Neural Networks
Decision trees to understand CNNs

Giammarco D'Alessandro 1753102
Maria Rosaria Fraraccio 1760072
Luca Gioffrè 1763652

October 2020

1 Introduction

Convolutional neural networks are nowadays widely used for different tasks in many fields, becomes thus important to understand what knowledge a CNN learns.
In order to do so, based on [1], we modified a VGG-16 network by adding a mask layer, trained it for image classification and subsequently, based on [2], built a decision tree that could help us explain which object parts contributed the most to the final prediction and quantify these contributions.

These are indeed described by the decision modes represented in the tree nodes that allow us to know what were the filters that contributed to the final prediction given an input image.
In conclusion we evaluated the predictions made with several metrics, in order to know how accurate they are and where the error comes from.

2 Dataset and Architecture

2.1 Datasets

The dataset on which we realized binary classification includes the PASCAL-Part Dataset [3] and the CUB200-2011 dataset [4], in particular the annotated versions, useful when executing the evaluation metrics. However, the annotations are not required during the training of the CNN, hence any of the severl pre-trained CNNs on the ImageNet dataset can be used.

The CUB200-2011 dataset contains images of birds, which is our target category, while from the PASCAL-Part Dataset we took images of other animal categories.

Thus, our entire dataset Ω can be described as the union of the set of birds' images Ω^+ and the set of other animals' images Ω^-.

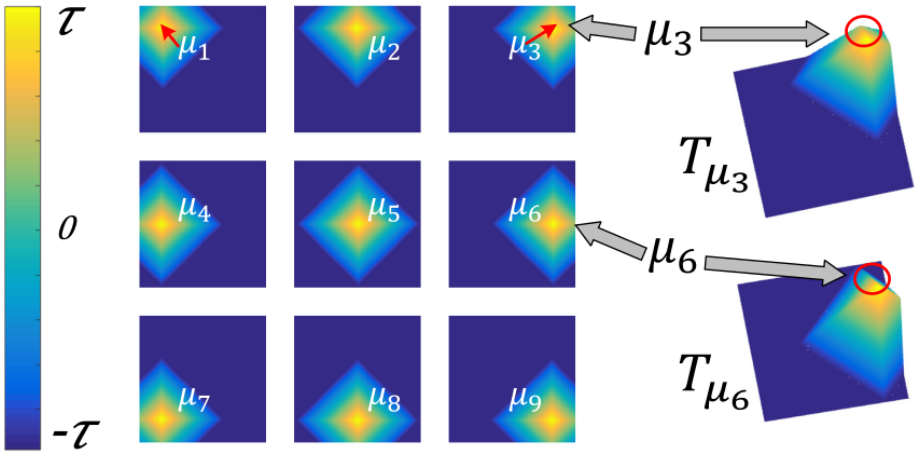


Figure 1: Explanatory figure on how the mask are applied. The center μ_i for the mask T_{μ_i} applied to the feature map x, is chosen as $\mu_i = argmax_{\mu=[i,j]} x_{ij}; 1 \leq i, j \leq n$

2.2 Network

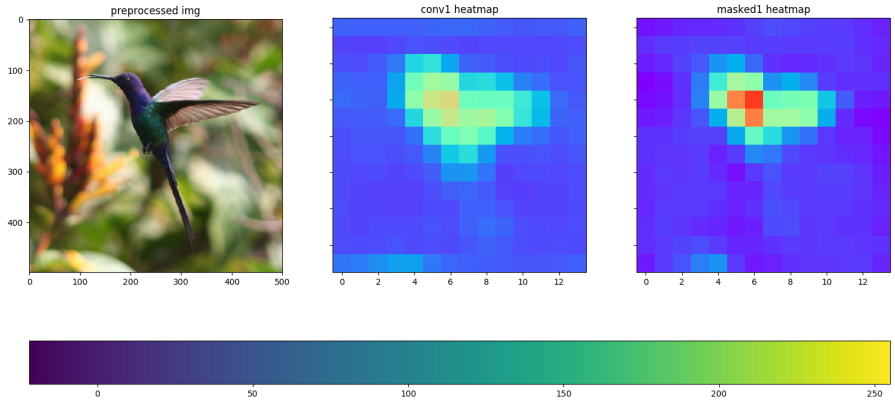
The CNN we used is based on a VGG-16 network [5], hence it was trained on the ImageNet dataset, in particular on its subset ILSVRC 2013 DET Animal-Part dataset that contains images of 1000 different categories.
So as to make it interpretable, we slightly modified the VGG-16 by adding a mask layer above the last conv-layer. In classical CNNs, a filter can be activated by multiple object parts, while what we want in our network is for a filter to be activated by a single part, the most significant one. The mask is just to filter out noisy activations and keep a single activation peak.

