Image Classification with Distribution Shift

Ashish Singh Bisht ABISHT17@YAHOO.COM

1. Introduction

Image classification is a fundamental task in computer vision, aiming to identify and localize objects within images. Despite significant advancements, a persistent challenge is distribution shift. This project addresses the variance between a training set with diverse backgrounds and a testing set with a uniform background, which can lead to reduced model performance.

To tackle this issue, we focus on adapting the `Faster R-CNN model`(Ren et al., 2015), renowned for its accuracy and speed in object detection, to perform well across different data distributions. Our solution leverages bounding box information to focus on objects of interest, reducing the impact of the uniform backgrounds in the testing set. We evaluate our approach on a custom dataset and analyze its performance to understand the effectiveness of our strategies in improving classification accuracy and robustness.

2. Model Description

The model I decided to utilize for this study is based on `Faster R-CNN` architecture, which is a state-of-the-art approach for object detection tasks, the considerable variation is that my model used `Faster R-CNN_ResNet50_FPN v2` with `ResNet-50` as the **backbone** along with `FPN (Feature Pyramid Network)`. This model has been fine-tuned to perform classification tasks on a custom dataset.

Residual Network or `ResNet-50` is a deep neural network which is responsible for feature extraction from the input image. The FPN enhances the backbone by creating a pyramid of feature maps that improves detection of objects at multiple scales.

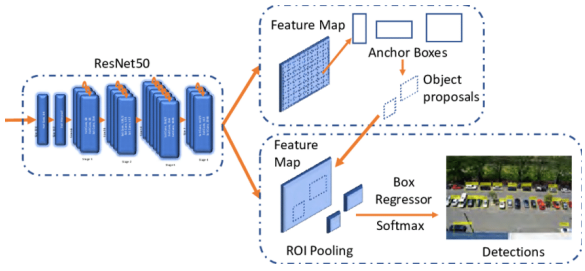

Figure 1: Architecture of Faster R-CNN ResNet50 FPN

The model takes an input image, which can vary in size. This image is passed through a `ResNet-50` network, a deep convolutional neural network with 50 layers. It generates a detailed feature map from the input image. This feature map captures essential

information about the image at various spatial locations. The next step involves the `Feature Pyramid Network (FPN)`, which creates multiple anchor boxes at each spatial point in the feature map. These anchor boxes vary in size and aspect ratio, allowing the model to effectively detect objects of diverse dimensions and shapes. Each anchor box is then assessed for its overlap with ground truth bounding boxes to classify them as positive or negative. Positive anchors closely match the ground truth, while negative anchors do not.

To refine these anchor boxes, a 1x1 convolutional network predicts the necessary adjustments, which are applied to transform the anchor boxes into more accurate region proposals. This process is managed by the **Region Proposal Network (RPN)**.

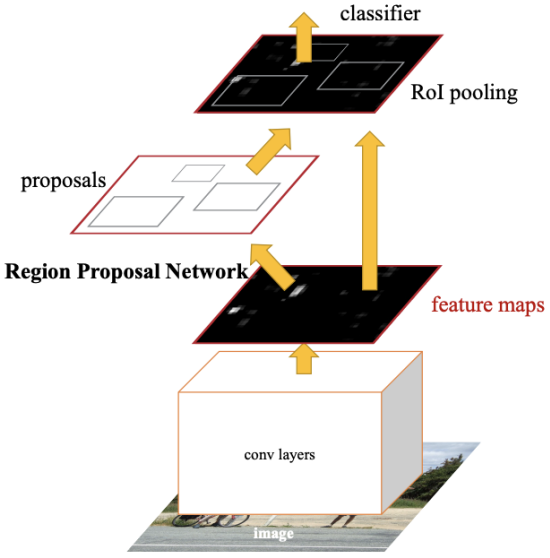

Figure 2: RPN Architecture

For each proposal generated by the `RPN`, the model employs a process called `RoI` Align to extract a fixed-size feature map from the original feature maps. This process ensures that the features are accurately aligned with the proposal, even if the proposal doesn't perfectly align with the grid of the feature map. This vector is then analyzed by two fully connected layers: the classification layer (cls) evaluates whether each region proposal contains an object or should be categorized as background, while the regression layer (reg) predicts a refined bounding box for each object. The classification layer provides an objectness score, indicating the likelihood of an object being present, and the regression layer adjusts the bounding box coordinates, ensuring precise object localization.

