Class disjointness constraints as specific objective functions in neural network classifiers

François Scharffe
Axon Research
fscharffe@axon.com

Abstract

Increasing performance of deep learning techniques on computer vision tasks like object detection has led to systems able to detect a large number of classes of objects. Most deep learning models use simple unstructured labels and assume that any domain knowledge will be learned from the data. However when the domain is complex and the data limited, it may be useful to use domain knowledge encoded in an ontology to guide the learning process. In this paper, we conduct experiments to introduce constraints into the training process of a neural network. We show that explicitly modeling a disjointness axiom between a set of classes as a specific objective function leads to reducing violations for this constraint, while also reducing the overall classification error. This opens a way to import domain knowledge modeled in an ontology into a deep learning process.

1 Introduction

As deep learning research progresses, efforts are made to add structure to the set of labels used as annotation data. This seems a logical step as the number of labels in object classification tasks becomes larger. For example the ImageNet challenge (Russakovsky et al., 2015) in its 2017 edition invites system to localize objects in 1000 categories, detect 200 types of objects in images, and detect 20 types of objects in videos.[1] However little about the organization of these classes is provided. Models are expected to implicitly acquire the knowledge that, for example, a person has two legs and a car four wheels from the training examples provided as part of the dataset. What works on a controlled dataset in an academic challenge task might however be less practical in a real world example where annotated data can be scarce, and the modeled domain vast and complex.

The work presented in this paper is part of a larger project for automating the understanding of body worn camera footage in a law enforcement setting. The goal is to provide annotations that will facilitate the manual footage reviewing process that is mandatory across many agencies. We train deep learning models to categorize videos by their type, for example a Traffic Stop, or a Pedestrian Stop; and to give annotations indicating the type of event ongoing at a specific time segment, for example an Officer Driving, or a Handcuffing. The illustration below shows an example timeline our tool generates on an unseen video.

In parallel to developing models able to capture the specificities of this visual domain, we develop an ontology modeling this domain. Type of detections are represented as classes modeled in RDFS and OWL (Dean et al., 2004). There are three higher level classes for topics, segments, and objects that represent different levels of granularity: topics apply to a whole video, segments apply to a scene in a video that may vary from a few seconds to many minutes, and objects can be detected at any point in the video.

Because of legal and privacy concerns, and because of the cost of annotation, our training data set is limited to a few hundred videos. This results in the need for models able to learn from small training sets. In particular, we would like to be able to reflect the constraints modeled in the ontology into the neural network training process, so that this additional knowledge helps the neural network converge faster and compensates for the small training set. We indeed assume that if a much larger training set would be available, the constraints implicitly modeled in the annotations could be learned by the neural network.

