CNN-based detection of geometric shapes in images, and question answering

Patrick Hammer

December 2019

1 Introduction

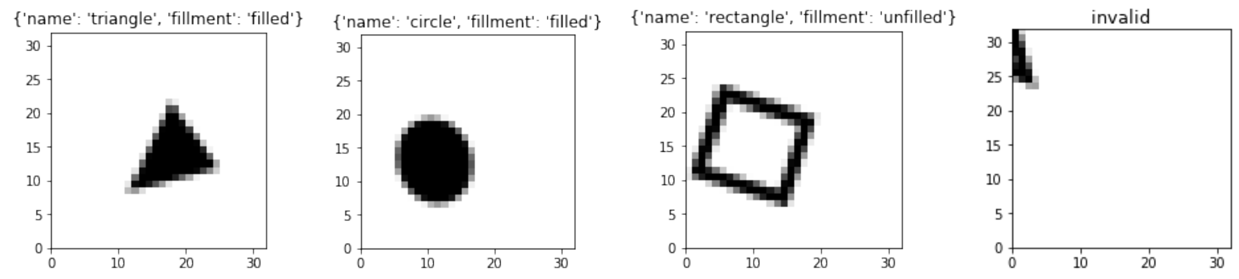
Convolutional Neural Networks for image classification and object detection are commonly trained to detect certain high-level object classes. Mostly due to their innate ability to exploit translation invariance through Convolution, and solutions to vanishing gradient issues which allowed for the use of much more layers, they have become the most successful existing technology to identify objects of interest in images. However, they do not generalize to other domains where different objects are of interest than they were trained to detect. While solving this to unsupervised learning related issue, is outside of the scope of this project, this project concentrates on a small sub-problem, to detect geometric shapes, and their properties in black&white images. Geometric shapes are not domain-specific, can also be composed into more complex shapes, and their spatial configurations among each other can be reasoned on with mechanisms such as proposed in [4]. The core of this project is the construction and training of an image classifier. For the bigger picture, a simple sliding-window detector utilizing the image classifier is then introduced. This detector translates its output into a format viable for reasoning. Basic experiments for Reasoning-Based Visual Question Answering are shown in the results section, together with the classifier evaluation and detection outcomes.
The idea of a general-purpose perception system is not new: for instance, Harry Foundalis has worked on a system system called “Phaeaco”, for his PhD thesis at the Center for Research on Concepts and Cognition. His system solves Bongard problems through building and comparing semantic representations using a sophisticated resource allocation strategy called “Parallel Terraced Scan” [3], guided by Simulated Annealing. Back then, Deep Learning was not a thing, so he did the low-level processing using hand-coded computer vision code, including code for preprocessing, extraction of line segments etc.. More popularly, and since then, Xavier Glorot and Yoshua Bengio, have pushed the capabilities of various techniques in Deep Learning, and have experimented with geometric shape detection in images [2], to explore issues when training larger networks. This project takes an approach which is closer to Bengio’s, using recent Artificial Neural Network technology to achieve what was done by handcoded computer vision algorithms in Foundalis system, to detect different types of shapes, their location, and their properties. Different to Foundalis system, Non-Axiomatic Reasoning System (NARS) is used as “cognitive” component, allowing for Reasoning-Based Question Answering about user-drawn shapes based on the representations built on top of the detector outputs. As we will see, this allows for answering of questions without knowing and training for specific question types beforehand.

