Precog Programming Task Report

Harshvardhan Rana

Introduction

I had chosen to work on tasks related to the theme Computer Vision. I have done all the tasks of Computer Vision including the bonus tasks. The tasks were very interesting and I have learnt a lot from working on these tasks as they were a mixture of computer vision, image processing and NLP.

All the machine learning was done in PyTorch and the image processing was done using Open-CV.

Object Detection

Methodology

For object detection, I used fasterrcnn_resnet50_fpn model from torchvision.models.detection. This model is based on the COCO dataset which contains 80 classes. The reason I chose this model was because of its good accuracy. I used other models like resnet50 from torchvision.models which was based on ImageNet dataset which has 1000s of classes but they were not very accurate. Using a model which has 1000s of labels for the hateful memes challenge which has around 10000s of labelled images would have also been challenging.

After running the object detection model on 2000 images. ‘Person’ was the most occurring class of object in the images with 6931 appearances in all images with ‘tie’ being the second most occurring with 511 appearances.

For toxic memes, ‘Person’ was the most occurring class of object with 3674 appearances in all toxic memes with ‘Tie’ being the second most occurring with 262 appearances.

For the bonus subtask which asks us to develop an image classification system using the number of appearances of objects that we have, I developed a toxicity score for each object which is based on the appearances in toxic memes and in all

memes. The toxicity score of an image is then the average of the toxicity scores of the objects in the image. Formally,

$$Toxicity\ Score\ of\ an\ object = \frac{Number\ of\ appearances\ in\ toxic\ memes}{Number\ of\ appearances\ in\ all\ memes}$$

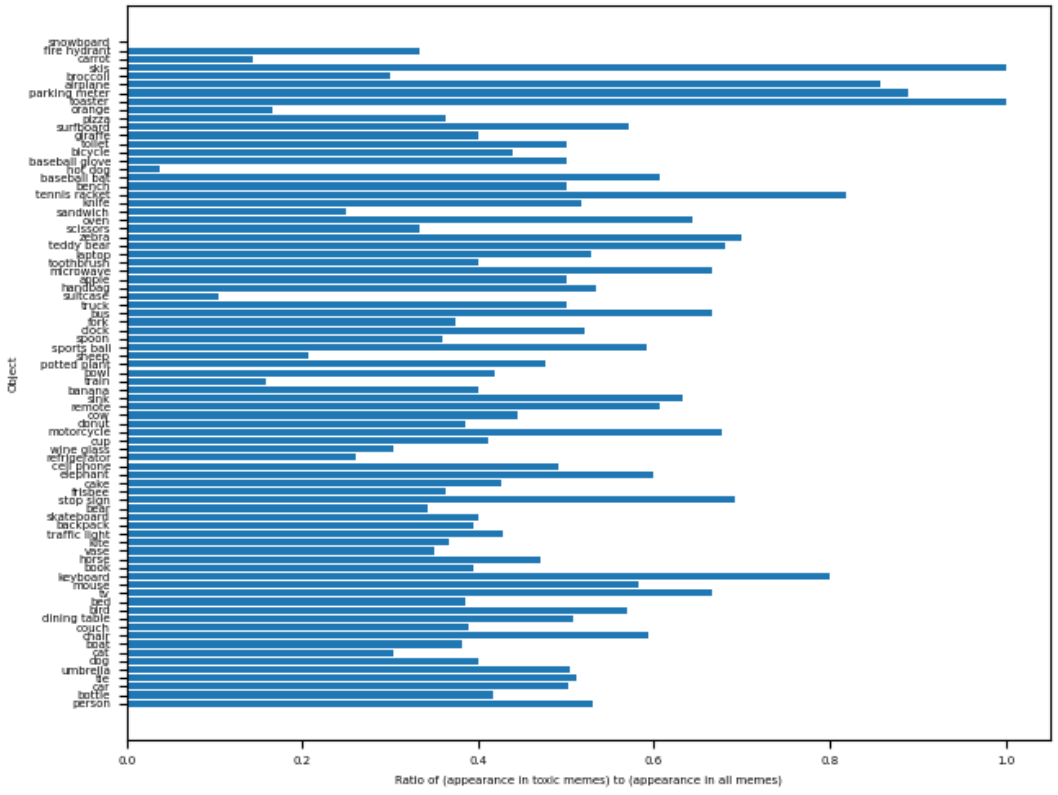
$$Toxicity\ Score\ of\ an\ image = \frac{\sum Toxicity\ score\ of\ object}{Number\ of\ objects\ in\ the\ image}$$