[1] `http://image-net.org/challenges/LSVRC/2017/` retrieved on 07/05/2017

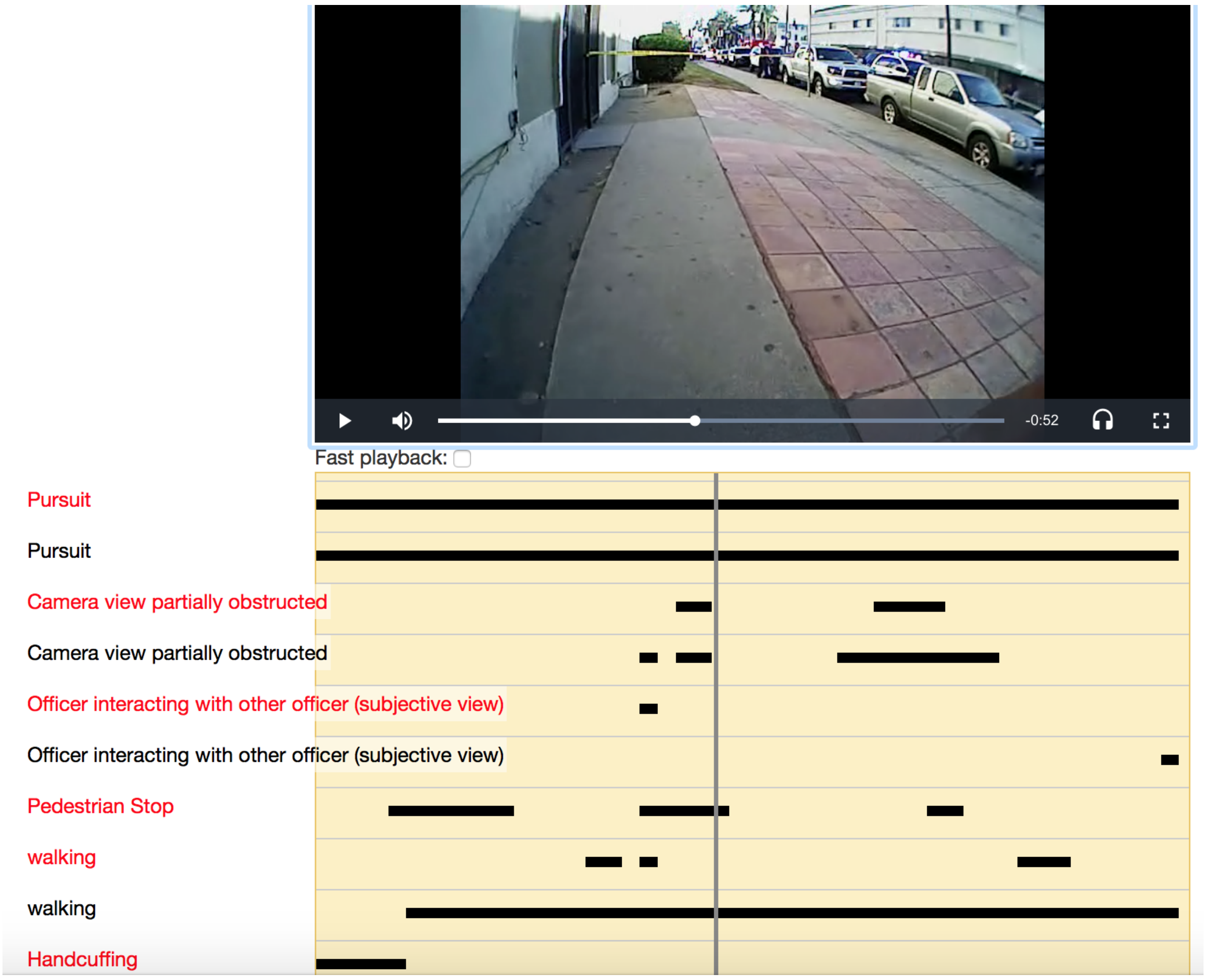

-0:52
Fast playback:
Pursuit
Pursuit
Camera view partially obstructed
Camera view partially obstructed
Officer interacting with other officer (subjective view)
Officer interacting with other officer (subjective view)
Pedestrian Stop
walking
walking
Handcuffing

Figure 1: Illustration of the problem tackled in this paper with a timeline over a body worn camera video. Ground truth annotations are represented in black. Machine detected segments are represented in red. The neural network detect at the same time *Pedestrian Stop* and *Pursuit* which are disjoint activities.

This paper focus on modeling and then learning a class disjointness constraint. After giving class disjointness modeling examples related to our domain in Section 2 we formulate the constraint as a loss function applied on the disjoint classes. Then in Section 3 we conduct experiments on a synthetic dataset showing that this method effectively leads to an improvement in enforcing the constraint.

2 From an ontology axiom to a loss function

A long term goal of this work is to design a system where changes in the ontology would result in changes in the neural network training process. Especially we are interested in studying how modeling constraints as ontology axioms can result in adding specific losses on the neural network output classes corresponding to the ontology classes on which these axioms apply.