2 Approach

Model The classifier is the heart of the system, without a proper classifier, the sliding-window detector cannot be built. This is where recent Deep Learning technology is exploited. The classifier is a Convolutional Neural Network, as translation invariance drastically reduces the amount of training examples the classifier needs to be trained with. Additionally, Max-pooling helps to break down dimensionality, and introduces a bit of scale-invariance as well. As input of the network, a 2D one-channel image of size $(32, 32)$ is given, and the output is the shape type and value of the filled property (filled, unfilled). The overall structure is as follows: the network consists of 3 Convolution layers, each followed by Max-Pooling. Each has stride 1 and a filters amount (amount of hidden neurons in the layer) of 512, and kernel size $(3, 3)$ except of the first one which has $(5, 5)$. Then there are 3 dense layers with 256 hidden neurons each, except the last one which is just the output with 7 classes which summarizes shape type and filled property into a one-hot encoded vector of size 7. The interpretation of each component is: $(rectangle, filled)$, $(triangle, filled)$, $(circle, filled)$, $(rectangle, unfilled)$, $(triangle, unfilled)$, $(circle, unfilled)$, $invalid$, where the invalid class is for images with shapes not completely within the image. This class was introduced to make it more viable for sliding window detection, to get rid of the issue of mis-detections caused by shapes being partly of the image. Also, ReLU was chosen as activation function consistently to avoid vanishing gradient issues, except for the output which uses the Softmax activation function, a proper choice for the one-hot encoded output.

As reasoner for question-answering, OpenNARS, an implementation of the Non-Axiomatic Reasoning System was chosen. Other than traditional reasoners, this system allows to learn regularities in the input which can be incorporated in question-answering. Inductive logic programming systems as described in [7] can do this to some degree too, but they are not designed to learn in an incremental manner. A property which isn't relevant for static image analysis, but is relevant for real time applications where re-training isn't feasible after new examples arrive, and where questions can have temporal aspects.

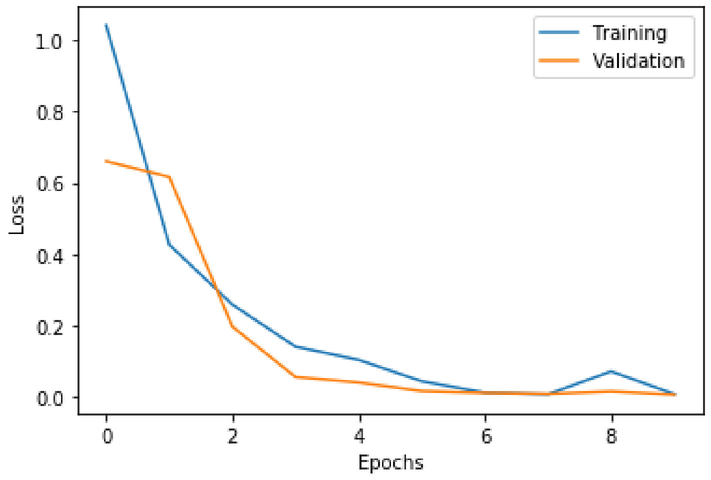
Dataset For training, the Shapeset dataset was chosen (see [5]). Shapeset contains shapes with different properties, of which the filled-property was chosen for the CNN to predict together with their shape type, as visible in the following Shapeset examples:



It is however an infinite-sized dataset, as it comes as a Python script that allows to generate as many shapes as one wants, with the ability to parametrize them in various ways. Hence, for the image classifier, a subset of 30K images was chosen, with output image size of $(32, 32)$, and a padding of 9, together with the label information ShapeType (categorical attribute) and Filled Property (also categorical, Yes/No). Some other parameters such as a padding of 9 is involved too, see the Notebook for the particular other values. The notebook allows to reproduce the exact 30K image subset of Shapeset which was utilized. This has been more difficult than desired, as the original script was written in an ancient Python version not even compatible with recent Python 2.7 versions. It was made to work by downloading an old Python version and selecting dependencies carefully, re-compiling them. One of its dependencies was an old version of PyGame, which itself used some Ubuntu dependencies not commonly available anymore, neither in Ubuntu nor in other distributions. To make it run, a VM with an old Ubuntu version was necessary. It was used to verify whether the parameters did what they are supposed to. Given these verifications, the code was ported over to the scientific Python stack, using only the commonly available PyCairo drawing library as extra dependency (which is available on all platforms!), see the Notebook for the code. The dataset was split into a distinct training and test set as usual, with 25 percent reserved for testing. Also, the data was transformed to zero mean and unit variance, by subtracting the mean and dividing by the standard deviation, for each attribute (pixel value in this case).

