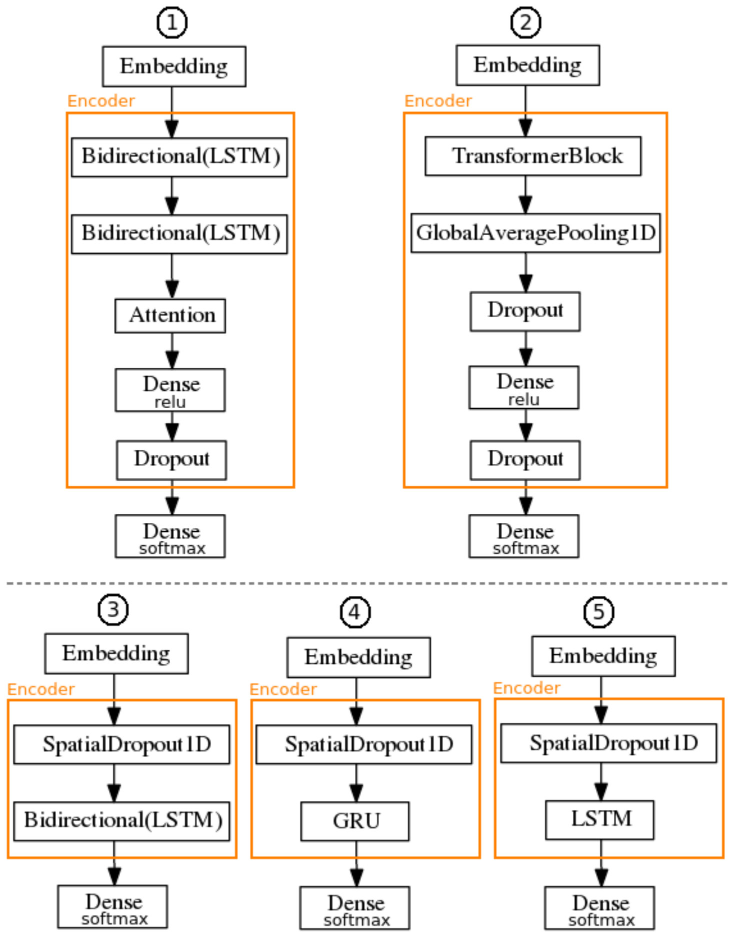

Figure 5.8: The five different encoder layers used in the experiment: 1) Bidirectional LSTM with attention mechanism; 2) A typical Transformer block; 3) Bidirectional LSTM; 4) GRU, and; 5) LSTM.

5.4.5 *Configuring Resnet-50 for Comparison*

We compare our method with the state-of-the-art Resnet-50. We choose the Resnet-50 because we use it as the backbone in our Object Parts Detection module.

We adopt three distinct training strategies to Resnet-50:

- A base model pre-trained with weights from ImageNet. In this model, we froze all layers in the base model and created new layers on top of it. The layers on top consist of one Dense layer with 1024 units, followed by a Dropout layer and two other Dense layers with 1024 and four (number of object class of our problem) units, respectively. We call this model ResNet-50 Transfer learning.

- A fine-tuning technique that uses the ResNet-50 Transfer learning model, which we call ResNet-50 Fine-tuning. Here, we freeze the layers from the previously trained model for training only the layers on top. Then, we unfroze the model and retrained it using a lower learning rate (0.0001).

- The ResNet-50 Transfer learning model trained from scratch with the weights randomly initialized. We call this strategy of ResNet-50 from scratch.

In all strategies, we set to 0.2 the dropout rate. We use the Adam optimizer with a learning rate set at 0.001 and a weight decay of 0.0001. Besides, we use early-stopping with patience set to 5 and train with 500 epochs.

5.4.6 Data Sampling

We conducted an experiment where used only a limited number of examples for the training. The aim was to verify if our method learns to classify insect images from only a few samples of each class. To this end, we carried out the experiment by gradually reducing the number of training and validation images, always maintaining the same number of testing images. In the first test, we randomly split the dataset into 70% for testing and 30% for training. In the second, we split it into 70% for testing and 20% for training. Finally, we split the dataset into 70% for testing and 10% for training. In all tests, we separated 10% of the training set for validation. For each of the tests, we carried out ten repetitions.

We applied an oversampling technique to equal the number of component parts (bounding boxes) in the training and validation sets in all repetitions. The oversampling technique consists of randomly rotating the image and its bounding boxes between $0°$ to $90°$.

The repetitions, as well as the oversampling, were generated at the beginning of the experiment to use the same images in the training, validation, and tests in our method and in the Resnet-50.

5.4.7 Metrics and Result Analysis

We produced averages for accuracy to evaluate our method and the training strategies adopted for Resnet-50. Besides, we used an analysis of variance to compare the performance of each encoder layer by varying the input length in the embedding layer and compare our method with Resnet-50.

In the analysis of variance, we adopted one-way ANOVA on a 0.05 level of significance using the accuracy as metric. We adopted the one-way ANOVA test because, in general, the accuracies present normality/homogeneity of variance.

5.5 Results and Discussion

We first show the result of each Encoder varying the input length in the embedding layer. In Table 5.2, accuracy rates for Encoders with input length 32, 64, and 128, respectively, are presented. When the input length in the embedding layer is 32, the Transformer encoder had the highest average accuracy on data sampling arrangement with 10% and 20% of images for training, accuracies of 85.9% and 92.0%, respectively. On data sampling arrangement with 30% of images for training, the Bidirectional LSTM encoder achieved the highest average accuracy, 94.7%. When the input length is 64 and 128, Transformer and Bidirectional LSTM encoders continued to present the highest average accuracies. However, the Transformer encoder had the highest average accuracy only on data sampling arrangement with 10% of images for training.

Table 5.2: Accuracy averages of 10 repetitions for input length of 32, 64, and 128 in the embedding layer on the different data sampling arrangements. The highest average accuracies in each data sampling arrangement are highlighted.

images per class		GRU	LSTM	BLSTM	BLSTM Attention	Transfor-mer
train	test					
32-dimensional input						
10%	70%	78.8 ±5.09	79.6 ±4.45	80.8 ±4.38	79.1 ±3.89	**85.9** ±5.19
20%	70%	90.5 ±1.46	90.5 ±0.83	91.9 ±1.00	90.8 ±1.14	**92.0** ±0.96
30%	70%	93.5 ±1.31	93.7 ±1.42	**94.7** ±1.60	93.6 ±1.71	94.4 ±1.62
64-dimensional input						
10%	70%	79.7 ±3.81	80.8 ±4.47	82.0 ±4.65	79.9 ±4.33	**84.7** ±4.71
20%	70%	91.0 ±1.76	90.7 ±1.04	**91.8** ±1.36	90.7 ±1.40	91.5 ±1.39
30%	70%	93.5 ±1.74	93.9 ±1.70	**94.7** ±1.60	94.2 ±1.47	94.6 ±1.48
128-dimensional input						
10%	70%	79.8 ±4.91	80.2 ±4.78	82.9 ±4.92	80.7 ±4.73	**83.2** ±4.62
20%	70%	91.2 ±1.26	91.2 ±1.07	**92.6** ±1.22	90.9 ±1.23	91.1 ±1.07
30%	70%	94.0 ±1.31	94.2 ±1.56	**95.3**±1.49	94.4 ±1.47	94.0 ±1.23

