SUSTech CS324 Final Project

Fine-tuned SSD Model for Face Detection

Bingwen Dong (12211111) Jinyue Chen (12212251) Ruiyang Dong (12212227)
Diancheng Kang (12212221)

December 2024

1 Introduction

This project focuses on running a deep learning model for face detection on the Jetson Nano development board. Given the limited computational power of the development board, achieving real-time performance requires the use of lightweight models. After evaluation, we selected the SSD (Single Shot Multibox Detector) model for its balance between accuracy and efficiency. This report describes the model's theoretical basis, the datasets used, the training process, and the results of fine-tuning the model for detecting human and cartoon faces.

2 Motivation

The pre-trained SSD model often misidentifies various objects as human faces and lacks the capability for classification. This limitation poses challenges when accurate face detection and classification are required, particularly in applications involving both real human faces and cartoon faces. To address this issue, we propose to fine-tune the SSD model using a curated dataset consisting of real human faces and cartoon images. The goal is to enable the model not only to detect faces but also to classify them into human or cartoon categories effectively. Fine-tuning the model with such a specialized dataset will enhance its robustness and expand its applicability to diverse face detection scenarios.

3 Model Description

The SSD (Single Shot MultiBox Detector) is a one-stage object detection network designed for high efficiency and accuracy. It uses a set of default boxes at multiple feature map scales, allowing it to predict both the bounding boxes and class probabilities directly.

3.1 Network Architecture

The SSD [4] model comprises the following main components:

- **Base Network:** A convolutional neural network (e.g., MobileNet [2] or VGG16 [5]) that extracts feature maps from the input image.
- **Extra Feature Layers:** Additional convolutional layers added to capture multi-scale information.
- **Default Boxes:** Pre-defined bounding boxes at each scale to detect objects of varying sizes.
- **Prediction Heads:** Convolutional layers that predict class scores and box offsets for each default box.

The model structure is visualized in Figure 1. It consists of convolutional layers for feature extraction, followed by prediction heads at multiple scales.

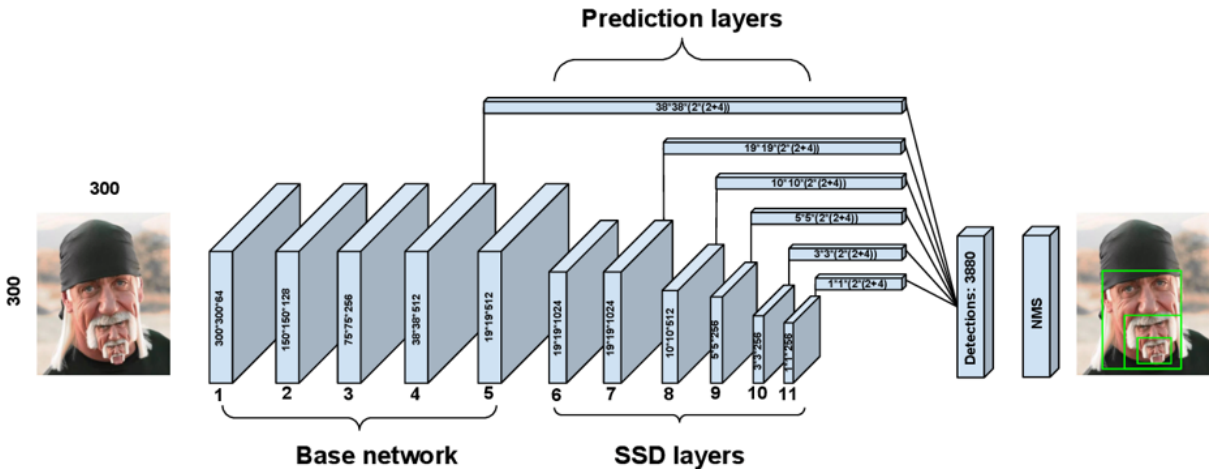

Figure 1: SSD Model Architecture

3.2 MultiBox Loss Function

The loss function used in SSD combines classification loss and localization loss:

$$L(x, c, l, g) = \frac{1}{N} \left(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g) \right), \tag{1}$$

where:

- L_{conf} is the classification loss, calculated using the softmax cross-entropy function.
- L_{loc} is the localization loss, computed as the Smooth L1 loss between predicted box offsets l and ground truth g.
- x is the match indicator (1 if a default box matches a ground truth box, 0 otherwise).
- c represents the predicted class scores.
- α is a weighting factor to balance the two losses.
- N is the number of matched default boxes.

