Deep Learning in Computer Vision - Project 1

Till Ariel Aczél (s203216), Jan Piotr Latko (s193223), Jonas Søbro Christophersen (s153232), Technical University of Denmark, Deep Learning in Computer Vision (02514), 10/06/2021, Github Repository

Part 1: Hotdog or Not Hotdog

Introduction
The purpose of this part is to make a CNN capable of classifying images into two classes: hotdog and not hotdog.

Data
The data consists of 3986 labelled images, where 2047 of these contain a hotdog. The images come from the ImageNet categories: pets, furniture, people, food, frankfurter, chili-dog, hotdog. All images are resized to a 224x224 resolution, and standardized using mean and standard deviation values commonly used for ResNet. During training, input-images are augmented using crops, horizontal flips and color jitter.

Network
The network used is the ResNet50, with the last layer substituted by a Linear classification layer with 1 output neuron to fit this binary-class problem. We used Adam optimizer with a 0.0003 learning rate and a plateau learning rate scheduler.

Results
Using a powerful model as ResNet50 we expected it to overfit the training set, as it contains less than 4000 images. This can be observed in the losses: our training loss is 0.1709 compared to the 0.5526 validation loss. Transfer learning is a powerful tool to combat this issue. Learning curves comparing the effect of utilizing transfer learning on ResNet50 is visualized, along with saliency maps for a few test images. The ResNet50 achieved a validation accuracy of 94.41%.

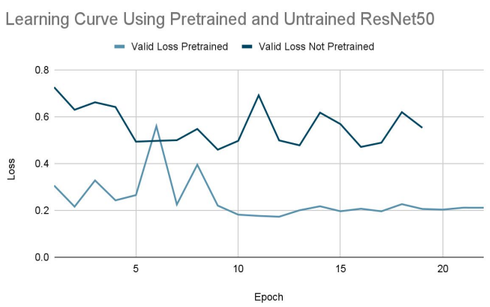

Figure 1: Validation loss using pretrained and untrained ResNet50. Utilizing transfer learning decreases validation loss substantially.

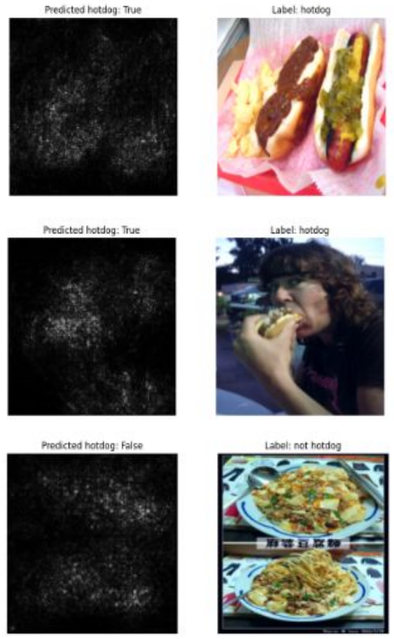

Figure 1: Gradient-based pixel attribution. The intensity of the pixel corresponds to the magnitude of the gradient wrt. the predicted class probability. For images of hotdogs, the network prediction is mostly affected by pixels containing a hotdog compared to regions without hotdogs.

Discussion & Further Work
Due to time constraints we cut several corners during this project. First we overestimate our model performance as we did not have a test set and performed the final evaluation of the models on the validation set. However, in this case it should not be an issue, as hyperparameters are not tuned. If time would allow it, we would have used an automatic hyperparameter tuning algorithm like bayesian hyperparameter optimization.

Model performance can be further improved by combating overfitting. Transfer learning can be improved by using different learning rates for the pre trained weights and the new weights, and gradually increasing the pre trained learning rate. Having a bigger, higher quality dataset could help, but would be expensive. Using other (and fine tuning the already used) data augmentation could be an easy way to improve model performance.

Part 2: SVHN Bounding Box Identification

Introduction
The purpose of this part is to make a CNN capable of detecting and classifying house numbers by using a sliding window algorithm.

Data
The data comes in two formats, of which the first format is used for testing, and the second for training. Format 1 contains images of varying resolution images of house numbers. Format 2 contains centered house digit images, with a fixed 32-by-32 pixel resolution. We expanded this dataset with images of house walls and backgrounds as negative samples. About 28% of the data were negative samples. Like in Part 1, all images were standardized using mean and standard deviation values commonly used for ResNet. During training, input-images were augmented using crops and color jitter.

Network & Sliding Window Method
Again, the ResNet50 network was used to build a classifier for the single digits, with the last layer being a Linear layer with 11 outputs - one for each of the 10 digits and one for 'wall'. The hyperparameters were the same as in Part 1.

Once the network was trained, the bounding boxes were identified using a convolutional implementation of the sliding window algorithm. This is exemplified in Figure 3.

Figure 3: Sliding window algorithm. The window traverses the entire image and feeds the selected region through the trained ResNet50 classifier.

Bounding boxes are generated by selecting the receptive field of outputs which yields the highest predicted probability for a given digit class. E.g. when the window is directly on top of the digit '7', the predicted probability for this class should be high, and a bounding box is generated here. Bounding boxes are also pruned using non-max suppression.

Results & Discussion
The model achieved a validation accuracy of ~95% validated using the Format 2 data set, which is low for a task where we need to classify multiple times per image. To enable to sliding window algorithm using the model the images of Format 1 had to be resized, so the sliding window (receptive field) would fit the size of the house numbers. The sizes of house digits in relation to the size of the images were investigated for the Format 1 data set, and it was seen that this ratio was similar for most images. All images could thus be scaled to the same resolution, and the receptive field would be similar to the the size of a single digit.

Some examples of bounding boxes generated by the network is seen below.

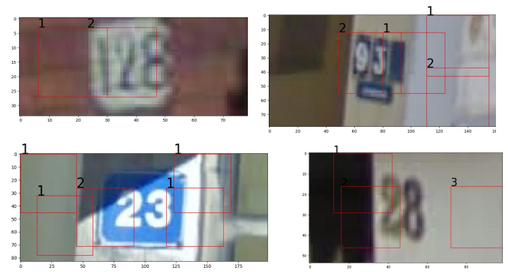

Figure 4: Generated bounding boxes with predicted classes.

The model is prone to some error, as there are several instances of boxes generated far from digits. It should also be noted that the network predicts lower labels (digit 1, 2 and 3) more often than other labels. This is due to an unbalanced dataset. Out of the number of classes 19% were of number 1, 14% of number 2 and 11% of number 3 (this is due to Benford's law). This imbalance could be mitigated by using weighted sampling corresponding to the class proportions.

The model failed to generalize from the classification setup to the sliding window domain. We hypothesize that the classification data had slightly different characteristics (digits were centered and scaled) and the resulting model was not robust enough. We could promote model robustness by generating the classification dataset in a similar manner as used for detection or further augmenting the data.

Additionally, improvements discussed for the Part 1 model would likely prove beneficial in this case also.