In order to compute the right mask for the given filter, we consider the neural unit with the strongest activation as the target part location and apply the mask centered in that location.

To create the mask we can use different kinds of norms, as L1 norm or L2 norm. However, for computational reason, we used the L1 norm. The image in Fig.1 shows a toy example [1], with the mask T_{μ_i} applied in nine different locations (μ_i).

The following images (Fig.2) show the usefulness of using a mask layer; the first is a comparison between the averaged input of the first dense layer with the mask applied and without, while the second shows 32 of the 512 filters individually.

In order to then use it for learning a decision tree for birds category, the network was finally fine-tuned on bird and not-bird categories using 20000 images approximately.



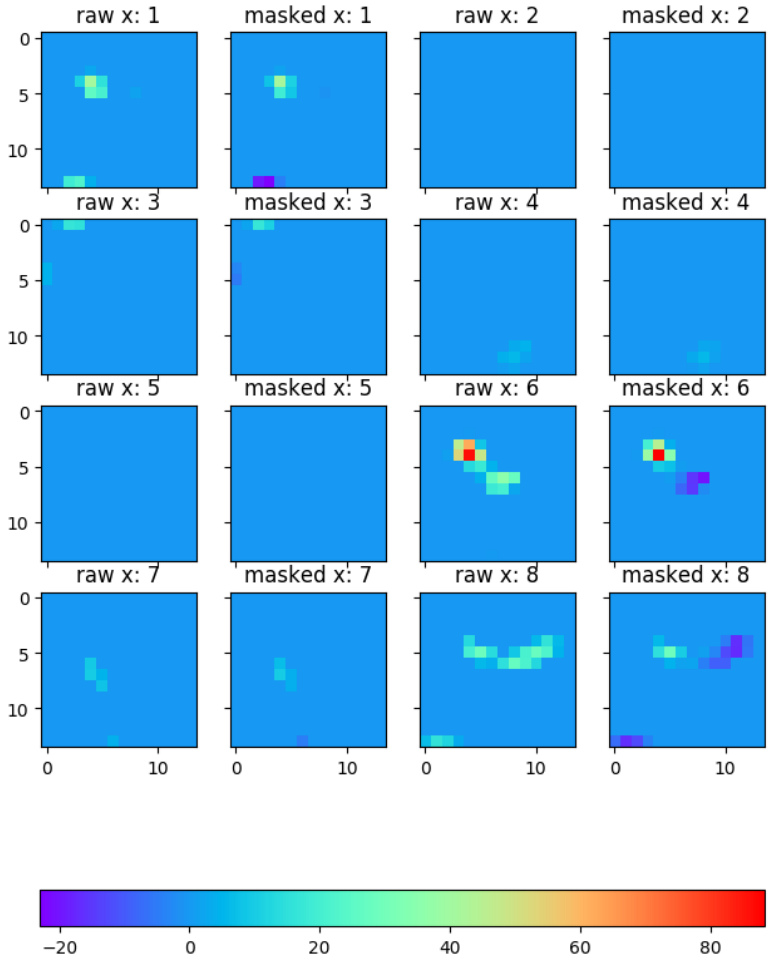


Figure 2: The (averaged) heatmap of all feature maps after the last convolutional layer, before and after applying the masks (top image) and the first 8 filters at the same layer both before and after the mask application (bottom image). We can clearly see that the mask operation enhances the most activated pixel and reduces the intensity of the others, resulting in a more disentagled feature map.