We applied analysis of variance to determine whether it makes a difference to use 32, 64, or 128 input lengths in the embedding layer of the Encoders when are trained with one of the data sampling arrangements (10/70, 20/70, or 30/70). For example, this analysis can determine whether it makes some difference to use 32, 64, or 128 input lengths in the Transformer encoder on data sampling arrangement with 10% of images for training. The test resulted in a p-value

Table 5.3: Highest average accuracies according to the average of 10 repetitions for GRU, LSTM, BLSTM, BLSTM Attention, Transformer, ResNet-50 with Transfer learning, ResNet-50 with Fine-tuning, ResNet-50 trained from scratch, and Count. The highest average accuracies in each data sampling arrangement are highlighted.

images per class		GRU	LSTM	BLSTM	BLSTM Attention	Transformer	ResNet-50 Transfer learning	ResNet-50 Fine-tuning	ResNet-50 from scratch	Count
train	test									
10%	70%	79.8 ±4.91	80.8 ±4.47	82.9 ±4.92	80.8 ±4.47	**85.9** ±4.62	78.1 ±6.03	81.0 ±5.21	30.7 ±6.42	62.5 ±5.84
20%	70%	91.2 ±1.26	91.2±1.07	**92.6** ±1.22	90.9 ±1.23	92.0 ±0.96	87.6 ±2.50	86.8 ±4.13	29.2 ±10.53	79.6 ±2.44
30%	70%	94.0 ±1.31	94.2 ±1.56	**95.3** ±1.49	94.4 ±1.47	94.6 ±1.48	89.9 ±2.83	88.8 ±2.06	36.0 ±16.17	85.3 ±2.89

> 0.05 for all Encoders, indicating no statistically significant difference in the average accuracy when using different input lengths in the embedding layer on a given data sampling arrangement.

Since the average accuracy of the Encoders did not present a statistically significant difference when the input lengths in the embedding layers is varied, we picked up the highest average accuracies of each Encoder on the different data sampling arrangements and showed them in Table 5.3 in order to perform some comparison with the ResNet-50 and Count. The Table 5.3 shows that the Transformer encoder had the highest average accuracy on data sampling arrangement with 10% of images for training, an accuracy of 85.9%. On data sampling arrangement with 20% and 30% of images for training, the Bidirectional LSTM encoder achieved the highest average accuracies, 92.6% and 95.3%, respectively.

We also performed the analysis of variance to compare the Encoders, the ResNet-50, and the Count (classification based on the component parts detection provided by Object Parts Detection Module, described in the Section 5.4.4) based on its highest average accuracies presented in Table 5.3. On data sampling arrangement with 10% of images for training, the analysis of variance showed a statistically significant difference between the Transformer encoder and the ResNet-50 strategies (transfer learning and fine-tuning), whose test resulted in a p-value < 0.05. Besides, the analysis of variance determined a statistically significant difference between Encoders and Count, and between the ResNet-50 strategies and Count. This analysis showed evidence that the Transformer presented better performance when trained with 10% of the images from the dataset, as it had the highest average accuracy (85.9%) between Encoders and differed statistically from the ResNet-50 strategies and Count. Using 20% and 30% of the images for training, the Encoders didn't differ statistically from each other, but they differed from the ResNet-50 strategies and Count, whose resulting p-value < 0.05. Besides, in both data sampling arrangements, the ResNet-50 using the transfer learning and fine-tuning strategies did not show a statistically significant difference from each other, but they differed from Count.

In Figure 5.9, we show three confusion matrices. They represent the Encoders that achieved the best results in each data sampling arrangement shown in Table 3: Transformer, BLSTM, and

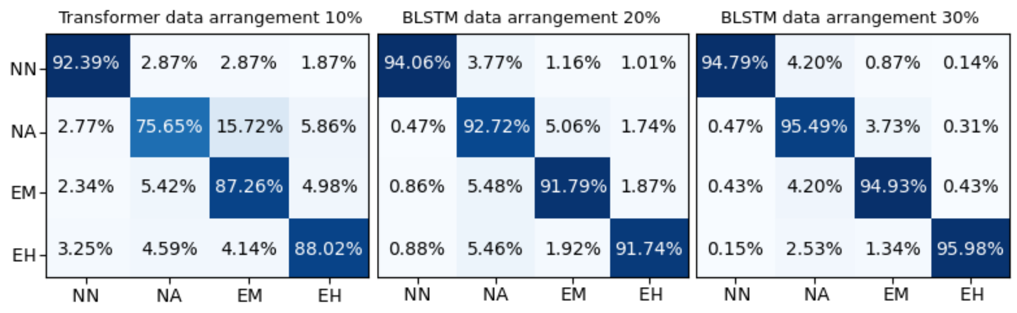

Figure 5.9: Confusion matrices for the Encoders that achieved the best results in each data sampling arrangement shown in Table 3. Left to right: Transformer, BLSTM, and BLSTM, data sampling arrangement with 10%, 20%, and 30% of images for training, respectively. Object classes: Nezara nymph (NN), Nezara adult (NA), Edessa meditabunda (EM), and Euschistus heros (EH).

BLSTM, data sampling arrangement with 10%, 20%, and 30% of images for training, respectively. Note that Nezara adult (NA) and Nezara nymph (NN) classes have low false-positive rates between them. This observation is important, as the classes share two common component parts with each other (head and paw). The low false-positive rate indicates that the Encoders learned the relationships between the component parts of both classes, and most importantly, they learned to differentiate them, even though they have only one distinct part. This is evidence that the visual pattern representation strategy adopted in this work, in which objects share common component parts with each other, can give a higher capacity for generalization, besides allowing an unlimited number of visual variations for a given object from few variations in their sequence of symbols. This can allow the visual variation of the object to be extrapolated beyond the samples in the training set, giving conditions to deal with pose variation, deformation, and different perspectives of objects.

Although ResNet-50 has gained immense popularity in classification problems on large datasets, our experiment showed evidence that it may not generalize well from a few samples for training, same when is used transfer learning and fine-tuning to deal with small datasets (Oquab et al., 2014). On the other hand, even though evaluated with few object classes, our method showed significantly better results on the classification addressed in our experiment.

The average accuracies presented by our method substantially outperform the average accuracies presented by Count (see Table 5.3). That shows that the results achieved by our method are not only mere component parts count provided by Faster-RCNN used in Object Parts Detection Module (section 5.3.3).

