1. The Tree Restructuring Algorithm

The tree restructuring algorithm is shown in Alg. 1. The TreeRestructure procedure is used on an MP4 tree $T$ to generate the restructured tree $T_r$.

2. String Processing

The string processing step consists of the following steps:

- Noise removal: non-printable and non-ASCII characters are removed from the input string
- String partition: the input string is broken down into parts separated by white space characters, where each part can be a textual word, URL, version string, ratio string, or numeric string
- URL decomposition: the input strings can also contain URLs which are decomposed into multiple parts; for example, www.example.com/about/index.html will be broken down into www.example.com, about, index.html
- Version string decomposition: version strings found in the input string are decomposed into multiple numeric parts; the numeric parts are surrounded by two special words #VER_BEG and #VER_END to emphasize that they are the members of a version string
- Ratio string conversion: ratio strings such as 1/48000 found in the input string are converted into their corresponding numeric values
- Numeric string conversion: string representations of floating point numbers and integers found in the input string are converted into their corresponding numeric values

3. MTN Model Details

3.1. Graph Pooling (GP) Readout

Denote the output of the Graph Pooling (GP) layer by $E'$ . The first dimension of $E'$ is variable. To produce a fixed-sized vector for classification, a readout mechanism [1] is defined as follows:

$$\text{ReadOut}(E') = \text{MaxP}(E') \parallel \text{MinP}(E') .$$

Here, MaxP denotes the max pooling operation, and $\parallel$ denotes concatenation. The readout operation always produces a $2C$-dimensional vector, which can be used by a classifier. Here, $C$ is the number of “channels” in the Graph Convolution (GC) layer.

In MTN, we use the hierarchical pooling architecture described in [1]. That is, there are multiple GC+GP layers in the Graph Neural Network (GNN). The readout operation is used for each GP layer in the GNN, and the results are summed together to generate a single vector for classification.

3.2. Model Size of the Node Embedding Network (NEN)

The relationship between the choice of $M$ (embedding size) and the number of parameters in the Node Embedding Network (NEN) are shown in Tab. 1.

<table>
<thead>
<tr>
<th>$M$</th>
<th>128</th>
<th>256</th>
<th>384</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. Parameters</td>
<td>6.56M</td>
<td>15.39M</td>
<td>26.52M</td>
</tr>
</tbody>
</table>

3.3. Model Architecture

We implemented MTN using the PyTorch library. We used the GC and GP implementation from the PyG library. We used PyTorch’s transformer encoder implementation (TransformerEncoderLayer). In the number encoder, we chose $\tau_{\text{max}} = 10^{12}$ and $\tau_{\text{min}} = -\tau_{\text{max}}$ (see Sec. 2.2.1). The String Object List (SOL) encoder in the NEN contains 4 transformer encoder layers with $M = 384$ and $h = 4$. The Graph Attention Network (GATN) in the Graph Analysis Module (GAM) contains 6 transformer encoder layers with $M = 384$ and $h = 4$. The choice of $M$ is discussed in Sec. 4.1. For GC, we used the GCNConv layer from the PyG library with $C = 128$. For GP, we used the SAGPool layer from the PyG library with $\alpha = 0.5$. The definition of $C$ and $\alpha$ can be found in the PyTorch documentation [1].

---

2 https://pytorch-geometric.readthedocs.io/en/latest/
Algorithm 1: The tree restructuring procedure.

```
1 Procedure TreeRestructure(T):
    Input: MP4 tree T
    Output: restructured MP4 tree T_r
    Create a new empty tree T_r;
    // Make sure the new tree T_r has the same topology as T
    foreach node n in T do
        Create a new node n' in T_r with tag n;
        foreach child node c of n do
            Find the corresponding nodes of n and c in T_r, which are denoted by n' and c';
            Connect n' → c' in T_r;
        endforeach
    endforeach
    // Add data from T to T_r
    foreach node n in T do
        Find the corresponding node of n in T_r, which is denoted by n';
        foreach key-value pair (key,val) in n do
            RecursiveAdd((key,val), n');
        endforeach
    endforeach
    return T_r

2 Procedure RecursiveAdd(obj, u'):
    Input: an object to be added (obj), which can be a string, a number, a key-value pair, a list, a dictionary; the node where the data is added (u')
    // Recursively adds data into the restructured tree T_r
    if obj is a number or a string then
        Create a new node v' in T_r with tag v;
        Set the data of v' as obj;
        Connect u' → v' in T_r;
    else if obj is a key-value pair (key,val) then
        Create a new node k' in T_r with tag k;
        Set the data of k' as key;
        Connect u' → k' in T_r;
        RecursiveAdd(val, k');
    else if obj is a dictionary then
        Create a new node w' in T_r with tag n;
        Set the data of w' as #DICT;
        Connect u' → w' in T_r;
        foreach (key,val) in obj do
            RecursiveAdd((key,val), w');
        endforeach
    else if obj is a list then
        Create a new node w' in T_r with tag 1;
        Set the data of w' as #LIST;
        Connect u' → w' in T_r;
        foreach item in obj do
            RecursiveAdd(item, w');
    endforeach
```

found in Sec. 2.3. The MTN has 3 GC+GP layers. Under this setting, the GC+GP layers contain 281k parameters. For classification, we used a three layer multilayer perceptron network with $2C, 2C, K$ neurons. Here, $K$ is the number of classes in the video forensics task.

References