3 Building a decision tree

3.1 Algorithm

```
Initialize a tree P_0 and set t = 0;
for I_i, i ∈ Ω^+:
    g = g_i
    α = 1
    P_0.addLeaf(g, α)

while ΔlogE > 0:
    t = t + 1
    nodes ← P_{t-1} second layer
    (v, v') ∈ nodes s.t. maxΔlogE
    P_t ← P_{t-1}
    u = P_t.merge(v, v')
    g_u = max Σ cos(g_i g_u)
    α_u = min 1/‖Ω‖ Σ(w x_i + b − y_i)^2 + λ‖α‖

return P_t

Assign filters with semantic object parts to obtain A:
A ← 0
for I_i ∈ Ω^+:
    x_i ← feature map_i
    a_centers ← [ ]
    for a ∈ annotations_i:
        a_centers.append(center(a))
    for d ∈ D:
        obj_part ← mindist(a_centers, center(RF(x_i[d])))
        A[d][obj_part] ← 1
```

3.1.1 Explanation

i. Instantiate a leaf node for each image in the dataset belonging to the target category with the following attributes:

- x_i: feature map of the top-conv layer after a ReLU and mask operation;
- y_i: classification score of the target category before the softmax operation;
- g_i: rationale of the prediction that can be expressed as $\frac{\partial y}{\partial x}$;
- $\alpha \in \{0,1\}^D$: binary selection of filters that determines the decision modes, in the leaves $\alpha = [1, 1, ..., 1]^T$;

- $w = \alpha \circ g$: rationale of the decision mode, indeed we can see that in the leaves it coincides with g.

ii. Select two nodes, v and v', such that they maximize $\Delta logE$ (see section below).

iii. Merge the two chosen nodes in a new one u and determine its parameters based on the following formulas:

(a) $\bar{g}_u = max_{\bar{g}} \sum_{i \in \Omega_u} cosine(g_i, \bar{g}) \quad \text{s.t. } \bar{g}^T \bar{g} = 1$

(b) $\alpha_u = min_\alpha \frac{1}{\|\Omega_u\|} \sum_{i \in \Omega_u} (w^T x_i + b - y_i)^2 + \lambda \|\alpha\|_1$

(c) $h_u(x_i) = w^T x_i + b$

These computations can be simplified as the following:

(a) $\bar{g}_u = max_{\bar{g}} \sum_{i \in \Omega_u} \frac{g_i \bar{g}}{\|g_i\| \|\bar{g}\|} = max_{\bar{g}} \sum_{i \in \Omega_u} g_i \bar{g}$ and it can be solved as an optimization problem with g being a unit vector.

(b) This can be solved as a Lasso regression [6] for feature selection with penalty parameter set to $\lambda = 10^{-6} \sqrt{\|\Omega_u\|}$ [2].

We indicate the set of u's children as $\Omega_u = \Omega_v \cup \Omega_{v'}$, where v and v' are the two chosen nodes used.

At the end of the process we will have a decision tree that hierarchically organizes decision modes in a coarse-to-fine manner.

At this point we build a matrix $A \in \{0, 1\}^{MxD}$ such that we can instantiate a relation between filters and semantic object parts, where D is the number of filters (in the VGG16 architecture, $D = 512$) and M is the number of object parts we want to recognize (in our case $M = 4$). In order to do so, we use the annotation in the dataset and we group them in this way:

- **Head parts:** head, beak and eyes annotations
- **Torso parts:** torso, neck and wings annotations
- **Leg parts:** legs and feet annotations
- **Tail parts:** tails annotations

3.2 Implementation

After having instantiated all the leaves, it is not trivial to choose the which nodes to merge, so we show the unravelled formula below:

To make it more understandable we define:

$$\check{h}(x_i) = e^{\gamma h(x_i)}, \quad P_t(x_i) = \frac{\check{h}(x_i)}{\sum_{j\in\Omega}\check{h}(x_j)};$$

$$\Delta logE = log\frac{E_{t+1}}{E_t} = log\Big(\frac{\prod_{i\in\Omega^+}P_{t+1}}{\prod_{i\in\Omega^+}P_0}\frac{\prod_{i\in\Omega^+}P_0}{\prod_{i\in\Omega^+}P_t}\frac{e^{\|V_t\|}}{e^{\|V_{t+1}\|}}\Big) = log\Big(e\frac{\prod_{i\in\Omega^+}P_{t+1}}{\prod_{i\in\Omega^+}P_t}\Big) =$$

$$= log(e) + log(\prod_{i\in\Omega^+}P_{t+1}) - log(\prod_{i\in\Omega^+}P_t) = 1 + \sum_{i\in\Omega^+}[log(P_{t+1}) - log(P_t)] =$$

$$= 1 + \sum_{i\in\Omega^+}[log(\check{h}_{t+1}(x_i)) - log(\Theta_{t+1}) - log(\check{h}_t(x_i)) + log(\Theta_t)] =$$

$$= 1 + \underbrace{\sum_{i\in\Omega^+}[log(\check{h}_{t+1}(x_i)) - log(\check{h}_t(x_i))]}_{(A)} + \underbrace{\|\Omega^+\|[log(\Theta_t) - log(\Theta_{t+1})]}_{(B)} = (*)$$

$$(A) = \sum_{i\in\Omega^+}\gamma h_{t+1}(x_i) - \gamma h_t(x_i) = \gamma\Big[\underbrace{\sum_{i\in\Omega^+}h_{t+1}(x_i)}_{\mathcal{E}_{t+1}} - \underbrace{\sum_{i\in\Omega^+}h_t(x_i)}_{\mathcal{E}_t}\Big] =$$

$$= \gamma\sum_{z\in\Omega_u}h_{t+1}(z) - h_t(z) \quad with\ \Omega_u = \Omega_{v_1}\cup\Omega_{v_2}$$

$$\mathcal{E}_{t+1} = \mathcal{E}_t - h_t(x_{v_1}) - h_t(x_{v_2}) + h_{t+1}(x_{v_1}) + h_{t+1}(x_{v_2})$$

$$(B) = \Theta_{t+1} = \Theta_t - \check{h}_t(x_{v_1}) - \check{h}_t(x_{v_2}) + \check{h}_{t+1}(x_{v_1}) + \check{h}_{t+1}(x_{v_1})$$

$$(*) = 1 + \gamma\sum_{z\in\Omega_u}\Big[h_{t+1}(z) - h_t(z)\Big] + \|\Omega^+\|log\Big(\frac{\Theta_t}{\Theta_{t+1}}\Big)$$

This way of seeing the formula enabled us to speed up the computation. On the first step we need to compute (A) and (B) for every possible pair of nodes in the second tree layer and save them; in this way, we can reuse them, computing only the ones actually changing in one step (that is to say the values related to the couples generated by the node resulted from the merge).
It's then easy to get the new node's g and α with the aid of appropriate libraries.

4 Interpreting and evaluating a CNN

4.1 CNN interpretation

Once obtained the decision tree, given an image I_i we can find a parse tree that tells us which is the decision process that led the CNN to the prediction y_i.
To point out this parse tree we start from the root and at each step we choose the best child $\hat{v}$ that maximizes the compatibility with I_i's rationale, according to the following formula: $\hat{v} = argmax_{v \in \Omega_u} cosine(g_i, w_v)$.
Thus, we will get to the leaf with the decision mode most compatible with the input image.
However, the second tree layer nodes are what we are more interested in, because they describe more general decision modes shared by several images and based on those we can tell in average what object part is the most relevant for that category.
We can do this by defining ρ and $\hat{\rho}$. The first one evaluates the contribution of different filters and can be computed, for each node, as $w_v \circ x_i$; the second one evaluates, instead, the contribution of different object parts and is expressed as $A\rho_i$, where A is a matrix that assigns each filter with a specific object part.

