Joakim Jensen

Final project – CNNs on edge maps

Introduction and motivation

Convolutional neural networks are one of today's strongest and most popular methods for automating computer vision tasks. One of the most basic computer vision tasks is classification. Classification problem in computer vision is often the task of telling what kind of object is present in the image, without saying anything about where. There exist multiple kinds of classification. In binary classification only one of two classes can be predicted by our model. Multi classification takes on the task of classifying one class out of more than 2 possible classes. Multi label classification, on the other hand, classifies one or more classes out of two or more possible classes. For this report we will only deal in binary and multi class classification.

Classification of an image is achieved by passing the image through a CNN. The different layers of the CNN will process the image and output a vector. This vector will have the length of the same number of possible classes. For our classification tasks, we are only trying to predict one class for each image. The CNN will therefore try to output as close as possible to a one hot encoded vector whenever the network is certain of what class object is in the image. The index of this vector which has the highest value is the class the model is trying to predict. Often, we set up a fixed list of classes where each index holds a possible class. Before a model can start classifying images accurately the network will have to be trained on labeled images. This means we show the network a ton of images which have been classified by humans. By letting the network train on these already classified images the network can learn how to classify images. Training happens by letting the network make its own prediction of an image, thereafter, looking at what the answer was, calculating how wrong the network predicted, backpropagating the error backwards in the network and at last adjusting its weights based on the backpropagated error. This process is repeated many times, each time the network improving a bit on its wrongness. Getting a network which can accurately classify images, is all about finding the right weights for the network, which can lower its wrongness. When wrongness goes down the model will be more correct and start classifying images more and more accurate.

Edge maps are maps that hold the edges of an image. There exist a ton of methods for finding the edges and they all have their pros and cons. Used methods of this report is very shortly explained later.

In the course Image Analysis, we have learned how edge maps can be created by use of different methods. We also learned edge maps can be used to sharpen edges of an image and how the same techniques used to create edge maps are also used to create handcrafted feature extractors, like the use of Sobel operator in HoG (histogram of oriented gradients).

Creating an edge map, we will only see the edges of the image. This often being the edges of the different objects in the image. By only showing the edges and outlines of objects and removing color shouldn't that technically be an easier to understand image to the computer? In this project, I've built the CNN AlexNet to see how the model performs in image classification when feeding the edge maps of different methods in comparison to a normal RGB colored images.

Hypothesis

Applying different edge operators to an image and then feeding that edge map to AlexNet should give the network an easier chance understanding the image leading to either or both better performance in accuracy and training time. However, the efficiency of feeding edge maps should reduce when class objects have similar shape or are more dependent on color to make the right classification.

NI - Natural images Dataset

Natural images dataset is multiclass dataset of 8 classes with 6899 images. The classes of this dataset are airplane, car, cat, dog, flower, fruit, motorbike and person. Images come in different sizes and without a train-, valid-, test-split. Images were therefore resized to AlexNet preferred sized of 224x224 and split into 70% train, 20% valid and 10% test. The dataset contains objects of very varying shapes per class object. The dataset is available at: https://www.kaggle.com/datasets/prasunroy/natural-images

DD – Dandelion vs Daisy Dataset

Dandelion vs Daisy is a binary classification dataset with 1821 images. The classes of the dataset are Dandelion and Daisy. Images are of size 512x512 and came in a train-, valid-, test-split of 70% train, 20% valid and 10% test. Images were also resized to 224x224. The images are of flowers, both object classes mainly having a round shape, the distinguishing feature is the color. Dandelion most often carrying yellow and daisy carrying a white color. The dataset is available at: https://public.roboflow.com/classification/flowers_classification/2

Method

Creating new dataset variations, I used several edge detection methods which was introduced in the Image analysis course. Some of the edge detectors like the Roberts, Sobel and Laplacian uses the derivatives of the image to find the magnitude and direction of edges. Laplacian using the second derivative though, while Roberts and Sobel use the first. These algorithms find the gradients of image by convolving the image with a kernel. The kernels are different for Roberts, Sobel, and Laplacian. The output image is an edge map. Canny edge detection algorithm is an algorithm used for finding edges. The algorithm first smoothens the image. Then applies the Sobel operator to find magnitude and orientation. Non-max suppression is then applied to each pixel. Non-max suppression works by looking at parallel neighbors in N_8 neighborhood. Which parallel neighbors we look at is decided based on the direction. If the pixel in focus has a higher magnitude than both neighbors, it gets to keep its magnitude. Otherwise, it's set to 0. Thresholding is then applied to all pixel values. The upper and lower bounds are decidable. For this project I use lower bound of 100 and upper bound 200. If the magnitude is above the higher bound set pixel value to 255. If between bounds, mark as possible edge and if below the lower bound set pixel value to 0. Hysteresis is then applied by looking at each pixel marked as a possible pixel. If at least one neighbor in the N_8 neighborhood is considered an edge (it has a value of 255), this pixel should also be considered an edge, its value is set 255. Morphological gradient is the dilation of an image subtracted by the eroded image. The structuring element used for the morphological operations is a 5x5 square. Despite many of the algorithms recommending running a low pass filter over the image to reduce noise before using the edge operators, I've chosen not to that. I've chosen not to that because the network will deal with noise regardless of smoothening or not. In some cases, smoothening

might accidently erase edges of the object from the image. It should also be interesting to see if the more noise sensitive edge detectors perform worse because of the missing smoothening.