5.6 Conclusion

In this work, we present a method for applying natural language processing to image classification problems. The method infers spatial relations between component parts of objects to represent their visual patterns. The method connects these component parts, represented by alphabet symbols, to build sequences. The resulting sequence, which encapsulates the spatial relationship between object parts, is sent to an encoder (E.g .: some form of GRU, LSTM, or Transformer) to train a sequence classifier. In parallel, a model to detect object parts is trained. During the inference, the object parts detector model is used to identify the parts in the image and then relate them to derive a sequence. Finally, the sequence is sent to the sequence classifier to classify it and determine the object class. In one test, the method achieved an average accuracy of 95.3% overcoming the ResNet-50 that achieved 89.9%.

Our method presented better results compared with Resnet-50 because it learns the spatial relationships between high-level component parts of the objects contained in the sequences. Even though there are a limited number of samples for training, in most cases, the variations in the visual pattern of the object are represented by the sequences, since the objects have few combinations of high-level component parts that co-occur. On the other hand, the learning paradigm of the Resnet-50 is based on feature learning which lacks precise spatial relationships between high-level parts. That requires an enormous amount of samples to represent all possible visual patterns of the objects.

Chapter
6

Conclusions

6.1 Final Considerations

This work proposed an approach to represent visual patterns of objects in a syntactic way for applying natural language processing models to image classification problems. In this regard, a systematic literature review about syntactic pattern recognition in images was carried out to find gaps of study for being explored by this work. The survey results showed that the natural language processing models had not yet been applied to image classification problems in a natural way, i.e., in the same way as they are applied to natural language problems, receiving the visual patterns from the image in the form of text. Then, two visual pattern representation approaches of objects in a syntactic way were investigated: the representation using keypoints and the representation using component parts of objects.

In Chapter 4, the visual pattern composition of objects using keypoints of images is explored in order to represent visual patterns in a compositional way. The central idea is to identify key points in images, treated here as primitives, associate them with alphabet symbols, and then relate them to derive strings from images. Strings are the inputs for training an LSTM encoder. The experiment showed evidence that the syntactic pattern representation can represent visual variations in superpixel images captured by Unmanned Aerial Vehicles, even when there is a small set of images for training. The proposed method showed an accuracy of 80.5% and 84.6% when trained with 5% and 30% of the images from the dataset, respectively. Moreover, the proposed method differed statistically from the Xception and Resnet-50 architectures on all the data sampling arrangements.

Component parts of objects are explored in Chapter 5 as primitives to compose visual patterns of objects. The component parts of the object are provided by means of bounding boxes in the images. They are associated with alphabet symbols and related with each other to derive a sequence of symbols from the object for representing its visual pattern. Then, some form of GRU, LSTM, or Transformer are trained to learn the spatial relationships between component parts of the objects contained in the sequences. During the inference, the models are assisted by a domain knowledge base to classify the images within object classes. The experiment showed evidence that the visual variations of objects, such as poses and scales variation, occlusions, among others, can be represented by the sequence of symbols even when there are a limited number of samples for training since the objects have few combinations of high-level component parts that co-occur. In one test, the proposed method presented an average accuracy of 95.3% against 89.9% of the ResNet-50. The limitation of the method is to represent visual patterns of objects in an autonomous way.

Both forms of visual pattern composition of objects, using keypoints and component parts of objects as primitives, showed evidence that from a finite set of primitive structures is possible to obtain many variations in the visual pattern of the object when there are few samples for training. Besides, the syntactic visual pattern representation allowed NLP models to be applied to image classification problems in the same way as they are applied to sentences, i.e., receiving a sequence of symbols as input.

6.2 Future Work

In this work, a supervised training approach was employed, using annotations with bounding boxes, to obtain samples of component parts of high-level objects (e.g.: head, body, and paws of insects). However, as the number of object classes increases, the task of annotating component parts of objects on the image with labels becomes more costly. A suggestion for future research includes the use of weakly supervised learning, focusing on extracting salient maps (Mundhenk et al., 2019) from images to be used as component parts of objects.

Another issue that can be better explored is related to the domain knowledge provided to the model to assist it in the classification task. Concepts such as object parts and relations can be combined by means of compositional logic structures to provide some prior knowledge to the model to directly guide it in the learning. As the object parts and relations are chosen by humans, this strategy becomes the model more interpretable.

Finally, another suggestion for future research includes directly applying NLP models on graphs used to relate component parts of objects. When defining that the input data will be represented as a graph, NLP models can learn neighborhood relations between object parts beyond distance information between them.

6.3 Conclusion

In this work a method for applying natural language processing to image classification problems was presented. Experiments showed evidence that natural language processing models generalize from a few samples in image classification problems when trained with visual patterns of objects represented syntactically. The two classification problems addressed are different, and both showed evidence of good generalization of the NLP models. In the first, an aerial image classification problem, all information in the image is relevant to determine the target class of a given image. In the second, an object classification problem, the image background is an irrelevant factor to determine the object class, but it creates many difficulties for identifying objects. Besides, it has been shown that NLP models can be applied to image classification problems in the same way as they are used to natural language, i.e., treating an image as a sentence. It is hoped that the results achieved by this work can motivate new research to explore new ways to apply NLP models in computer vision problems.

Bibliography

Abid, N., ul Hasan, A., and Shafait, F. (2018). Deepparse: A trainable postal address parser. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Canberra, Australia, Australia. IEEE. 46, 57, 72

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2010). Slic superpixels. Technical Report EPFL Technical Report 149300, School of Computer and Communication Sciences and École Polytechnique Fédrale de Lausanne Joint Repor. 10, 64

Álvaro, F., Sánchez, J.-A., and Benedí, J.-M. (2014). Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters*, 35:58 – 67. Frontiers in Handwriting Processing. 34, 49, 55

Álvaro, F., Sánchez, J.-A., and Benedí, J.-M. (2016). An integrated grammar-based approach for mathematical expression recognition. *Pattern Recognition*, 51:135 – 147. 34, 49, 55, 72

Amorim, W. P., Tetila, E. C., Pistori, H., and Papa, J. P. (2019). Semi-supervised learning with convolutional neural networks for uav images automatic recognition. *Computers and Electronics in Agriculture*, 164:104932. 60

Andreopoulos, A. and Tsotsos, J. K. (2013). 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8):827 – 891. 1, 19

Astolfi, G., Pache, M. C. B., Menezes, G. V., da Silva Oliveira Junior, A., Menezes, G. K., de Weber, V. A. M., Tetila, E. C., de Souza Belete, N. A., Matsubara, E. T., and Pistori, H. (2020). Combining syntactic methods with LSTM to classify soybean aerial images. *IEEE Geoscience and Remote Sensing Letters*, 1(1):1–5. 46, 58, 72

Ayeb, K. K., Echi, A. K., and Belaïd, A. (2015). A syntax directed system for the recognition of printed arabic mathematical formulas. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 186–190, Tunis, Tunisia. IEEE. 36, 55