OWL2 (Motik et al., 2009) provides three constructs for expressing class disjointness. The main one `owl:disjointWith` is a class axiom specifying disjointness between this class and another. Multiple owl:disjointWith statements can be defined within a class description. In our use case, a set of topics are all disjoint for a given annotated video. OWL2 introduces a convenient way to model this using the owl:DisjointUnion construct allowing to specify the disjointness outside of the class definition. The statement below expresses that the class Topic has three disjoint subclasses Pedestrian Stop, Traffic Stop and Car Pursuit.[2]

```
owl:DisjointUnion(leo:Topic
                  leo:PedestrianStop
                  leo:TrafficStop
                  leo:CarPursuit)
```

Similarly we find it more convenient to express that two classes are disjoint in a separate axiom than in the classes definitions. The OWL2 construct `owl:DisjointClasses` allows to express that a set of classes is pairwise disjoint. The statement below expresses that an officer cannot be at the same time driving a car and be a the passenger of that car.

```
owl:DisjointClasses(leo:OfficerDriving
                    leo:OfficerPassenger)
```

With a large enough training data set, the constraints should be learnt by the model from the data. However, the dataset would need to cover all possible conditions that would happen when seeing new videos, and the space of possibilities is large. For example light conditions, vegetation, urban setting, police officers vehicles and uniforms can vary widely between police departments. There are multiple cases in other domains where the amount of data available for training may be limited, while the domain is quite complex. This can happen for reasons of data privacy, for example when dealing with personal or sensitive data, or when training happens on a user mobile device. Explicitly using ontological knowledge in the neural network training process will help the model avoiding to break these constraints on new data.

We propose here to study the effect of a specific objective function applied to the neural network outputs corresponding to the classes on the ontology for which a disjointness axiom was defined. This function should maximize loss when the constraint is broken, allowing only one class to be positively output. More specifically, in this paper we treat the case where one of the disjoint classes has to be selected by the network. A common objective function to achieve this is to use categorical cross-entropy (De Boer et al., 2005). This objective is usually used in combination with a Softmax activation. However, we do not want to modify the neural network architecture when introducing constraints, but only the objective function. We thus apply Softmax on the predictions before passing them to the objective function.

Equipped with this loss function for class disjointness we will next run experiments to study its influence on model training.

3 Experiments

Our experiments consist in testing the effect of the custom loss on forcing the neural network to learn the constraint. As motivated in the introduction of this paper, the sensitive character of the data we

[2]The leo namespace stands for Law Enforcement Ontology, an ongoing effort to model police officers interactions.

$$\begin{array}{r}\text{Disjoint classes}\begin{Bmatrix}0.32543 & 0\\ 0.93456 & 1\\ 0.62334 & 0\end{Bmatrix}\\ \text{Other classes}\begin{Bmatrix}0.43655 & 0\\ 0.92824 & 1\\ 0.16593 & 0\\ 0.61232 & 1\end{Bmatrix}\end{array} \tag{1}$$

Figure 2: Example generated dataset. Input on the left columns are randomly generated. Output for disjoint classes is 1 for the highest value, 0 for the others. Output for the other classes is 1 if the input is greater than a threshold here set to 0.5, 0 otherwise.

are working with does not allow to share it for reproducibility. We thus present experiments on a synthetic dataset presented below. All the experiments and dataset generation code is open source and available at: `https://github.com/OpenAxon/constrained-nn`.

3.1 Dataset

In order to be able to control parameters of the dataset we work with and to speed up training experiments, we generate synthetic data using the following procedure. A number of pseudo-random numbers is generated for the input. Outputs are separated into two sets: one set that represents classes under the disjointness axiom and for which we would like the disjointness constraint to apply, and another set which represent the other classes. We set the number of outputs to be equal to the number of inputs and generate output values according to this: we take the argmax of the input set that represents disjoint classes, and assign the corresponding output to one. Every other output in the disjoint set is set to zero. The rest of the outputs are set to one if their corresponding input is greater than a given threshold, and to zero otherwise. Figure 1 illustrates inputs and corresponding output values.

