Report

Task 1 - Object Detection

Dataset Used

- A toy dataset of the Hateful Meme Challenge is used for the Object Detection task which comprises of 1500 training images inorder to decrease the training time

Preprocessing

- Used the PIL→Python Image Library to open the image from the given image path.
- Used torchvision.transformers.functional module to extract the PIL image in the form of a tensor. The resulting tensor consisted of the raw pixel values of the meme.
- Used torch.unsqueeze(dim=0) to add a dimension to the torch tensor and change its representation. Suppose for a N x D vector it becomes ambiguous where the new dimension needs to be added, which is parsed as an argument to the function.
- Normalizing the image → The image consists of the RGB value which range between 0 to 255, however the Pretrained Neural Network on the COCO dataset showed better results if the range of RGB values was set to be between 0 to 1.

Training Dataset

- COCO Dataset (Common Objects in Context) is a large-scale dataset of images and annotations for object detection, segmentation, and captioning.
- It contains over ***330,000 images***, each annotated with ***80 object categories*** and ***5 captions*** describing the scene
- The images are organized into a hierarchy of directories, with the top-level directory containing subdirectories for the train , validation and test sets. The annotations are provided in JSON format

Model - Pretrained Faster R-CNN

- The Model used for object detection for the task of Object Detection and Recognition is Faster RCNN supported by ResNet-50-FPN backbone trained using the above COCO dataset
- Faster R-CNN is composed of 2 modules → the Region Proposal Networks (RPN) and the Fast R-CNN detector
- RPN's are trained using image-centric sampling strategy using back-propagation and Stochastic Gradient descent. Multiscale anchors are determined which are translational invariant and are cost-efficient. For anchors, 3 scales with box areas of 128^2,256^2, and 512^2 pixels, and 3 aspect ratios of 1:1, 1:2 and 2:1 were used
- Loss Funtion using the probability scores of the positive and negative anchors and parametrized co-ordinates of the bounding boxes is given in terms of Classification loss and regression loss by :

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

- Shared convolutional layers are initialized by drawing weights from 0 mean 0.01 std deviation Gaussian distribution , use learning rate of 0.001 for 60k mini-batches and 0.0001 for next 20k mini-batches.
- 4-step alternating training is used, 1) Train RPN initialized with ImageNet pretrained weights, 2) Train a seperate Fast R-CNN using proposals generated by RPN using same pretrained weights , 3) Use detector network to initialize RPN training to train the shared convolutional layers and only fine-tuning the RPN layers. 4) finally fine-tune the layers of Fast R-CNN
- Cross-Boundary anchors are ignored , about 6000 anchors per image are there for training 1000 × 600 image. Also non-maximum supression (NMS) is applied to cls scores which leaves about 2000 proposal images
- Model is trained on COCO dataset with following changes : a) 8GPU implementation 2) 240k iterations with learning rate 0.003 and 80k iterations with 0.0003 due to change in mini-batch size for both the R-CNN and RPN steps 3) 3 aspect ratios and 4 scales (adding the 64^2) for handling

small objects 4) In Fast R-CNN step, the negative samples have a maximum IoU with ground truth in the interval of [0, 0.5), instead of [0.1, 0.5)