3.3 Model Variants

For this project, we utilized two variants of the SSD model:

- **Slim SSD:** A lightweight variant optimized for speed. It reduces the number of convolutional filters and layers while maintaining accuracy.
- **RFB SSD:** Incorporates Receptive Field Blocks (RFBs) [3] to enhance feature extraction by capturing multi-scale and contextual information.

3.4 Mathematical Formulation for Predictions

For each default box d_k at a given scale, the network predicts:

- **Class scores:** $c_k = \text{softmax}(W_c f_k + b_c)$, where W_c and b_c are the weights and biases, and f_k is the feature vector at scale k.
- **Bounding box offsets:** $\Delta l_k = W_l f_k + b_l$, where W_l and b_l are the weights and biases.

The final bounding box coordinates are computed as:

$$\begin{aligned} x' &= x_{\text{default}} + \Delta x \cdot w_{\text{default}}, \\ y' &= y_{\text{default}} + \Delta y \cdot h_{\text{default}}, \\ w' &= w_{\text{default}} \cdot \exp(\Delta w), \\ h' &= h_{\text{default}} \cdot \exp(\Delta h), \end{aligned} \tag{2}$$

where $(x_{\text{default}}, y_{\text{default}}, w_{\text{default}}, h_{\text{default}})$ are the coordinates and dimensions of the default box, and $(\Delta x, \Delta y, \Delta w, \Delta h)$ are the predicted offsets.

3.5 Receptive Field Blocks (RFB)

In the RFB-SSD variant, the receptive field is enhanced by:

- Using dilated convolutions to capture spatial context.
- Combining features from multiple scales with different receptive fields.

The RFB module can be expressed as:

$$f_{\text{RFB}} = \sum_{i=1}^{n} W_i \cdot f_i, \tag{3}$$

where f_i are feature maps from different scales, and W_i are learnable weights.

3.6 Default Box Design

The default boxes are designed to cover a range of aspect ratios and scales:

- Aspect ratios: $\{1:1, 2:1, 1:2, 3:1, 1:3\}$.
- Scales: Linearly spaced between $s_{\min}$ and $s_{\max}$, where $s_k = s_{\min} + \frac{k-1}{m-1}(s_{\max} - s_{\min})$.

This design ensures comprehensive coverage of object sizes within the image.

3.7 Visualization of Predictions

Figure 2 shows a schematic of the SSD detection process, highlighting the multi-scale predictions and default boxes.

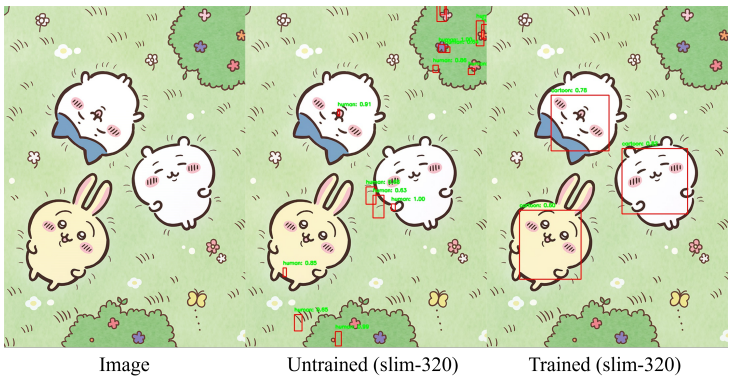

Figure 2: Visualization of SSD Detection Process

4 Dataset Collection and Processing

To train the model, we utilized three datasets. The snapshots for out data used for training is shown is 3:

1. **iCartoonFace Dataset [7]:** A publicly available dataset sourced from `https://paperswithcode.com/dataset/icartoonface`, designed for cartoon face detection tasks. The dataset provides annotated bounding boxes and labels for cartoon faces, making it an ideal choice for fine-tuning the `cartoon` category.
2. **WIDER FACE Dataset [6, 1]:** This dataset, available at `http://shuoyang1213.me/WIDERFACE/`, contains a variety of human face images with bounding box annotations. It was chosen to enhance the model's ability to detect human faces, particularly under challenging conditions such as occlusions.