Bah, M. D., Hafiane, A., and Canals, R. (2018). Deep learning with unsupervised data labeling for weed detection in line crops in uav images. *Remote Sensing*, 10. 60

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. J. V. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359. 46

Bengio, Y., Ducharme, R., and Vincent, P. (2001). A neural probabilistic language model. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press. 1

Bienenstock, E., Geman, S., and Potter, D. (1997). Compositionality, mdl priors, and object recognition. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 838–844. MIT Press. 2

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 7, 8

Blake, A., Kohli, P., and Rother, C. (2011). *Markov Random Fields for Vision and Image Processing*. The MIT Press, Cambridge, Massachusetts, EUA. 44

Boulch, A., Houllier, S., Marlet, R., and Tournaire, O. (2013). Semantizing complex 3d scenes using constrained attribute grammars. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, pages 33–42, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 37, 49, 55

Bourdev, L., Maji, S., Brox, T., and Malik, J. (2010). Detecting people using mutually consistent poselet activations. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 168–181, Berlin, Heidelberg. Springer-Verlag. 27

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, Boca Raton, Florida. 38

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *Computer Vision – ECCV 2020*, pages 213–229. Springer International Publishing. 71

Chanda, G. and Dellaert, F. (2004). Grammatical methods in computer vision: An overview. Technical Report GIT-GVU-04-29, Georgia Institute of Technology. 19, 20, 60

Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., and Gao, W. (2020a). Pre-trained image processing transformer. *CoRR*, abs/2012.00364. 71

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Change Loy, C., and Lin, D. (2019). MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv e-prints*, page arXiv:1906.07155. 81

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. (2020b). Generative pretraining from pixels. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR. 2, 4, 73

Cho, K., van Merrienboer, B., Gülçehre, c., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *EMNLP*, pages 1724–1734. ACL. 15, 71, 77

Choe, T. E., Deng, H., Guo, F., Lee, M. W., and Haering, N. (2013). Semantic video-to-video search using sub-graph grouping and matching. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 787–794, Sydney, NSW, Australia. IEEE. 38, 55

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807. 61

Chua, J. and Felzenszwalb, P. F. (2016). Scene grammars, factor graphs, and belief propagation. *CoRR*, abs/1606.01307:1–46. 32, 49, 53

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*. 11, 13

Dahm, N., Gao, Y., Caelli, T., and Bunke, H. (2013). Matching non-aligned objects using a relational string-graph. In *2013 IEEE International Conference on Image Processing*, pages 3394–3398, Melbourne, VIC, Australia. IEEE. 45, 57

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, San Diego, CA, USA. 1

de las Heras, L.-P., Terrades, O. R., and Lladós, J. (2015). Attributed graph grammar for floor plan analysis. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 726–730, Tunis, Tunisia. IEEE. 36, 49, 55

de Souza Pio, J. L., de Oliveira, C. J. S., de Araújo, A. A., and de Oliveira, N. J. (2006). *Atualizações em Informática*, chapter Reconhecimento Sintático e Estrutural de Padrões. Editora PUC-Rio. 2, 3

Demir, I., Aliaga, D. G., and Benes, B. (2015). Procedural editing of 3d building point clouds. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2147–2155, Santiago, Chile. IEEE. 28, 52

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA. IEEE. 71

Deufemia, V., Risi, M., and Tortora, G. (2014). Sketched symbol recognition using latent-dynamic conditional random fields and distance-based clustering. *Pattern Recognition*, 47(3):1159 – 1171. Handwriting Recognition and other PR Applications. 47, 57

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, Vienna, Austria. 2, 4, 71, 73

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159. 64

Dyrmann, M., Jørgensen, R. N., and Midtiby, H. S. (2017). RoboWeedSupport - detection of weed locations in leaf occluded cereal crops using a fully convolutional neural network. *Advances in Animal Biosciences*, 8(2):842–847. 60

Eden, M. (1961). On the formalization of handwriting. *Am. Math. Soc. Appl. Math Symp*, 12:83–88. 18

Fang, H., Xu, Y., Wang, W., Liu, X., and Zhu, S. (2018). Learning pose grammar to encode human body configuration for 3d pose estimation. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 6821–6828, New Orleans, Louisiana, USA. AAAI Press. 35, 55

Feng, W., Liu, R., and Zhu, M. (2014). Fall detection for elderly person care in a vision-based home surveillance environment using a monocular camera. *Signal, Image and Video Processing*, 8(6):1129–1138. 25, 48, 52

Ferber, G. (1986). Classifying and validating intermittent eeg patterns with syntactic methods. *Pattern Recognition*, 19(4):289–295. 36

Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–264–II–271 vol.2. 8

Ferreira, A. D. S., Freitas, D. M., da Silva, G. G., Pistori, H., and Folhes, M. T. (2017). Weed detection in soybean crops using convnets. *Computers and Electronics in Agriculture*, 143:314 – 324. 60

Fire, A. and Zhu, S.-C. (2017). Inferring hidden statuses and actions in video by causal reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 48–56, Honolulu, HI, USA. IEEE. 30, 49, 54

Flasiński, M. and Jurek, J. (2014). Fundamental methodological issues of syntactic pattern recognition. *Pattern Analysis and Applications*, 17(3):465–480. 20

Forney, G. D. (2001). Codes on graphs: normal realizations. *IEEE Transactions on Information Theory*, 47(2):520–548. 32

Forsyth, D. A. and Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, USA. 18

Fu, K. S. (1974). *Syntactic methods in pattern recognition*, volume 112 of *Mathematics in Science and Engineering*. Academic Press. 2

Fu, K. S. (1982). *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ. 7

Fu, K.-S. and Rosenfeld, A. (1976). Pattern recognition and image processing. *IEEE Transactions on Computers*, C-25(12):1336–1346. 7, 8, 19, 62

Gadde, R., Marlet, R., and Paragios, N. (2016). Learning grammars for architecture-specific facade parsing. *Int. J. Comput. Vision*, 117(3):290–316. 40, 49, 56

Ghahramani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42. 43, 44

Gonfaus, J. M., Pedersoli, M., González, J., Vedaldi, A., and Roca, F. X. (2015). Factorized appearances for object detection. *Computer Vision and Image Understanding*, 138:92 – 101. 27, 52

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press. 8

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., Montreal, QC, Canada. 33, 34

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610. 15, 71

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232. 1, 5, 14, 35, 60, 61, 71, 77

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., and Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377. 2, 71

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA. 1

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. 1, 61, 71, 72, 77

Hentschel, C. and Sack, H. (2014). Does one size really fit all?: Evaluating classifiers in bag-of-visual-words classification. In *Proceedings of the 14th International Conference on Knowledge Technologies and Data-driven Business*, pages 1–7, New York, NY, USA. ACM. 47, 61