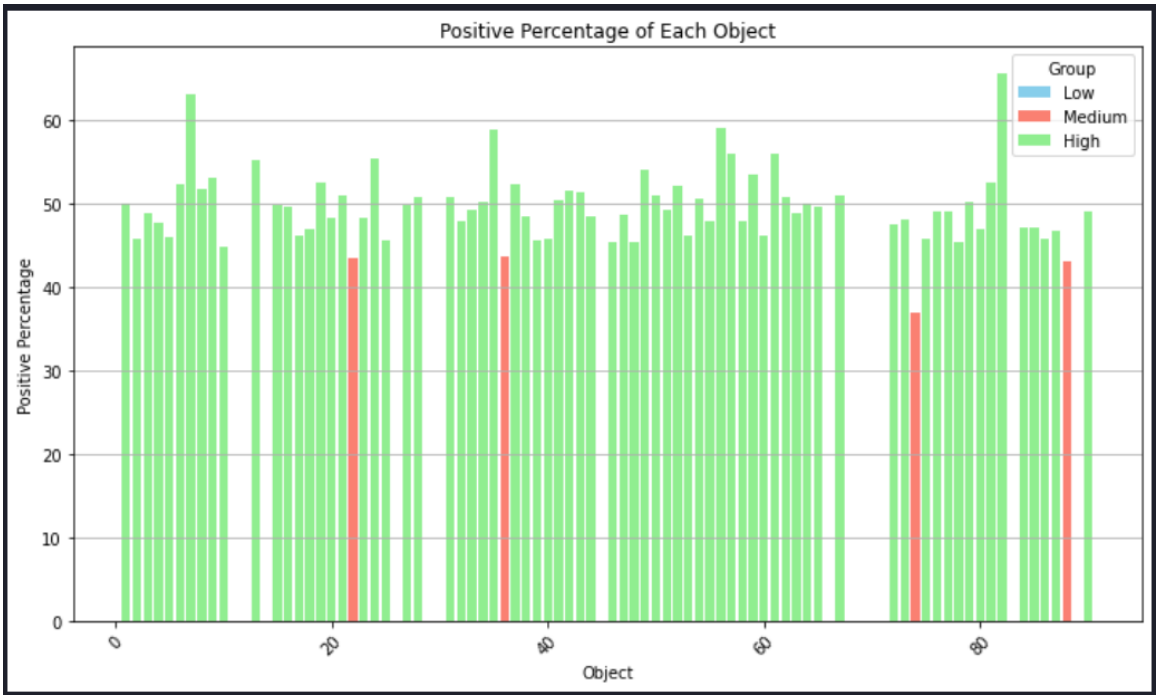
- https://arxiv.org/pdf/1506.01497v3.pdf link to research paper

Experimental Details

- Inorder to reduce computational time the Task of Object detection was performed on a comparatively smaller dataset of 1500 images Toy dataset
- For an input image the model provided a dictionary of 3 values :
  a) Co-ordinates of bouding boxes
  b) Classes of the corresponding boxes
  c)Confidence scores of each predicted class and bounding box
- Only the bounding boxes with a confidence score greater than 0.3 are considered for further analyses

Results

- The results of the Data Cataloguing and Analysis were not decisive i.e. the object classes detected in the Hateful as well as the non-hateful memes had a probability range of 40-60% for the hateful meme class.
- Due to such an observation it is not feasible or wise enough to use this idea for the classification of images however, if one wants to do so they may proceed with an Accuracy of about 45-55% which is very close to the guessing accuracy of 50% which shows that the idea doesn’t hold
- However on deeper analysis one may derive the result by varying the threshold confidence score but 0.3 is optimal enough to detect multiple important feature objects in the image



Task 2 - Caption Effect

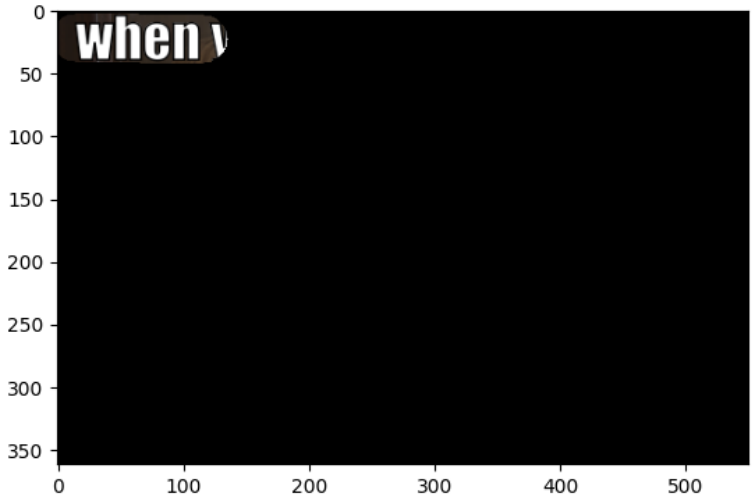
Dataset Used

- A toy Dataset of 200 images is used for analyses whether the caption of the image has an effect on the task of object detection
- The 200 images are the first 200 images of the above toy dataset used for Object Detection

Preprocessing

- Here, preprocessing forms the major task as we need to find the effect of caption/text present on the image from the perspective of object detection.
- The idea is to extract the boxes where the text lies and in-paint it with the colours of the neighbouring pixels so that the text disappears from the image
- Using the Keras-OCR pipeline to detect the bounding boxes of the text/words present in the image as shown in the below example

- Determining the midpoint for each bounding box and the corresponding thickness so as to decide the region to be painted so as to remove the text
- Using cv2.line module so as to see the part of the text which was coloured
- Using the cv2.bitwise_abd to get the part which had been coloured