Results

DD dataset

DD dataset	Test accuracy	Best epoch
No edge detection	**0.8901**	16
Canny	0.6483	19
Laplacian	0.8077	23
Roberts	0.7473	27
Sobel	0.7692	25
Morph-gradient	0.7967	17

NI dataset

NI dataset	Test accuracy	Best epoch
No edge detection	**0.8947**	27
Canny	0.1478	1
Laplacian	0.8129	27
Roberts	0.8845	27
Sobel	0.8933	29
Morph-gradient	0.8728	25

Analysis

In both DD and NI dataset we see that the AlexNet performs better when no edge detection is applied to the images. The edge detection doesn't seem to have any effect on the training time either, the model fed normal images always converging faster or at the same rate as those who were fed edge maps. Despite my hypothesis of models being able to perform better when fed edge maps being wrong, my hypothesis of models trained on edge maps performance dropping when the objects have more similar shape could seem to be correct.

Starting in DD dataset, the no edge detection model has the highest accuracy on the testset by a rather large margin. I believe this large margin difference has to do with the fact that the object depends more on color then shape to make these predictions. Dandelion and daisy are 2 flowers that have very similar in shape with their round head and long stem, their differences are more dependent on texture and color. It's should therefore be very believable that the model has a harder time classifying these images then those of the NI dataset when we remove features like color and texture that are necessary to make more accurate classifications. Looking into some of the edge map datasets we also see that the edge detection probably does more evil than good. This often happens if the background is very noise like. For example, if the

flower is growing out of gravel. Edge detection is then also performed on small stones around the flower, these are not relevant to the classification and could make it harder to spot where the flower is present in the image. This issue could probably be reduced by passing the image through a low pass filter reducing the number of small edges.

Strangely enough Laplacian on the DD dataset performs better than all other edge detectors despite being the most sensitive to noise. It seems that when the Laplacian is applied to a noisy background as gravel ground, the object becomes clearer as a black object instead of a white. In a sense, the background is turned to white while the object is black. This could be something the network could pick up on, to both predict white and black shapes of objects. Learning this fact, the model might be able to have a better prerequisite than that of the other object detectors.

For NI dataset, the performance difference is a lot smaller. In this dataset the object differs a lot in shapes, this is possibly the reason why there isn't as big of a performance difference as seen in the DD dataset. The object differs more in shape and therefore isn't as dependent on texture and color as in the DD dataset. Still applying no edge detection to the image still seems to have better performance. However, Roberts, Sobel, and Morph-gradient in close follow. These create strong edge maps and aren't as sensitive to noise as the Laplacian. This is probably the reason the Laplacian is one of the worst performers on NI.

As can be seen from the results tables canny worked less than great on any of the two datasets. In the DD dataset it achieves the worst accuracy out of all the models and in NI the model couldn't converge on anything useful. The canny edge detector seems to be very threshold parameter dependent on each individual image for it to work. Meaning that sometimes if the threshold parameters aren't set right for an image, some edges of the image might not be registered or to many edges are registered. This means that sometimes objects might have missing edges or are hidden in lots of edges that aren't related to the object. This would obviously make it harder on the network to learn how object look when certain features are missing. In NI I think the case is so bad that the whole training scheme needs to be redone for the model to learn anything. Changes in the learning rate and threshold parameters are probably needed for the model to converge but based on the observed results it probably wouldn't perform that well either way.

Deep learning models like AlexNet doesn't use feature extractors. The network itself extracts features and performs the classification of those features it extracted. For AlexNet the convolutional and pooling layers creates a feature map, while the MLP part will do the classification by passing the feature maps through a couple of perceptron layers and finally make the classification. Having the feature extractor as a part of the network, the feature extractor becomes learnable, this means the network is free to learn whatever features the network should extract for the feature map to make the best predictions. This means that the network possibly can learn to look for certain colors and texture in different shapes. Removing color and texture, the results would suggest that we are taking away features that the model could use to make good predictions. In a sense, limiting how much the model can learn from the images.

With the observed results we should be able to conclude that with learnable feature extractors we are probably better off to just let the network work out which features it needs to look for on its own. Taking away color and texture seems to limit how much the model can learn, especially when objects shapes are very similar between different classes. However, some edge detectors were able to create very near performance to that of a model with no edge detection. So, the idea of feeding edge maps to a deep learning model shouldn't be dismissed completely. But when you should feed edge maps is

probably very image case dependent. The above results suggest object classes should have varying shapes and not be color or texture dependent. Night vision images of varying object shapes might be a good case for using edge detectors before feeding images. Such images shouldn't be very color or texture dependent and if the objects are varying in shapes, it might be the perfect case for feeding edge maps instead of the normal image.

Conclusion

In this project, I trained multiple AlexNet models on 2 datasets and the edge operator variations of those datasets to see if feeding edge maps of images has any performance improvements. Observed results would suggest that applying edge detectors before feeding them to a deep learning model do not yield any performance improvements. Performance on edge map fed models also seems to decline if object classes are less dependent on shapes and more on color and texture. However, the performance for some edge detectors weren't that far off from just feeding the normal image to a model. The idea of feeding edge maps should therefore not be completely dismissed. Feeding edge maps might be more beneficial when applied to images of object with varying shapes and were color and texture doesn't play a major role in making good classifications.

The report limitations

The report is limited to only 5 pages. A deeper explanation of how the edge detectors work has therefore been left out and is only explained in small detail. This project contains a lot of images as I created multiple variations of the two datasets, additionally, the project also contains pretrained weights for the AlexNet for the different datasets. As of this writing I haven't managed to load everything on Canvas because of size constraints. In the case of missing image datasets and pretrained AlexNet weights please follow the README.md to recreate the project results.

Sources:

https://machinelearningmastery.com/types-of-classification-in-machine-learning/

https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

Image analysis: presentation slides of different lectures