Then the images are then classified if they are toxic or not based on if their toxicity score is crossing a certain threshold T.

Insights

After experimenting with the threshold T, the average accuracy of the model was still around 50%. This meant that the system was as good as randomly picking the answer.

This implies that for a classification system, we would have to consider the text as well and not just the objects.

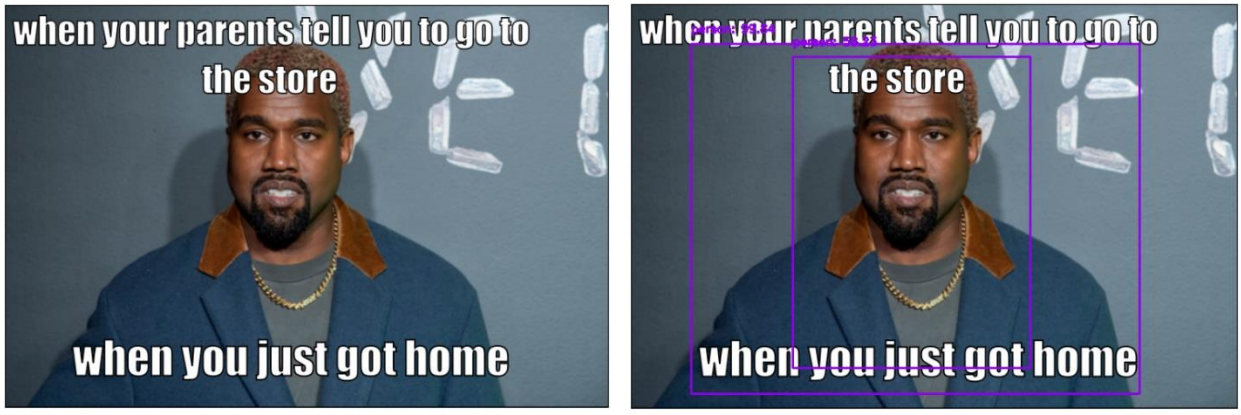


Plot of toxicity score of objects

Caption Impact Assessment

Introduction

Text on image confuses the object detection model. It even starts detecting objects that are not present in the image.



In this case, 2 ‘Person’ objects are detected because of the text.

Methodology

For removing the text from the image, I first generated a map of all the text that is present in the image. Since in the dataset, the meme text is white, it is easy to just take all the pixels whose brightness is more than 249. The map is then dilated to increase the area of the text. This makes sure that the border of the text is also colored.

Finally the original image is inpainted where the map pixels are white.

Now the objects are being detected properly in the image.