- Finally the image is inpainted using the cv2.inpaint library so as tho obtain the inpainted image

Model

- The model used for training is the same Faster R-CNN using ResNet-50-FPN as a backbone as the previous task. It is kept same so as to get a better understanding of the effect of Captions on the image in the task of object detection
- The ouput of images before and after text removal turns out somewhat like this

Task 3 - Image Classification

Creating Dataset

- Creating a dataset which consists of Images which are memes and those which are not memes
- Meme images are extracted from the Hateful Meme Challenge , 6992 Meme Images dataset whereas the training images are taken from the Random Images , Flickr Dataset,etc
- For the training of the Image classification model , we have used images from the Hateful Meme Challenge Dataset all of which were given the label 1 → is a meme whereas the images of non-memes were extracted from Random-Images Dataset all of which were given a label of 0 → is not a meme
- For the testing dataset , the images were sourced from other resources so as to get an appropriate judgement of the model , meme images were extracted from 6992 Meme Images whereas the non-meme images were extracted from the Flickr Image dataset
- All the images were not used for prediction and a total of 3618 meme images and 2035 images of non-images were used for testing

Preprocessing

- All the images were converted to a size of 224 × 224 pixels using the cv2.resize after reading the image using cv2.imread
- After that, to conbert the read values from BGR format to RGB format cv2.cvtColor was used
- Finally the image tensor was transposed to get the the values in the format of 3 × 224 × 224 shaped tensors
- On training data the similar preprocessing was done so as to obtain the tensors

Model

- **ReLU →** Rectified Linear Unit (ReLU) is one of the most commonly used activation functions in neural networks. Introduces non-linearity to the model increasing the model complexity still being computationally inexpensive. Unlike sigmoid and tanh functions, ReLU does not saturate for positive inputs, preventing the vanishing gradient problem during backpropagation.

$$f(x) = \max(0, x)$$

- **Max Pooling** → Max Pooling is a downsampling operation that reduces the spatial dimensions of the input volume in a Convolutional Neural Network (CNN) while preserving important features. It is an important layer providing Dimensionality Reduction , Translational Invariance and Feature Preservation

$$\text{Max Pooling}(x, i, j) = \max_{m,n} x_{i+m,j+n}$$

- **Dropout →** Dropout is a regularization technique commonly used in neural networks to prevent overfitting and improve generalization. This forces the network to learn redundant representations and prevents neurons from relying too heavily on specific features.

$$\text{Output}_{\text{dropout}} = \text{Output}_{\text{original}} \times \text{Mask}$$

- **Structure →** The following is the structure of the Neural Network used to train the dataset
  a) Convolutional Layer with ReLU activation (kernel size : 3 × 3 , padding : 1

x 1 ; I → 3 , O→32)
b) Max Pooling Layer (kernel size : 2 × 2 , stride : 2)
c) Convolutional Layer with ReLU activation (kernel size : 3 × 3 , padding : 1 x 1 ; I→32 , O→32)
d) Max Pooling Layer (kernel size : 2 × 2 , stride : 2)
e) Convolutional Layer with ReLU activation (kernel size : 3 × 3 , padding : 1 x 1 ; I→32 , O→64)
f) Max Pooling Layer (kernel size : 2 × 2 , stride : 2)
g) Dropout (Probability : 0.4)
h) Flatten / Reshape
i) Fully Connected Layer with ReLU activation (I→64 × 28 × 28 , O→ 128)
j) Fully Connected Layer (I→128 , O→2)
k) SoftMax Layer

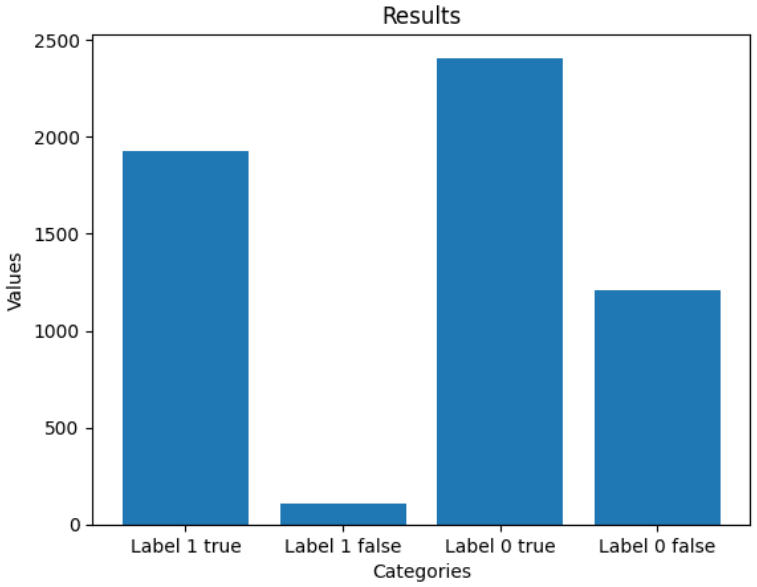
- **Adam's Optimizer** → Adam maintains two moving averages of the gradients: the first moment (mean) and the second moment (uncentered variance). These moving averages are used to adaptively adjust the learning rates for each parameter.
- **Cross Entropy Loss Function →** This is the Loss Function which is optimized by the loop of Gradient descent. It measures the dissimilarity between the predicted probability distribution and the actual probability distribution of the target labels.