Hinton, G., Sabour, S., and Frosst, N. (2018). Matrix capsules with EM routing. In *6th International Conference on Learning Representations (ICLR)*, pages 1–15, Vancouver, BC, Canada. ICLR. 51

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In *Lecture Notes in Computer Science*, pages 44–51. Springer Berlin Heidelberg, Springer, Berlin, Heidelberg. 50, 51

Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531. 2, 73

Horn, G. V., Aodha, O. M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. J. (2018). The inaturalist species classification and detection dataset. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8769–8778. IEEE Computer Society. 79

Ikehata, S., Yang, H., and Furukawa, Y. (2015). Structured indoor modeling. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1323–1331, Santiago, Chile. IEEE. 40, 56

Isola, P. and Liu, C. (2013). Scene collaging: Analysis and synthesis of natural images with semantic layers. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3048–3055, Washington, DC, USA. IEEE. 47, 57

Jaakkola, T. S. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 487–493, Cambridge, MA, USA. MIT Press. 26

Jain, A. K., Duin, R. P. W., and Jianchang Mao (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37. 20

Jalal, A., Salman, A., Mian, A., Shortis, M., and Shafait, F. (2020). Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecological Informatics*, 57:101088. 18

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 675–678, New York, NY, USA. Association for Computing Machinery. 38

Jiang, C., Qi, S., Zhu, Y., Huang, S., Lin, J., Yu, L.-F., Terzopoulos, D., and Zhu, S.-C. (2018). Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision*, 126(9):920–941. 33, 54

Jiang, Y. and Ma, J. (2015). Combination features and models for human detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 240–248, Boston, MA, USA. IEEE. 3, 27, 52

Julca-Aguilar, F. D., Mouchère, H., Viard-Gaudin, C., and Hirata, N. S. T. (2017). A general framework for the recognition of online handwritten graphics. *CoRR*, abs/1709.06389:1–14. 28, 53

Kembhavi, A., Salvato, M., Kolve, E., Seo, M., Hajishirzi, H., and Farhadi, A. (2016). A diagram is worth a dozen images. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 235–251, Cham. Springer International Publishing. 47, 57

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 66, 81

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3581–3589, Cambridge, MA, USA. MIT Press. 50

Kirsch, R. A. (1964). Computer interpretation of english text and picture patterns. *IEEE Transactions on Electronic Computers*, EC-13(4):363–376. 18

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report. 18, 20

Kong, W. and Ranganath, S. (2014). Towards subject independent continuous sign language recognition: A segment and merge approach. *Pattern Recognition*, 47(3):1294 – 1308. Handwriting Recognition and other PR Applications. 44, 49, 56

Kortylewski, A., Wieczorek, A., Wieser, M., Blumer, C., Parbhoo, S., Morel-Forster, A., Roth, V., and Vetter, T. (2019). Greedy structure learning of hierarchical compositional models. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, June 16-20, 2019*, pages 11612–11621, Long Beach, CA, USA. Computer Vision Foundation / IEEE. 45, 57

Koziński, M., Gadde, R., Zagoruyko, S., Obozinski, G., and Marlet, R. (2015a). A mrf shape prior for facade parsing with occlusions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, Boston, MA, USA. IEEE. 39, 49, 56

Koziński, M. and Marlet, R. (2014). Image parsing with graph grammars and markov random fields applied to facade analysis. In *IEEE Winter Conference on Applications of Computer Vision*, pages 729–736, Steamboat Springs, CO, USA. IEEE. 40, 44, 49, 56

Koziński, M., Obozinski, G., and Marlet, R. (2015b). Beyond procedural facade parsing: Bidirectional alignment via linear programming. In Cremers, D., Reid, I., Saito, H., and Yang, M.-H., editors, *Computer Vision – ACCV 2014*, pages 79–94, Cham. Springer International Publishing. 39, 49, 56

Krüger, V. and Herzog, D. (2013). Tracking in object action space. *Computer Vision and Image Understanding*, 117(7):764 – 789. 43, 49, 56

Kuehne, H., Gall, J., and Serre, T. (2016). An end-to-end generative framework for video segmentation and recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8, Lake Placid, NY, USA. IEEE. 26, 48, 52

Kuehne, H., Richard, A., and Gall, J. (2017). Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78 – 89. Language in Vision. 26, 48, 52

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178, New York, NY, USA. IEEE. 47

Le, T. H. N., Zhu, C., Zheng, Y., Luu, K., and Savvides, M. (2017). Deepsafedrive: A grammar-aware driver parsing approach to driver behavioral situational awareness (db-saw). *Pattern Recognition*, 66:229 – 238. 27, 48, 52

Lee, K., Ognibene, D., Chang, H. J., Kim, T.-K., and Demiris, Y. (2015). Stare: Spatio-temporal attention relocation for multiple structured activities detection. *IEEE Transactions on Image Processing*, 24(12):5916–5927. 29, 49, 53

Lemus, E., Bribiesca, E., and Garduno, E. (2015). Surface trees representation of boundary surfaces using a tree descriptor. *Journal of Visual Communication and Image Representation*, 31:101 – 111. 46, 57

Li, B., Chen, Y., and Wang, F.-Y. (2015). Pedestrian detection based on clustered poselet models and hierarchical and-or grammar. *IEEE Transactions on Vehicular Technology*, 64(4):1435–1444. 31, 49, 53

Li, B., Song, X., Wu, T., Hu, W., and Pei, M. (2014). Coupling-and-decoupling: A hierarchical model for occlusion-free object detection. *Pattern Recognition*, 47(10):3254 – 3264. 31, 49, 53

Li, X., Song, X., and Wu, T. (2019). Aognets: Compositional grammatical architectures for deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6220–6230, Long Beach, CA. IEEE. 41, 56

Li, X., Wu, T., Song, X., and Krim, H. (2017). Aognets: Deep AND-OR grammar networks for visual recognition. *CoRR*, abs/1711.05847:1–12. 18

Liu, L., Wang, S., Peng, Y., Huang, Z., Liu, M., and Hu, B. (2016). Mining intricate temporal rules for recognizing complex activities of daily living under uncertainty. *Pattern Recognition*, 60:1015 – 1028. 44, 49, 57

Liu, X., Ji, R., Wang, C., Liu, W., Zhong, B., and Huang, T. S. (2015). Understanding image structure via hierarchical shape parsing. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5042–5050, Boston, MA, USA. IEEE. 44, 49, 57

Liu, X., Xu, Y., Zhu, L., and Mu, Y. (2018a). A stochastic attribute grammar for robust cross-view human tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2884–2895. 22, 38, 55

Liu, X., Zhao, Y., and chun Zhu, S. (2014). Single-view 3d scene parsing by attributed grammar. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 684–691, Columbus, OH, USA. IEEE. 55

Liu, X., Zhao, Y., and Zhu, S.-C. (2018b). Single-view 3d scene reconstruction and parsing by attribute grammar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):710–725. 37, 38, 49, 55