In the context of these experiments we use a dataset with a hundred thousand data points, having 50 generic classes and 5 disjoint classes. We set the threshold parameter for generic classes to 0.5. We selected the number of generic classes and disjoint classes to be about the same size of the number of classes and disjoint classes we have in our law enforcement use case.

3.2 Model architecture

Our hypothesis is that adding the ontology constraint in the network as a specific loss function will help the network enforcing this constraint. We verify this by training and testing a model on the dataset presented above. The model architecture is detailed in Figure 3 below. It consists in a fully connected network with three layers having non-linear activations, and an output layer with a sigmoid activation. Each intermediate layers has 200 neurons. We use an Adam optimizer with a learning rate of 0.001.

We use this architecture to define two models by only changing the objective function: for the baseline we use binary cross-entropy on the whole set of output classes. For the constrained model we use a combination of binary cross-entropy defined on the non constrained classes, and the softmax categorical cross-entropy loss defined in Section 2 for the disjoint classes. A parameter controls the weight of the constraint loss with regards to the generic loss. We present our results below.

3.3 Results

Our goal is to compare how the model performs on learning the constraint for this dataset. In order to report this we define two metrics that measure how well the network can predict constrained and non-constrained classes correctly. Given P the set of predictions, G the ground truth and C the set of constrained classes, metrics are defined as follows:

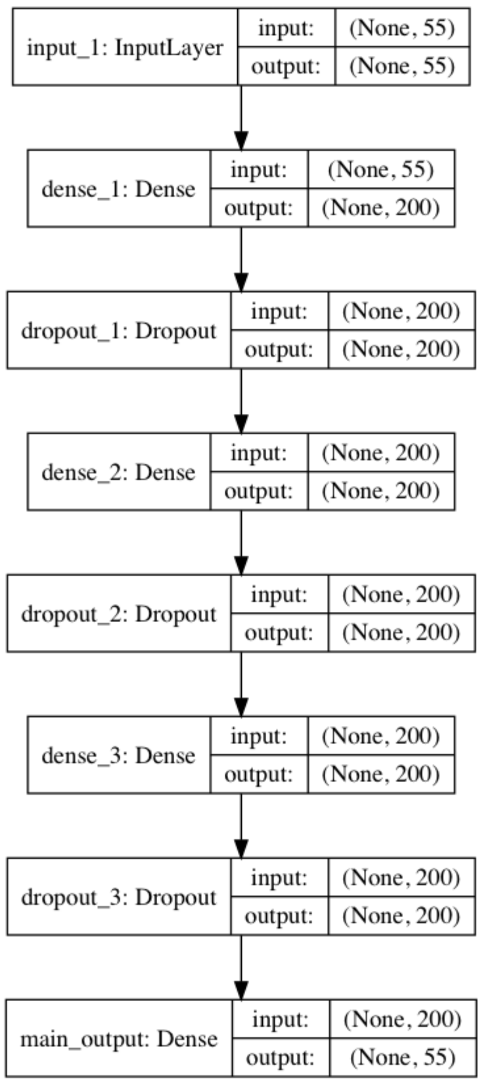


Figure 3: Model architecture. The None values are placeholders for the batch size.

- Constrained classes metrics: categorical classification accuracy

$$M_i = \frac{\sum_{c \in C}([P_i(c)] = G_i(c))}{|C|} \tag{2}$$

Where $[P_i(c)]$ represents the nearest integer to the prediction.

- Unconstrained classes metric: binary classification accuracy

$$M_i = (argmax_C(P_i) = argmax_C(G_i)) \tag{3}$$

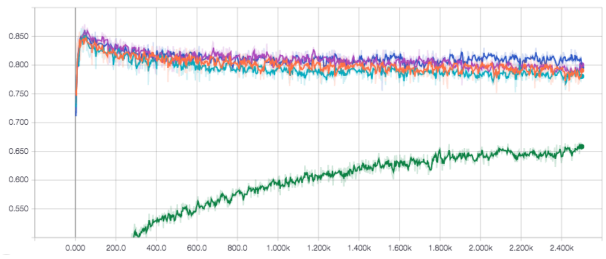
for each batch b of the training set.