$$\text{Cross-Entropy Loss}(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right)$$

- Number of Epochs is set to 5 (Trained 1 with 10 epochs but couldn't store due to some issues) and The size of batch for batch normalization is set to 32

Results

- Outputs a 2 dimensional tensor giving the probability values of the 2 classes
- The accuracy of the Model turns of to be 76.72%
- The accuracy for detecting the memes is about 66.56%
- The accuracy for detecting the non-memes is about 94.79%



- The results can be improved significantly by using deeper neural networks and increasing the number of epochs from 5 to 500 which would increase the accuracy by about 5%
- Also a decrease in the accuracy occurs because of use of testing images not being a part of the same dataset and thus, leading to variations
- Accuracy can also be increased by carrying out more appropriate preprocessing like Mean Centering and Normalization

Task 4 - Image Captioning and Hatred Detection

Dataset

- The dataset used is a toy dataset of the Hateful Memes Challenge comprising of 500 images so as to analyses the impact of visual semantics and the overlying text on the hatefulness of the meme
- Use of toy dataset allows us to train the model at better time rates and also perform various analysis on the predicted ouputs

BLIP (Bidirectional Latent Intention Prediction)

- BLIP incorporates bidirectional information flow and latent intention modeling to enhance prediction accuracy, offering a promising approach for various NLP applications.

- A pretrained opensource model of BLIP is used from the HuggingFace website using the tokenizer and the model from the Transformers library
- The model is used to generate caption for the input image taking into consideration the visual aspects of the image so as to detect the hatefulness in the speech.
- Important Preprocessing includes skiping of special tokens and converting the image from BGR to RGB format
- Some examples of generated captions:

Generated Caption : `there are two different pictures of a man with a bald head`

Generated Caption : `there is a man and woman that are kissing each other`

Hate-Speech Detection Model (C-NERG BERT)

- We then load a hate-speech detection model so as to detect the hatefulness or offensiveness of the generated caption as well as that of the text overlaid on the meme image
- The model used is Hate-speech-CNERG from the HuggingFace opensource API which takes in the text as input tokenizes it and provides a label for the Overlaid text as well as the generated caption
- The model outputs 3 labels given by 1 - normal , 0 - hateful , 2 - offensive.
- These labels are stored in the jsonl file for further analyses

Sentence Matching

- The model then performs a sentence matching between the overlaid text and the image caption so as to identify the Benign Confounders among the dataset
- Benign confounders are examples which convert a hateful meme template to non-hateful by changing the overlying text or underneath image which is mostly the description of image

- The model used to get the embedding of the overlying text and the caption is Sentence Transformers with pretrained weights which gives the

embeddings for a list of sentences

- Embeddings are also preprocessed by using Mean pooling and Normalization so as to increase the accuracy of the output
- Finally the Embeddings are converted to a similarity score with the help of a cosine similarity function. However, other similarity functions such as Euclidean and Manhatten Distance may show a lower accuracy.

Analysis

- A detailed analysis of the generated caption , hatred score from the caption and the ground-truth label shows that the visual aspects of the memes do not contribute to the hatefulness of the memes as about 98% of them showed normal label
- Mainly the overlying text acts as the culprit of making the meme hateful due to offensive and hateful speech as detected by the model
- Some memes also show cases where both the caption and the overlying text are normal but the ground-truth label identifies the meme as hateful. This is due to the holistic assimilation of the context of the meme
- Finally the careful observation of sentence matching helps us to find the benign confounders which are mostly the non-hateful memes generated with the same image template which can be analysed distinctively
- Cosine similarity results in some negative scores which can be hooked to 0