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110. 1, 3, 9, 61

Lu, Y., Wu, T., and Zhu, S.-C. (2014). Online object tracking, learning, and parsing with and-or graphs. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3462–3469, Columbus, OH, USA. IEEE. 31, 49, 53

Luong, T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *CoRR*. 1

Martinovic, A. and Gool, L. V. (2013). Early parsing for 2d stochastic context free grammars. Technical Report KUL/ESAT/PSI/1301, Department of Electrical Engineering (ESAT), University Hospital Gasthuisberg, Kasteelpark Arenberg, België. 35

Martinovic, A. and Van Gool, L. (2013). Bayesian grammar learning for inverse procedural modeling. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 201–208, Washington, DC, USA. IEEE Computer Society. 3, 35, 54

Mihalkova, L., Huynh, T., and Mooney, R. J. (2007). Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 608–614, Vancouver, British Columbia, Canada. AAAI Press. 43

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In Kobayashi, T., Hirose, K., and Nakamura, S., editors, *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA. 1

Moore, D. and Essa, I. (2002). Recognizing multitasked activities from video using stochastic context-free grammar. In *Eighteenth National Conference on Artificial Intelligence*, page 770–776, USA. American Association for Artificial Intelligence. 49, 50

Morency, L.-P., Quattoni, A., and Darrell, T. (2007). Latent-dynamic discriminative models for continuous gesture recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, USA. IEEE. 47

Mundhenk, T. N., Chen, B. Y., and Friedland, G. (2019). Efficient saliency maps for explainable AI. *CoRR*, abs/1911.11293. 89

Narasimhan, R. (1962). A linguistic approach to pattern recognition. Technical Report 121, Digital Computer Laboratory, University of Illinois, Urbana, Illinois. 18

Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 841–848, Cambridge, MA, USA. MIT Press. 20, 50, 60

Nguyen, N.-D., Do, T., Ngo, T. D., and Le, D.-D. (2020). An evaluation of deep learning methods for small object detection. *Journal of Electrical and Computer Engineering*, 2020:1–18. 77

Ojala, T., Pietikainen, M., and Harwood, D. (1994). Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of 12th International Conference on Pattern Recognition*, volume 1, pages 582–585 vol.1, Jerusalem, Israel, Israel. IEEE. 31

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 86

Oyallon, E. and Rabin, J. (2015). An analysis of the surf method. *Image Processing On Line*, 5:176–218. 9

Özkural, E. (2014). An application of stochastic context sensitive grammar induction to transfer learning. In Goertzel, B., Orseau, L., and Snaider, J., editors, *Artificial General Intelligence*, pages 121–132, Cham. Springer International Publishing. 41, 42

Park, S., Nie, B. X., and Zhu, S.-C. (2018). Attribute and-or grammar for joint parsing of human pose, parts and attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(7):1555–1569. 41, 56

Park, S. and Zhu, S.-C. (2015). Attributed grammars for joint estimation of human attributes, part and pose. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2372–2380, Santiago, Chile. IEEE. 38, 55

Pedro, R. W. D. (2013). Inferência de gramáticas estocásticas para reconhecimento de padrões de imagens utilizando quadtrees. Master's thesis, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo. 8

Pedro, R. W. D., Nunes, F. L. S., and Machado-Lima, A. (2013). Using grammars for pattern recognition in images: A systematic review. *ACM Comput. Surv.*, 46(2):26:1–26:34. 19, 21

Pei, M., Si, Z., Yao, B. Z., and Zhu, S.-C. (2013). Learning and parsing video events with goal and intent prediction. *Comput. Vis. Image Underst.*, 117(10):1369–1383. 42, 56

Pfaltz, J. L. and Rosenfeld, A. (1969). Web grammars. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, IJCAI'69, pages 609–619, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 28

Pirsiavash, H. and Ramanan, D. (2014). Parsing videos of actions with segmental grammars. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 612–619, Washington, DC, USA. IEEE Computer Society. 26, 48, 52

Pistori, H., Calway, A., and Flach, P. (2013). A new strategy for applying grammatical inference to image classification problems. In *2013 IEEE International Conference on Industrial Technology (ICIT)*, pages 1032–1037, Cape Town, South Africa. IEEE. 18, 46, 57, 61, 64, 72

Potena, C., Nardi, D., and Pretto, A. (2017). Fast and accurate crop and weed identification with summarized train sets for precision agriculture. In *Intelligent Autonomous Systems 14*, pages 105–121. Springer International Publishing. 60

Qi, S., Huang, S., Wei, P., and Zhu, S.-C. (2017). Predicting human activities using stochastic grammar. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1173–1181, Venice, Italy. IEEE. 30, 49, 54

Qi, S., Zhu, Y., Huang, S., Jiang, C., and Zhu, S.-C. (2018). Human-centric indoor scene synthesis using stochastic grammar. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5899–5908, Salt Lake City, UT, USA. IEEE. 33, 54

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149. 71, 77

Robert, C. P. and Casella, G. (1999). The metropolis—hastings algorithm. In *Springer Texts in Statistics*, pages 231–283. Springer New York, New York, NY. 32

Rodríguez, A. F., Müller, H., and Depeursinge, A. (2017). From visual words to a visual grammar: using language modelling for image classification. *CoRR*, abs/1703.05571:1–17. 47, 57

Rothrock, B., Park, S., and Zhu, S.-C. (2013). Integrating grammar and segmentation for human pose estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3221, Portland, OR, USA. IEEE. 32, 53

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3859–3869, Red Hook, NY, USA. Curran Associates Inc. 51

Santos, A., Junior, J. M., de Andrade Silva, J., Pereira, R., Matos, D., Menezes, G., Higa, L., Eltner, A., Ramos, A. P., Osco, L., and Gonçalves, W. (2020). Storm-drain and manhole detection using the RetinaNet method. *Sensors*, 20(16):4450. 18

Sarawagi, S. and Cohen, W. W. (2004). Semi-markov conditional random fields for information extraction. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 1185–1192. MIT Press, Cambridge, MA, USA. 44

Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681. 34

Sethi, R. J. and Roy-Chowdhury, A. K. (2010). Modeling and recognition of complex multi-person interactions in video. In *Proceedings of the 1st ACM International Workshop on Multimodal Pervasive Video Analysis*, MPVA '10, pages 43–46, New York, NY, USA. ACM. 46

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015*, pages 1–14, San Diego, CA, USA. ICLR. 33

Slonneger, K. and Kurtz, B. (1995). *Formal Syntax and Semantics of Programming Languages: A Laboratory Based Approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition. 36, 49

Song, X., Wu, T., Jia, Y., and Zhu, S.-C. (2013). Discriminatively trained and-or tree models for object detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3278–3285, Portland, OR, USA. IEEE. 31, 53

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958. 64

Stiny, G. and Gips, J. (1971). Shape grammars and the generative specification of painting and sculpture. In *Information Processing, Proceedings of IFIP Congress*, volume 2, pages 1460–1465, Ljubljana, Yugoslavia. Elsevier, North Holland Publishing Co. 39, 49