3. **Custom Cartoon Dataset:** A dataset collected by crawling 360 cartoon meme images (e.g. Joke Bear) from the internet. Using the pre-trained model, we automatically generated bounding box annotations, followed by manual refinement to ensure high-quality labels. After filtering, 150 accurately labeled samples were included, all categorized as `cartoon`.

Figure 3: Visualization of SSD Detection Process

5 Training Process

The training process involved the following steps:

- **Data Augmentation:** Techniques such as random cropping, flipping, and color adjustment were applied using the `TrainAugmentation` class to enhance dataset diversity. These augmentations improved the model's ability to handle variations in image orientation, lighting, and occlusions, increasing robustness against real-world scenarios.
- **Optimization:** The Stochastic Gradient Descent (SGD) optimizer, with a momentum of 0.9 and weight decay of 10^{-4}, was employed. A layer-specific learning rate strategy was adopted: the base network had a lower learning rate to preserve pre-trained features, while the extra layers and prediction heads were trained with higher rates for faster convergence. Training was conducted in two stages—first freezing the base layers, then fine-tuning the entire model.
- **Loss Function:** The Multibox Loss function combines classification loss (softmax cross-entropy) and localization loss (Smooth L1). The overall loss is:

$$L(x, c, l, g) = \frac{1}{N} \left(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g) \right), \tag{4}$$

  where α balances the two losses, N is the number of matched default boxes, and L_{conf} and L_{loc} represent classification and localization losses, respectively.
- **Learning Rate Scheduling:** Two strategies were tested:
  - **Multi-step Scheduler:** Reduces the learning rate at predefined epochs.
  - **Cosine Annealing Scheduler:** Smoothly decreases the learning rate, improving stability during later training stages.

  The cosine annealing scheduler provided better convergence in the final epochs.
- **Model Checkpointing:** Periodic checkpoints saved the model state, optimizer state, and learning rate to allow training resumption or evaluation. The final checkpoint was used for testing.

This process, implemented with PyTorch and multi-GPU support, ensured effective fine-tuning for robust human and cartoon face detection.

Additionally, during the fine-tuning process, specific adjustments were made to handle both cartoon and human face datasets effectively. To address data imbalance, a weighted loss strategy was applied, assigning higher weights to the underrepresented cartoon face class. The base network layers were initially frozen, and only the extra layers and prediction heads were trained. After observing sufficient convergence, the entire model was fine-tuned end-to-end. Performance was evaluated using metrics such

as mean Average Precision (mAP) for detection and classification accuracy for distinguishing between human and cartoon faces.

The training process was implemented using PyTorch and supports multi-GPU training. The training pipeline involves several key steps, including data augmentation, model optimization, loss computation, validation, and model checkpointing. The detailed pipeline is illustrated in Figure 4.