We perform multiple training runs using a batch size of 32 on three dataset with varying sizes. Reported numbers below are averaged for each batch. For each dataset we perform:

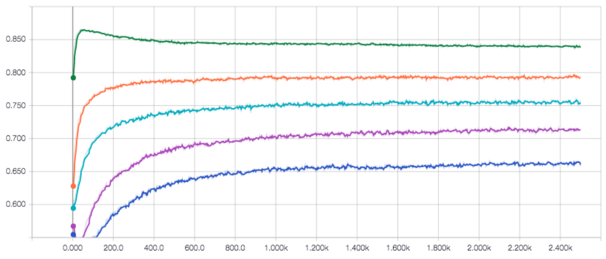
- one run for the baseline using a single loss.
- four runs for the constrained model using weights of 1, 2, 4 and 8 for on the constraint loss.

Three datasets are generated with size 5000, 10000 and 50000. Other dataset parameters are given in Section 3.1. Figures 4-6 below report results for the various configurations on each dataset.

As expected, overall accuracy increases as the datasets size increase. The weight on the constraint loss does not have a significant effect on the accuracy of the constrained outputs. However, as the weight increases, the accuracy on the unconstrained output decreases significantly. These observations are valid for the three datasets. As expected, the unconstrained model has difficulty learning

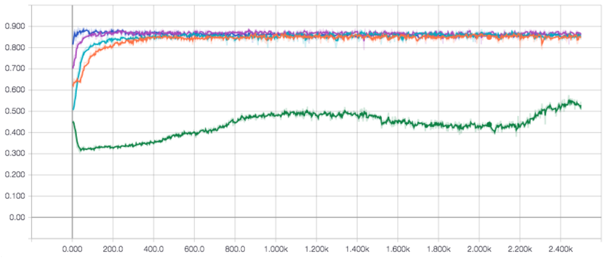


(a) Categorical classification validation accuracy on constrained outputs of the 5k dataset

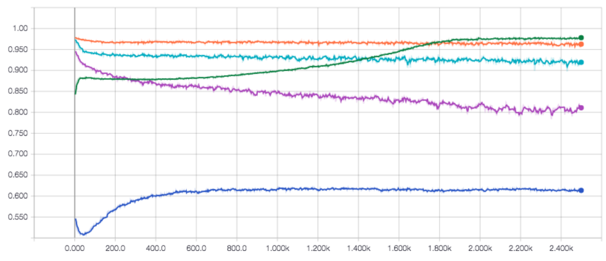


(b) Binary classification validation accuracy on unconstrained outputs of the 5k dataset

Figure 4: Constraints on 5k dataset using various weights for the constraint loss.
Legend: green: no constraint, orange: w=1, cyan: w=2, magenta: w=4, blue: w=8

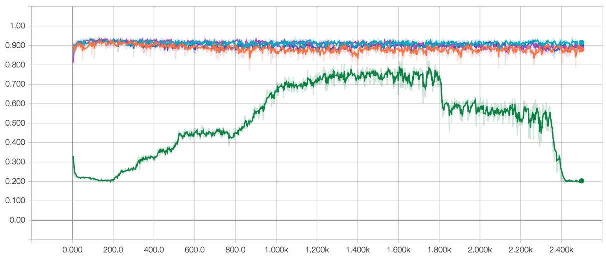


(a) Categorical classification validation accuracy on constrained outputs of the 10k dataset

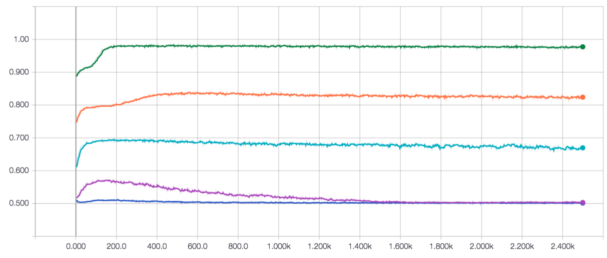


(b) Binary classification validation accuracy on unconstrained outputs of the 10k dataset