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press. 1, 11, 77

Tabernik, D., Kristan, M., Wyatt, J. L., and Leonardis, A. (2016). Towards deep compositional networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, December 4-8, 2016*, pages 3470–3475, Cancún, Mexico. IEEE. 48, 50, 51, 58

Tabernik, D., Leonardis, A., Boben, M., Skočaj, D., and Kristan, M. (2015). Adding discriminative power to a generative hierarchical compositional model using histograms of compositions. *Comput. Vis. Image Underst.*, 138(C):102–113. 51

Tayyub, J., Hawasly, M., Hogg, D. C., and Cohn, A. G. (2018). Learning hierarchical models of complex daily activities from annotated videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1633–1641, Lake Tahoe, NV, USA. IEEE. 30, 49, 54

Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., and Paragios, N. (2011). Shape grammar parsing via reinforcement learning. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 2273–2280, Washington, DC, USA. IEEE Computer Society. 44

Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., and Paragios, N. (2013). Parsing facades with shape grammars and reinforcement learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(7):1744–1756. 40, 49, 55

Tetila, E. C., Machado, B. B., Astolfi, G., de Souza Belete, N. A., Amorim, W. P., Roel, A. R., and Pistori, H. (2020). Detection and classification of soybean pests using deep learning with uav images. *Computers and Electronics in Agriculture*, 179:105836. 18, 79

Tetila, E. C., Machado, B. B., Belete, N. A. d. S., Guimarães, D. A., and Pistori, H. (2017). Identification of soybean foliar diseases using unmanned aerial vehicle images. *IEEE Geoscience and Remote Sensing Letters*, 14(12):2190–2194. 60

Tian, B., Tang, M., and Wang, F.-Y. (2015). Vehicle detection grammars with partial occlusion handling for traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 56:80 – 93. 18, 27, 48, 52

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877. 2, 4, 71, 73

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on*

Neural Information Processing Systems, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc. vi, 1, 11, 12, 71, 73, 77

Vo, N. N. and Bobick, A. F. (2014). From stochastic grammar to bayes network: Probabilistic parsing of complex activity. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2641–2648, Columbus, OH, USA. IEEE. 29, 49, 53

Vo, N. N. and Bobick, A. F. (2016). Sequential interval network for parsing complex structured activity. *Computer Vision and Image Understanding*, 143:147 – 158. Inference and Learning of Graphical Models Theory and Applications in Computer Vision and Image Analysis. 29, 49, 53

Walton, M., Lange, D., and Zhu, S.-C. (2017). Inferring context through scene understanding. In *AAAI Spring Symposium Series*, pages 356–360, Palo Alto, California, CA. AAAI Press. 32, 49, 54

Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013). Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79. 26

Wang, W., Wang, W., Xu, Y., Shen, J., and Zhu, S.-C. (2018). Attentive fashion grammar network for fashion landmark detection and clothing category classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4271–4280, Salt Lake City, UT, USA. IEEE. 33, 54

Weissenberg, J., Riemenschneider, H., Prasad, M., and Gool, L. V. (2013). Is there a procedural logic to architecture? In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–192, Washington, DC, USA. IEEE. 39, 49, 55

Wilson, A. D. and Bobick, A. F. (1999). Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900. 43

Windridge, D., Kittler, J., de Campos, T., Yan, F., Christmas, W., and Khan, A. (2015). A novel markov logic rule induction strategy for characterizing sports video footage. *IEEE MultiMedia*, 22(2):24–35. 43, 49, 57

Wu, B. (2013). Two-dimensional (2D) languages and application to handwritten graphical parsing. Technical report, Ecole Polytechnique de l'université de Nantes. 22, 25, 28, 34

Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., and Vajda, P. (2020). Visual Transformers: Token-based Image Representation and Processing for Computer Vision. *arXiv e-prints*, page arXiv:2006.03677. 2, 4, 73

Wu, Y. N., Si, Z., Gong, H., and Zhu, S.-C. (2009). Learning active basis model for object detection and recognition. *International Journal of Computer Vision*, 90(2):198–235. 45

Xing, X., Wu, T., Zhu, S., and Wu, Y. N. (2020). Inducing hierarchical compositional model by sparsifying generator network. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, June 13-19, 2020*, pages 14284–14293, Seattle, WA, USA. IEEE. 34, 51, 54

Xing, X., Zhu, S.-C., and Wu, Y. N. (2019). Inducing sparse coding and and-or grammar from generator network. In *AAAI Conference on Artificial Intelligence, Workshop on Network Interpretability for Deep Learning*, pages 1–4, Honolulu, Hawaii, USA. AAAI Press. 33, 51, 54

Xu, Y., Qin, L., Liu, X., Xie, J., and Zhu, S.-C. (2018). A causal and-or graph model for visibility fluent reasoning in tracking interacting objects. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2178–2187, Salt Lake City, Utah. IEEE. 30, 49, 54

Zarchi, M., Tan, R., van Gemeren, C., Monadjemi, A., and Veltkamp, R. (2016). Understanding image concepts using istop model. *Pattern Recognition*, 53:174 – 183. 18, 42, 56

Zhao, Y. and Zhu, S.-C. (2013). Scene parsing by integrating function, geometry and appearance models. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3119–3126, Portland, OR, USA. IEEE. 32, 49, 53

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics. 15, 71

Zhu, Y., Nayak, N., Gaur, U., Song, B., and Roy-Chowdhury, A. (2013). Modeling multi-object interactions using string of feature graphs. *Computer Vision and Image Understanding*, 117(10):1313 – 1328. 45, 57

Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., and Hu, S. (2016). Traffic-sign detection and classification in the wild. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 77

Zhu L. Chen, Y. Yuille, A. (2009). Unsupervised learning of probabilistic grammar-markov models for object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):114. 8

Zieliński, B., Skomorowski, M., Wojciechowski, W., Korkosz, M., and Sprężak, K. (2015). Computer aided erosions and osteophytes detection based on hand radiographs. *Pattern Recognition*, 48(7):2304 – 2317. 40, 56

Appendices

A Technical and Scientific Production

A.1 Papers published in journals

Astolfi, G., Rezende, F. P. C., Porto, J. V. D. A., Matsubara, E. T., and Pistori, H. (2021). Syntactic pattern recognition in computer vision. *ACM Computing Surveys*, 54(3):1–35.

Astolfi, G., Pache, M. C. B., Menezes, G. V., da Silva Oliveira Junior, A., Menezes, G. K., de Weber, V.A. M., Tetila, E. C., de Souza Belete, N. A., Matsubara, E. T., and Pistori, H. (2020). Combining syntactic methods with LSTM to classify soybean aerial images. *IEEE Geoscience and Remote Sensing Letters*, 1(1):1–5.