3. Dataset

The dataset that utilized in this study was provided by my professors Mr Concetto Spampinato and Mr Giovanni Bellitto and was presented in the form of a kaggle competition. It

comprises of images and associated annotations for object detection tasks, organized into training and testing subsets. The dataset contains:

- 8 categories
- 1,600 images for Training
- 800 images for Testing

Figure 3: Train set Sample

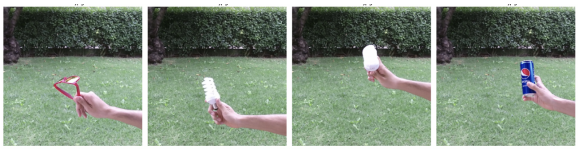

Figure 4: Test set Sample

These images (`Figure 3`) are composed of a variety of backgrounds unlike the testing set (`Figure 4`) that consists of a uniform background which leads to the problem of Distribution Shift.

Annotations for the training images are also provided in a CSV file (`train.csv`), detailing bounding boxes in the format `[min_x, min_y, max_x, max_y]`, which correspond to the original image dimensions ultimately making our model to train on the data inside the bounding boxes on a custom Dataset.

The `Albumentations` library was used to enhance the robustness of the model through various image augmentation techniques. `Horizontal Flip` with a probability of 50%, `Random Brightness and Contrast Adjustment` also with a probability of 50%, `Gaussian Blur` 30% probability and `Random Rotation` with a limit of 15 degrees and a probability of 50%

4. Training details and procedure

1. **Evaluation Metrics**

    - To define the training procedure, we start with the evaluation metrics. The primary evaluation metric used in the training process is the loss, which includes both classification and bounding box regression losses. The classification loss, often a cross-entropy loss, measures the performance of the model's classification component, assessing how well the model can identify the object categories. The

bounding box regression loss, typically a Smooth L1 loss, evaluates the accuracy of the bounding box predictions, ensuring that the predicted boxes closely match the ground truth annotations. The combined loss function can be expressed as follows:

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

where L_{cls} is the classification loss, L_{reg} is the regression loss, p_i is the predicted class probability, p_i^* is the ground truth class, t_i is the predicted bounding box, and t_i^* is the ground truth bounding box. The term λ is a weighting factor to balance the two loss components, N_{cls} is the number of classification samples, and N_{reg} is the number of regression samples.

2. **Training procedure**

- Epochs: The total number of epochs used for training is 6.
- Batch Size: The batch size used for training is 8.
- Validation: It is performed using a test split. The dataset is divided into training and testing subsets, the testing subset is used to evaluate its performance.
- Stopping Criteria: The training ran for a fixed number of epochs (6 in this case).

3. **Training details**

- **Optimizer:** The optimizer used for training is Stochastic Gradient Descent (SGD) with the following parameters:
  - **Learning Rate:** 0.005
  - **Momentum:** 0.9
  - **Weight Decay:** 0.0005
- **Hyperparameters:**
  - **Learning Rate Schedule:** The learning rate is scheduled using the `OneCycleLR` scheduler with a maximum learning rate of 0.01. The scheduler is set for 5 epochs with steps per epoch equal to the length of the training data loader.
  - **Batch Size:** 8
  - **Number of Epochs:** 6 (on the T4 GPU by kaggle).

  No hyperparameter tuning methods are specified in the provided code.
- **Initialization:** The model's weights are initialized using the default weights provided by the pre-trained `fasterrcnn_resnet50_fpn_v2` model from the torchvision library. The pre-trained model's box predictor is replaced with a new one that has an appropriate number of output classes (9 in this case).

5. Experimental Results

- **Performance**: After Training for 6 Epochs on the kaggle competition my model resulted with an accuracy of **0.94**. We can also see the graph for Average loss per Epoch.

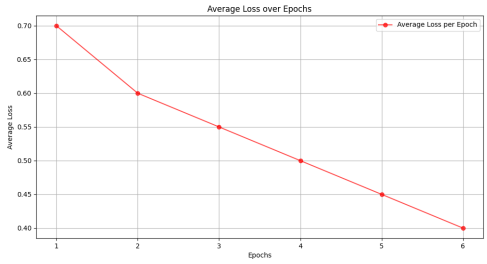

Figure 5: Average Loss per Epoch

- **Comparisons**: Apart from my model i also tried ResNet50 and faster R-CNN with ResNet50 FPN

Model Name	BB	Albumentations	Test Accuracy
ResNet50 (Baseline)	No	No	0.11
Faster R-CNN with ResNet50 FPN	Yes	Yes	0.87
Faster R-CNN with ResNet50 FPN v2	Yes	Yes	0.94

Table 1: Comparison of our model with others.

As we can see the best accuracy I achieved in the competetion was from Faster R-CNN with ResNet50 FPN v2 with bounding boxes and albumentations.

References

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.