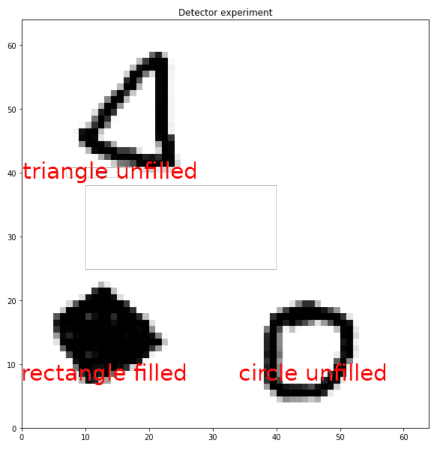
3 Results

Classification Training of the Convolutional Neural Network-based classifier was done using Mini-batch gradient descent using the categorical crossentropy loss, a reasonable choice given the one-hot encoded output layer with softmax activation. Vanilla SGD was used, so no momentum, and also no Nesterov momentum. As mini-batch size, 64 was chosen, and learning rate of 0.01 was applied. Training was performed using 10 epochs. This, in total, took 153 minutes on a 4 (8 threads) core i7-2670QM CPU, utilizing the 8 threads for training for maximum speed (GPU training would have been faster of course, but CPU training was feasible in this case). The training process, after 10 epochs, ended with a training loss of 0.0071, training accuracy 0.9998, validation loss 0.0056, and validation accuracy 0.9996. The following plot shows the loss over the epochs for each training epoch, for both the training and test set:



As we can see the training basically converged nicely with 10 epochs of training, with only marginal improvements after the 5th epoch. Also not only the training and validation loss converged to reasonable values, but also no over-fitting took place, hence early stopping wasn't necessary in this case. For illustration, an example classifier output from the empirical testing which followed after training:

Detection To initiate extending the solution towards the grand goal of a general-purpose perception system, the classifier was utilized to support Detection, using a simple sliding window approach. Since Shapeset contains only pictures with single shapes, only empirical evaluation was possible here. For this purpose the Notebook was extended with the ability for the user to draw examples with multiple shapes in it, which is used as input to the detector. The size of this image is 64, giving 4 times as much space to draw shapes. Thanks to the translation invariance gained by using convolution, the lattice that defines the points the classifier gets applied is not needed to be very fine-grained, it is defined to have 4 points per dimension, giving 16 detection points in total. Of course, this also defines the spatial accuracy. To avoid multiple detections for the same objects, additionally only one label was reported for neighbouring detections of same label. More sophisticated related techniques like Non-maximum Suppression do exit, but were not necessary in this case.



Clearly the detector generated the right shape class and filled property for each of the drawn shapes in the image. This is due to the Classifier which converged well on the dataset of examples similar to the drawn ones.

Reasoning OpenNARS v3.0.3 (see [6] and [4]) was utilized as reasoner. For this purpose, the detector output was converted into Narsese (NARS's I/O and internal language), resulting in Inheritance statements about instances. To define that a detection instance is of a certain shape type, $(instanceID \to ShapeType)$ is used, and a property $[filled]$ and $[unfilled]$ is introduced, leading to $(instanceID \to [filled])$ or $(instanceID \to [unfilled])$ dependent on the predicted value of the filled property. Additionally, spatial relations $R_{spatial}$ are introduced, where $R_{spatial}$ can be $topOf$ or $leftOf$, so the format is $((instanceID1 \times instanceID2) \to R_{spatial})$. The shown detection example leads to the reasoner inputs $(\{shape1\} \to rectangle)$. $(\{shape1\} \to [filled])$. $((\{shape1\} \times \{shape2\}) \to leftOf)$. $((\{shape1\} \times \{shape3\}) \to topOf)$. $(\{shape2\} \to circle)$. $(\{shape2\} \to [unfilled])$. $((\{shape2\} \times \{shape3\}) \to topOf)$. $(\{shape3\} \to triangle)$. $(\{shape3\} \to [unfilled])$. $((\{shape3\} \times \{shape2\}) \to leftOf)$.