Figure 5: Constraints on 10k dataset using various weights for the constraint loss
Legend: green: no constraint, orange: w=1, cyan: w=2, magenta: w=4, blue: w=8



(a) Categorical classification validation accuracy on constrained outputs of the 50k dataset



(b) Binary classification validation accuracy on unconstrained outputs of the 50k dataset

Figure 6: Constraints on 50k dataset using various weights for the constraint loss
Legend: orange: no constraint, cyan: w=1, magenta: w=2, blue: w=4, green: w=8

the constraint on smaller datasets, but the gap between the models trained with a constrained loss and the model without the constrained loss decreases as the dataset size increases. This confirms the intuition than given enough data, explicit constraints might not be necessary. A constrained loss with a weight of 1 provides a very good tradeoff in loss of unconstrained accuracy versus improvement of the constrained outputs accuracy.

4 Related Work

Computer vision challenges provide unstructured or minimally structured labels, usually a list or a shallow taxonomy. Even fewer methods make use of the structure when it is provided.

The fast and accurate object detection system YOLO9000 (Redmon and Farhadi, 2016) makes use of the WordNet lexical database to build a hierarchy of concepts from the flat list of ImageNet labels. Predictions are then conditional probabilities of a concept class given its parent class. Prediction computation is performed by computing the softmax of all siblings at every level of the concept hierarchy. The predicted class is obtained by following the highest confidence path from the top of the concept hierarchy, until a threshold is reached. This approach is complementing our work and will likely be used in the system we are building.

In (Pathak et al., 2015) constraints are applied using a specific loss function in weakly supervised image segmentations. Only images labels are provided and constraints help the network identify images segments according to these labels. Constraints are defined specifying the presence or not of a class in the background or in the foreground according to the presence of a label. Another constraint specifies the relative number of pixels a class label is expected to occupy. The domain knowledge encoded in these constraints is specific to the datasets. It is possible to define the relative size of a class because the datasets contain iconic images where labels occupy a central position and a significant proportion of the image. The domain knowledge encoded in the constraints is thus dataset specific, rather than ontological. Our goal is to be able to transfer ontological knowledge of a domain that could apply to various datasets.

5 Conclusion

This work introduced constraints modeled in an ontology as part of a neural network training process. With these initial experiments we studied the usage of one class disjointness axiom over a set of classes. Adding the ontology axiom as a specific objective on the outputs corresponding to the disjoint classes provide a significant improvement: the classification accuracy of the network for the constraint is improved by 15 to 35 points depending on the dataset size. It is useful to note that adding the constrained loss does not result in a strict restriction, and the network may still in a few cases output classifications violating the constraint.

We plan to apply the class disjointness axioms in training models on actual video data. In order to be able to report on findings on these data, we are working toward building and releasing an open dataset of law enforcement videos, that we hope will help foster new research.

In future work we are interested to study the effect of multiple disjointness constraints. We are also studying how to model other ontology axioms as constraints. In a longer term, we are interesting in studying how ontology axioms can influence a neural network training process, but also how from an unconstrained neural network trained on a large dataset we could derive and materialize ontology axioms.

References

De Boer, P.-T., D. P. Kroese, S. Mannor, and R. Y. Rubinstein (2005). A tutorial on the cross-entropy method. *Annals of operations research 134*(1), 19–67.

Dean, M., G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein (2004). OWL web ontology language reference. *W3C Recommendation February 10*.

Motik, B., P. F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and others (2009). OWL 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation 27*(65), 159.

Pathak, D., P. Krahenbuhl, and T. Darrell (2015). Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1796–1804.

Redmon, J. and A. Farhadi (2016, December). YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242 [cs]*. arXiv: 1612.08242.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015, December). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision 115*(3), 211–252.