Astolfi, A. C. M. N., Astolfi, G., Ferreira, M. G. A., Centurião, T. D., Clemente, L. Z., de Oliveira, B. L. M. C., de Andrade Porto, J. V., Roche, K. F., Matsubara, E. T., Pistori, H., Soares, M. P.,and da Silva, W. M. (2021). Recognizing and counting dendrocephalus brasiliensis (crustacea:Anostraca) cysts using deep learning. *PLOS ONE*, 16(3):e0248574.

Astolfi, G., Gonçalves, A. B., Menezes, G. V., Borges, F. S. B., Astolfi, A. C. M. N., Matsubara,E. T., Alvarez, M., and Pistori, H. (2020). POLLEN73s: An image dataset for pollen grains classification. *Ecological Informatics*, 60:101165.

Weber, V. A. M., de Lima Weber, F., da Silva Oliveira, A., Astolfi, G., Menezes, G. V., de Andrade Porto, J. V., Rezende, F. P. C., de Moraes, P. H., Matsubara, E. T., Mateus, R. G., de Araújo,T. L. A. C., da Silva, L. O. C., de Queiroz, E. Q. A., de Abreu, U. G. P., da Costa Gomes, R., and Pistori, H. (2020). Cattle weight estimation using active contour models and regression trees bagging. Computers and *Electronics in Agriculture*, 179:105804.

Tetila, E. C., Machado, B. B., Astolfi, G., de Souza Belete, N. A., Amorim, W. P., Roel, A. R., and Pistori, H. (2020). Detection and classification of soybean pests using deep learning with UAV images. *Computers and Electronics in Agriculture*, 179:105836.

Costa, C. S., Tetila, E. C., Astolfi, G., Sant'Ana, D. A., Pache, M. C. B., Gonçalves, A. B., Zanoni, V.A. G., Nucci, H. H. P., Diemer, O., and Pistori, H. (2019). A computer vision system for oocyte counting using images captured by smartphone. *Aquacultural Engineering*, 87:102017.

Tetila, E. C., Machado, B. B., Menezes, G. V., de Souza Belete, N. A., Astolfi, G., and Pistori, H. (2020). A deep-learning approach for automatic counting of soybean insect pests. *IEEE Geoscience and Remote Sensing Letters*, 17(10):1837–1841.

A.2 Papers Published in Conference Proceedings

Astolfi, G., de Moares Weber, V. A., Junior, A. D. S. O., Menezes, G. V., de Souza Belete, N. A., Tetila, E. C., and Pistori, H. (2019). Using syntactic methods and LSTM to the recognition of objects visual patterns. In *Anais do Workshop de Visão Computacional (WVC)*. Sociedade Brasileira de Computação - SBC.

de Souza Belete, N. A., Tetila, E. C., Astolfi, G., and Pistori, H. (2019). Classification of weed in soybean crops using unmanned aerial vehicle images. In *Anais do Workshop de Visão Computacional(WVC)*. Sociedade Brasileira de Computação - SBC.

A.3 Book Chapter Published

de Almeida Filho, O. P., Frihling, B. E. F., de Oliveira, M. A. C., Astolfi, G., Pistori, H., and Migliolo,L. (2019). Avaliação e seleção in silico de peptídeos antimicrobianos bioinspirados na l-aminoácido oxidase da serpente bothrops jararacussu. In Prospecção de Moléculas Bioativas em Animais e Plantas: Uma Visão Biotecnológica, pages 71–80. Atena Editora.

A.4 Abstracts Published in Conference Proceedings

Weber, F. L., Weber, V. A. M., Abreu, U. G. P., Pistori, Oliveira Junior, A. S., Astolfi, G., Moraes, P. H. Identificação de bovinos das raças Nelore e Girolando usando redes neurais profundas. In: Encontro de Ciências Agrárias, 2018, Aquidauana, MS. *Encontro de Ciências Agrárias Unidade Universitária de Aquidauana*, 2018.

Goncalves, A. B., Astolfi, G., Pistori. Using computer vision, machine learning and remote sensing to identify built areas and vegetation cover. In: IEEE/GRSS-Young Professionals & ISPRS WG V/5 and Student Consortium SS, 2018, Campo Grande, MS. *Proceedings of the IEEE/GRSS-Young Professionals & ISPRS WG V/5 and Student Consortium SS*, 2018. p. 33-33.

Weber, V. A. M., Weber, F. L., Oliveira Junior, A. S., Astolfi, G., Machado, M. I. C., Bonin, M. N., Itavo, L. C. V., Leal, E. S., Marcato Junior, J., Pistori. Cattle segmentation using uavs supported by computer vision techniques: preliminary results. In: IEEE/GRSS-Young Professionals & ISPRS WG V/5 and Student Consortium SS, 2018, Campo Grande, MS. *Proceedings of the IEEE/GRSS-Young Professionals & ISPRS WG V/5 and Student Consortium SS*, 2018. p. 43-44.

Weber, V. A. M., Weber, F. L., Oliveira Junior, A. S., Astolfi, G., Bonin, M. N., Itavo, L. C. V., Leal, E. S., Marcato Junior, J., Salgado Filho, G. R. Pistori. Prediction of live weight of cattle using UAVs: preliminary results. In: IEEE/GRSS-Young Professionals & ISPRS WG V/5 and Student Consortium SS, 2018, Campo Grande, MS. *Proceedings of the IEEE/GRSS-Young Professionals & ISPRS WG V/5 and Student Consortium SS*, 2018. p. 19-20.

A.5 Registered Software

Ferreira, A. S. , Silva, G. G., Weber, V. A. M., Menezes, G. K., Astolfi, G. , Pistori, H., Menezes, G. V., Goncalves, D. N. , Tetila, E. C., Oliveira Junior, A. S. , Borges, F. S. B. PYNOVISÃO. 2018. Número do registro: BR512019000427-2, data de registro: 10/01/2018, título: "PYNOVISÃO" , Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.

Migliolo, L., Pistori, H. , Astolfi, G., Oliveira, M. A. C. Antimicrobial Peptide Pattern. 2018. Número do registro: BR512019000564-3, data de registro: 01/06/2018, título: "Antimicrobial Peptide Pattern" , Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.

Dotto, F. , Weber, V. A. M., Menezes, G. V., Weber, F. L., Oliveira Junior, A. S., Astolfi, G., Espejo, R. A., Pistori, H., Oliveira, M. A. C., Garcia, R. A. M., Araujo, R. V. 2D4D Reader. 2019. Número do registro: BR512019001042-6, data de registro: 05/02/2019, título: "2D4D Reader", Instituição de registro: INPI - Instituto Nacional da Propriedade Industrial.

A.6 Papers Submitted to Journals

Graph to sequence to class: using syntactic pattern recognition for image classification problems. Astolfi, G., Sant'Ana, D. A., Porto, J., V., de A., Rezende, F., P., C., Tetila, E., C., Matsubara, E., T., Pistori, H. Paper recently submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.