4.2 CNN evaluation

Further, the prediction can be evaluated under various aspects by making use of some evaluation metrics. We took into consideration three metrics:

1. Object part contribution: the following table shows the percentage contributions of the filters related to each object part to the final CNN prediction, per tree layer.

contribution	head	torso	legs	tail	average
2nd layer	0.5585	0.4693	0.1732	0.1506	0.3379
3th layer	0.5060	0.5449	0.1818	0.1541	0.3467
4th layer	0.6350	0.4208	0.1844	0.1517	0.3480
5th layer	0.6445	0.4172	0.1798	0.1556	0.3493
6th layer	0.6780	0.3954	0.1881	0.1514	0.3532
7th layer	0.7304	0.3921	0.1881	0.1603	0.3677
leaves	0.5523	0.4740	0.1846	0.1542	0.3413

2. Error in object-part contributions: by de-activating one object-part per time and comparing the resulting prediction with the actual one, tells us how much error in the final prediction is given by that object-part contribution.

error	head	torso	legs	tail	average
2nd layer	0.5585	0.4693	0.1732	0.1506	0.3379
3th layer	0.5060	0.5449	0.1818	0.1541	0.3467
4th layer	0.6350	0.4208	0.1844	0.1517	0.3480
5th layer	0.6445	0.4172	0.1798	0.1556	0.3493
6th layer	0.6780	0.3954	0.1881	0.1514	0.3532
7th layer	0.7304	0.3921	0.1881	0.1603	0.3677
leaves	0.5523	0.4740	0.1846	0.1542	0.3413

3. Intersection over Union (aka Jaccard similarity coefficient): this metric measures the accuracy in the object detection, making a comparison between the ground-truth bounding boxes and the predictions; its name comes from the fact that it is a ratio between the area in which the boxes overlap and the union of the two areas.

Net	2nd	3rd	4th	5th	6th	7th	leaves
Vgg16	0.2575	0.3023	0.3099	0.2535	0.1900	0.1061	0.9999

4. Average classification accuracy (top table) and average prediction error w.r.t the CNN (bottom table) based on nodes in all tree layers and leaf nodes. The classification accuracy and the prediction error reflect the CNN knowledge not encoded by the decision tree.

CNN	2nd	3rd	4th	5th	6th	7th	leaves
88.5%	66.5%	66.0%	66.0%	56.9%	52.5%	50.0%	63.0%

CNN	2nd	3rd	4th	5th	6th	7th	leaves
Vgg16	27.8%	32.0%	31.7%	42.4%	51.1%	61.8%	39.2%

These metrics give us information about how accurate the predictions are and how much information we lose in the decision process.

All the results above show the same trend as the ones in the paper; however, our results are less accurate because of the size of our dataset, which is much smaller than the one used in the original paper [2]. The following (Fig.3) are some of the pictures of birds used for the evaluation, with recognized parts highlighted.