Classification System Development

Methodology

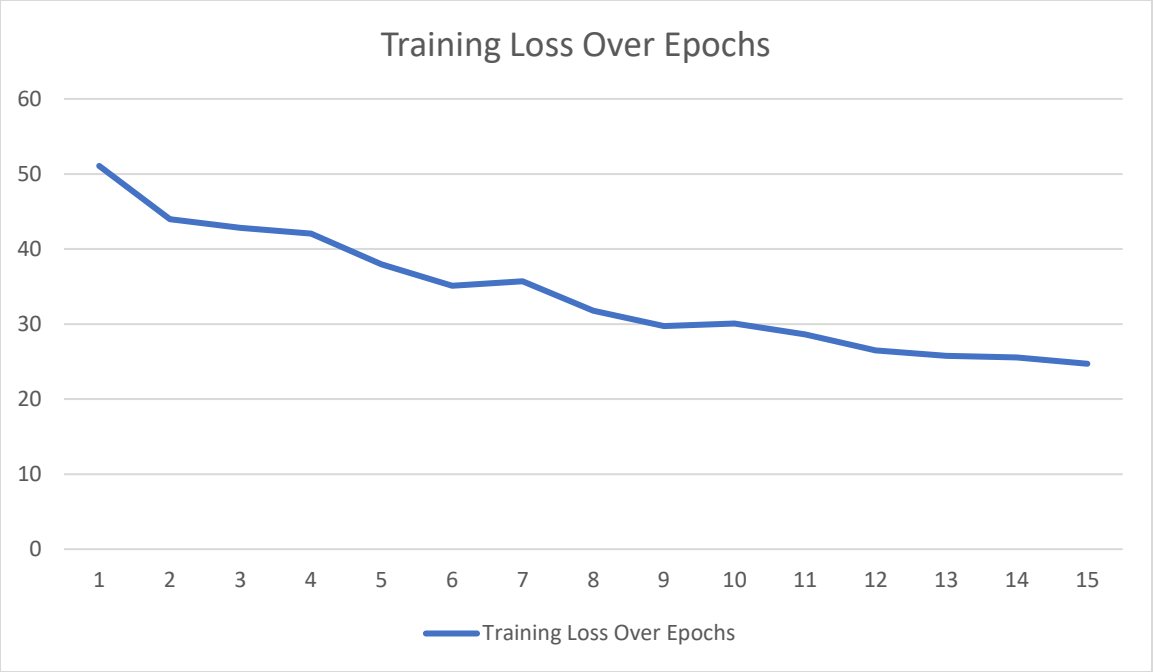
I developed a system which classifies whether an image is a meme or not a meme using a Convolutional neural network which was built with PyTorch. I chose to classify using CNN because they are effective in finding patterns in images to recognize objects, text, etc.

The model consists of 2 2D Convolutional Layers, 1 MaxPool Layer and 2 linear layers and the loss function used was CrossEntropyLoss.

The dataset for the non-meme images was sourced from https://www.kaggle.com/datasets/puneet6060/intel-image-classification/data. The dataset contained images that 150x150 in size. For this I had to convert the memes to 150x150 pixels as well. This helped in speeding up the training of the model as well. CUDA also significantly sped up the training.

Insights

The model took about 10 minutes to train with 2000 images which were evenly split between memes and non memes. The accuracy of the model was around 90% on the test data.



Bonus Task: Classifying memes based on text.

Methodology

I created a classification model based on the BERT model (transformers.BertModel.from_pretrained("bert-base-uncased")) with just 1 linear layer on top. I trained it on 3200 labelled texts with 800 labelled texts as testing data. Even after training for 15 epochs, the accuracy of the model didn't improve significantly.

The accuracy of the model on the testing data was 60% so it was better than random. To improve the accuracy further, the objects of the meme would also have to be considered.

To include the objects in the consideration, the objects of 4000 images were extracted for training purposes. This took around 2 hours. The objects were then captioned with text in the format, "objects: object_1, object_2, etc.". For example, an image with a cat and a person was captioned as "objects: cat, person." This caption was the concatenated with the text string. These concatenated strings were then used for training.

To use it on real images instead of labelled data, I had to use tesseract ocr to extract text from the model.

Insights

After training for 20 minutes, the accuracy of the new model was around 80% on the test data which was a great improvement over the previous model which only considers the meme text.

Improvements

Right now, the model only considers the object and the text in the image. It would be better if the model also considers the actions that are going on in the image. For example, when an object ‘Person’ is detected, we don’t know what the person is doing. A captioning model which could give the description of the image would increase the accuracy a lot further.

The model also thinks that the object caption is a part of the meme text. Another improvement would be to use a multimodal Bitransformer (MMBT) that could treat text and objects differently while considering both in its output.