based on which multiple questions can be asked, for instance "what is the filled rectangle left of?": $(((\&, [filled], rectangle) \times \{?1\}) \rightarrow leftOf)?$ which NARS answers correctly with $(((\&, [filled], rectangle) \times \{shape2\}) \rightarrow leftOf)$. (confidence omitted) Meaning the system has associated the description "filled rectangle" with Shape1, a form of perceptive instance binding. Also clearly it has identified Shape2 (the unfilled circle) to be left to Shape1. This is just one example of what the reasoner can do with these input representations. Alternative questions the solution supports, and which can be tried in the attached notebook for instance are: "are the rectangles filled?", $(rectangle \rightarrow [?filled])?$, "is there a filled rectangle left to a circle?", $(((\&, [filled], rectangle) \times circle) \rightarrow leftOf)$, queries the reasoner all answered correctly as desired for each of 30 example drawings with correct detections.

4 Conclusion

Using a Convolutional Neural Network with sufficient layer amount and Max-Pooling after the Convolutions allows for reasonable accuracies for shape classification, and allows for the additional prediction of shape attributes. The network converged well on the chosen subset of ShapeSet in the training phase, as shown by the accuracy on the test set. Also, this project has shown how a classifier can easily be extended to a detector, using a Sliding Window approach. Due to translation invariance gained by Convolution layers, the lattice for the sliding window does not need to be very precise as well, improving the computational effort over applying the network on all of the points of a finer lattice, compromising spatial accuracy of the detections though. Additionally, the translation to Narsese based on the semantic label output (shape type and fillment property) turned out to be very straightforward, and was extended with spatial relations. Also it was demonstrated how the extraction of semantic properties and spatial relations, can lead to a general-purpose visual question answering system, allowing different kinds of questions to be asked. Such as the shown queries, which are about the spatial arrangement of the shapes, their shape types and properties, and combinations thereof. While Shapeset does not contain ground truth for question-answering tasks, the system has shown reasoning competence by asking all questions correctly for 30 examples with correct detections.

Future work will include evaluation of the detector, and trying a YOLO model to overcome the drawbacks of the sliding window approach. In this case, the Shapeset generator needs to generate bounding box output for each example as well, as YOLO predicts both class and bounding box at the same time. Besides being more performant (no multiple sliding window positions or region proposal points in a region-proposal based approach), it would also give increased spatial accuracy for the detections than the sliding window approach. Also the reasoning part will be explored more in the future: multiple pictures could be shown in sequence, and the reasoner could be asked questions about both temporal and spatial aspects. This would further this project's intention to show how a quite simple architecture which combines Deep Learning methods with reasoning technology, can lead to a general visual Q/A system. A Q/A system which does not need to be trained on different combinations and relative arrangements of input stimuli, and can handle questions it was not particularly trained on. This is a distinguishing property to solutions such as [8] which, while borrowing symbolic knowledge representations, relies on Backpropagation completely rather than using proper Reasoning machinery on a higher level.

References

[1] Foundalis, H. (2006). PHAEACO: A cognitive architecture inspired by Bongard's problems, PhD thesis, https://www.foundalis.com/res/Foundalis_dissertation.pdf, last accessed: November 14, 2019.

[2] Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).

[3] Rehling, J., & Hofstadter, D. (1997, October). The parallel terraced scan: An optimization for an agent-oriented architecture. In 1997 IEEE International Conference on Intelligent Processing Systems (Cat. No. 97TH8335) (Vol. 1, pp. 900-904). IEEE.

[4] Wang, P. (2013). Non-axiomatic logic: A model of intelligent reasoning. World Scientific.

[5] Shapeset dataset: https://mila.quebec/datasets/Public/BabyAIShapesDatasets/, last accessed: November 14, 2019.

[6] Hammer, P., Lofthouse, T., & Wang, P. (2016, July). The OpenNARS implementation of the non-axiomatic reasoning system. In International conference on artificial general intelligence (pp. 160-170). Springer, Cham.

[7] Muggleton, S. (1991). Inductive logic programming. New generation computing, 8(4), 295-318.

[8] Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. arXiv preprint arXiv:1904.12584.