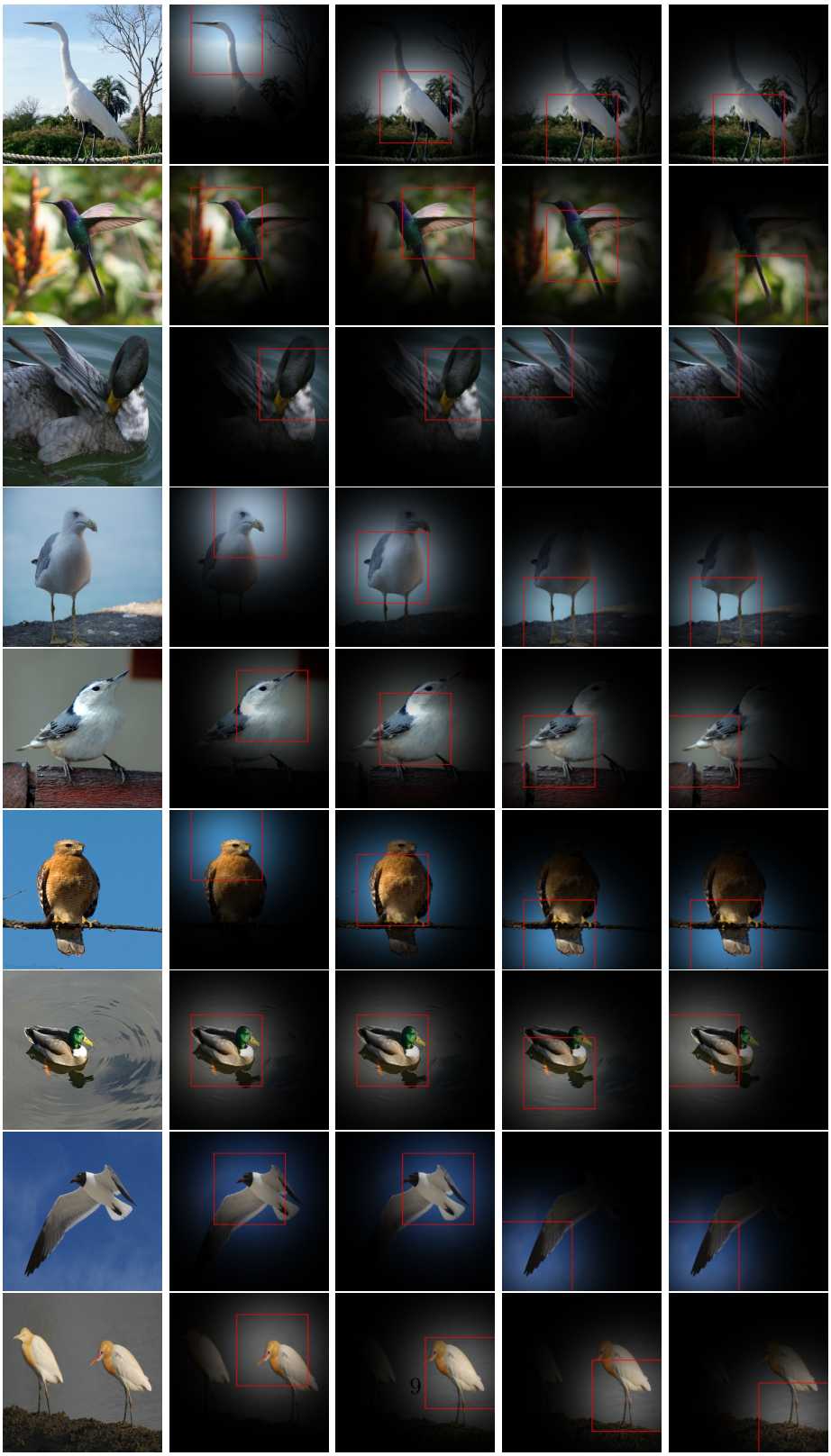


Figure 3: The original images (first column on the left), and the receptive fields computed from the max value of the averaged feature maps, grouped by object parts (*head, torso, legs, tail*).

References

Articles

[1] Quanshi Zhang et al. “Interpretable CNNs for Object Classification”. In: (2020). arXiv: 1901.02413 [cs.LG].

[2] Quanshi Zhang et al. “Interpreting CNNs via Decision Trees”. In: (2019). arXiv: 1802.00121 [cs.CV].

[3] Xianjie Chen et al. “Detect What You Can: Detecting and Representing Objects using Holistic Models and Body Parts”. In: (2014). arXiv: 1406.2031 [cs.CV].

[4] C. Wah et al. “The Caltech-UCSD Birds-200-2011 Dataset”. In: CNS-TR-2011-001 (2011).

[5] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2015). arXiv: 1409.1556 [cs.CV].

[6] Bolei Zhou et al. “Object Detectors Emerge in Deep Scene CNNs”. In: (2015). arXiv: 1412.6856 [cs.CV].