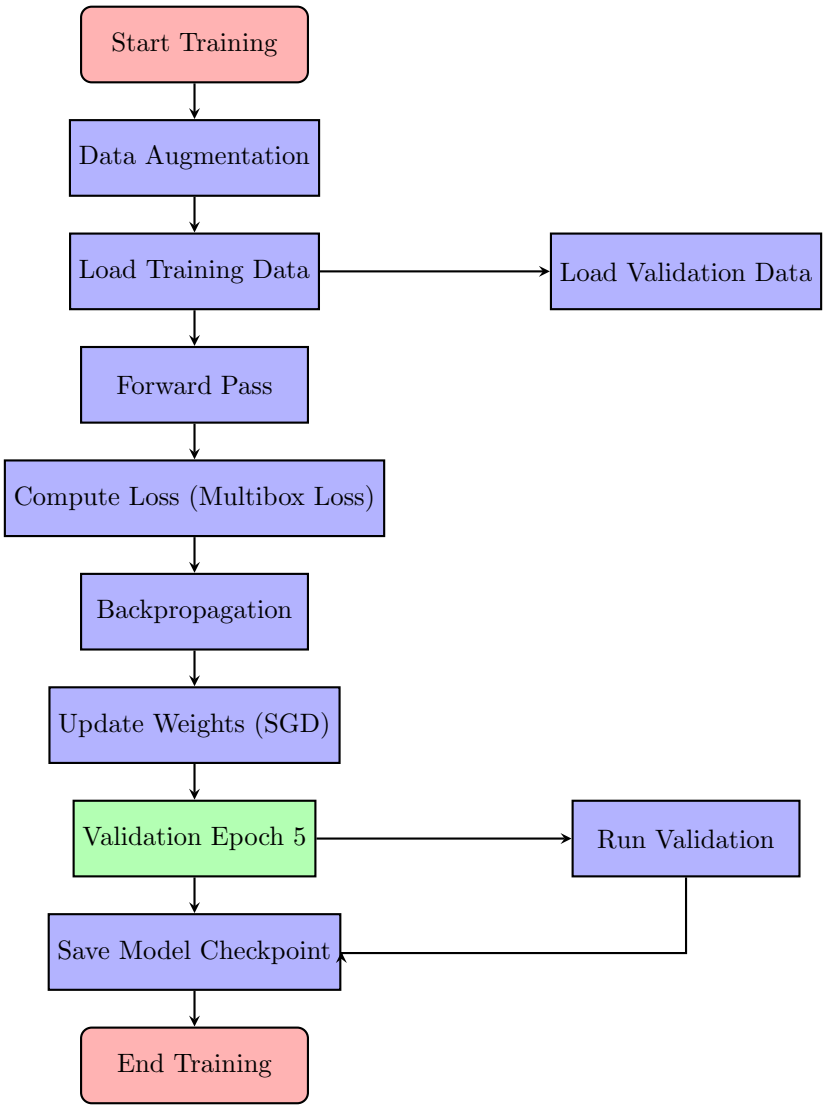

Figure 4: Training Pipeline for the Fine-tuned SSD Model

6 Results and Evaluation

6.1 Accuracy and Frame Rate

The performance of the fine-tuned SSD models on the Jetson Nano is summarized in Table 1. Accuracy is calculated based on the criterion that at least one correctly classified face in an image is sufficient for a positive result.

Table 1: Model Performance on Jetson Nano

Metric	**Slim (320)**	**Slim (640)**	**RFB (320)**	**RFB (640)**
Accuracy (%)	71.5%	83.3%	74.8%	84.4%
Frame Rate (FPS)	25.5	13.7	20.1	12.1
Latency (ms)	39.22	72.99	49.75	82.64

6.2 Pre-trained Models vs Fine-tuned Models Accuracy Comparison

Table 2: Accuracy Comparison Between Pre-trained and Fine-tuned Models

Model Variant	**Pre-trained Accuracy**	**Fine-tuned Accuracy**
RFB (320)	0.202	0.748
Slim (320)	0.222	0.715
RFB (640)	0.243	0.844
Slim (640)	0.243	0.833

7 Challenges and Solutions

During the project, we encountered several challenges and implemented corresponding solutions:

- **Misclassification:** The original model often misclassified objects such as cups as human faces, resulting in poor detection accuracy. This issue was mitigated by fine-tuning the model with additional datasets, specifically targeting the `cartoon` category.
- **Corrupted Dataset Images:** Some images in the two pre-existing datasets were corrupted, causing the training process to encounter `NaN` or `Infinity` losses after reaching 30-40% progress. This was resolved by filtering out the corrupted images during preprocessing.
- **Anti-Crawling Mechanism:** While crawling images for the custom cartoon dataset, the anti-crawling mechanisms of certain websites limited the frequency of requests. This was resolved by reducing the request rate and introducing delays between consecutive requests.
- **Hardware-Specific Challenges:** Deploying the model on the Jetson Nano required significant effort to optimize inference performance while ensuring compatibility. This was resolved by consulting the hardware manuals, experimenting with different configurations, and applying model pruning techniques.
- **Slow Training on Larger Models:** Training larger models, such as RFB with a 640 input size, was slow on the available hardware. The issue was resolved by borrowing a more powerful GPU from team members, significantly accelerating the training process.

References

[1] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5203–5212, 2020.

[2] Andrew G Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[3] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 385–400, 2018.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[5] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[7] Yi Zheng, Yifan Zhao, Mengyuan Ren, He Yan, Xiangju Lu, Junhui Liu, and Jia Li. Cartoon face recognition: A benchmark dataset. In *Proceedings of the 28th ACM international conference on multimedia*, pages 2264–2272, 2020.