Chinese Character Detection

LT2926: Assignment 1 Report

Vadym Kulish (guskulva)

Abstract

The task is to estimate, for each pixel, the probability of belonging to a bounding box that contains a Chinese character. The available server subset was small (< 900 images), so the emphasis was on a robust GPU-friendly pipeline rather than state-of-the-art performance. I implemented two architectures (a small FCN and a U-Net), used weighted BCE (Binary cross-entropy) + Dice loss to manage class imbalance, and evaluated with Accuracy and IoU (Intersection over Union) across thresholds. Full-resolution training is made feasible via random positive-biased crops. A 5-epoch U-Net run achieved a best validation IoU of 0.185 at threshold 0.80.

1 Data & Preparation

I used the CTW metadata (polygons and adjusted boxes) and verified parsing by rendering overlays: positive and ignore pixels looked correctly aligned. I split data into a standard **80/10/10** (train/val/test) to keep as much data for training as possible given the small subset.

Initial experiments used downsampled images to 512×512 for speed, but upon inspection of produced images, the characters became indistinguishable. I then trained on 2048×2048 images using random 512×512 crops with a minimum positive-pixel constraint (50px), which both reduces memory usage and ensures the model actually sees text instead of learning on noise.

2 Models

FCN (baseline). A lightweight fully convolutional network with stride/downsampling and upsampling back to the input size. Simple, fast to iterate, good for a baseline.

U-Net. Encoder–decoder with skip connections, widely used for segmentation with limited data because it preserves spatial detail. Still small enough for rapid training on GTX 1080Ti.

I considered heavier options (e.g., DeepLab, ResNet encoders, ViT) but kept these two for *faster/easier development, easier debugging, and reasonable performance* on a small dataset.

3 Loss & Optimization

Pixel imbalance is extreme (positives are sparse). I used a composite loss:

$$\mathcal{L} = \mathrm{BCE}_{\mathrm{weighted}} + \lambda \cdot \mathrm{Dice},$$

where $\mathrm{BCE}_{\mathrm{weighted}}$ applies a positive weight (e.g., `pos_weight=20`) and Dice encourages overlap. Optimizer: Adam with LR `1e-4`. To stabilize training, I used mini-batch random crops biased toward positives.

4 Evaluation

Accuracy can be misleading under imbalance (predicting "all background" yields high Acc). **IoU** is more informative. I sweep probability thresholds $t \in \{0.1, \ldots, 0.9\}$ where a pixel is classified positive if $\sigma(\mathrm{logit}) \geq t$.

Threshold meaning. The model outputs a probability map (after sigmoid). The threshold t converts probabilities to a binary mask: higher t favors precision (fewer false positives), lower t favors recall (fewer misses). I report IoU across thresholds and save the best-IoU checkpoint with its threshold.

5 Training Setup

All results (the graph and the images) here use **U-Net** trained for **5 epochs** on the mltgpu server. Full-res images with 512×512 crops, batch size 6, `pos_weight=20`, Dice weight 0.5, LR $= 10^{-4}$. Workers $= 4$ for I/O.

FCN was trained using the same hyperparameters except `pos_weight=60`. This increase was motivated by a lack of epoch-to-epoch progress (IoU) on val subset at `pos_weight=60`. Regardless, further parameter exploration could improve results with both models.

Split	80/10/10 (train/val/test)
Crop	512×512 from 2048×2048
Pos bias	Min positive pixels per crop
Loss	Weighted BCE + λ Dice (λ=0.5)
Optimizer	Adam (LR = 10^{-4})
Batch size	6
Hardware	1× GTX 1080Ti (mltgpu)

6 Results

U-Net (5 epochs). Best validation **IoU = 0.185** at **threshold 0.80** (val Acc ≈ 0.98 at that threshold). Loss decreased from ∼1.10 (epoch 1) to ∼0.75 (epoch 5). Given the trend, more epochs could improve results further, but time constraints limited training.

FCN (5 epochs). Trained with the same patch-based pipeline but a stronger positive weighting to counter sparsity (`pos_weight=60`). The average training loss decreased from ∼1.64 (epoch 1) to ∼1.50 (epoch 5). On validation, the best performance was **IoU ≈ 0.049** at **threshold 0.50** (val Acc ≈ 0.96). Overall, FCN was markedly weaker than U-Net for the same budget, which is expected given the absence of skip connections that preserve fine spatial detail needed for small characters.

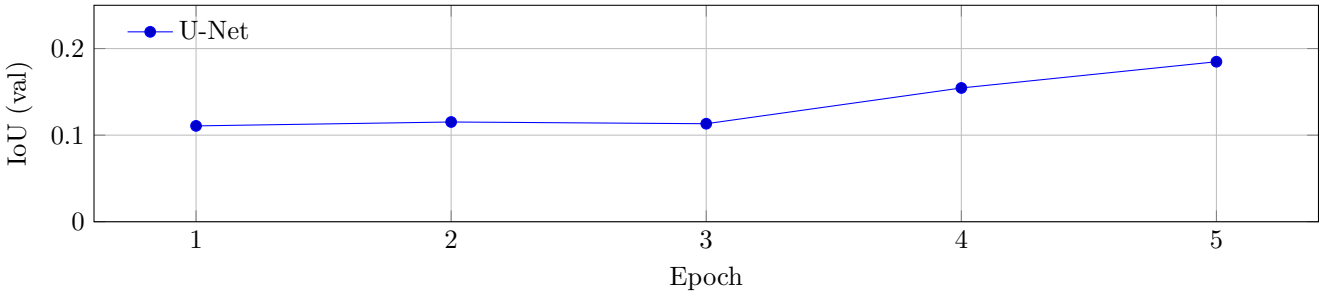
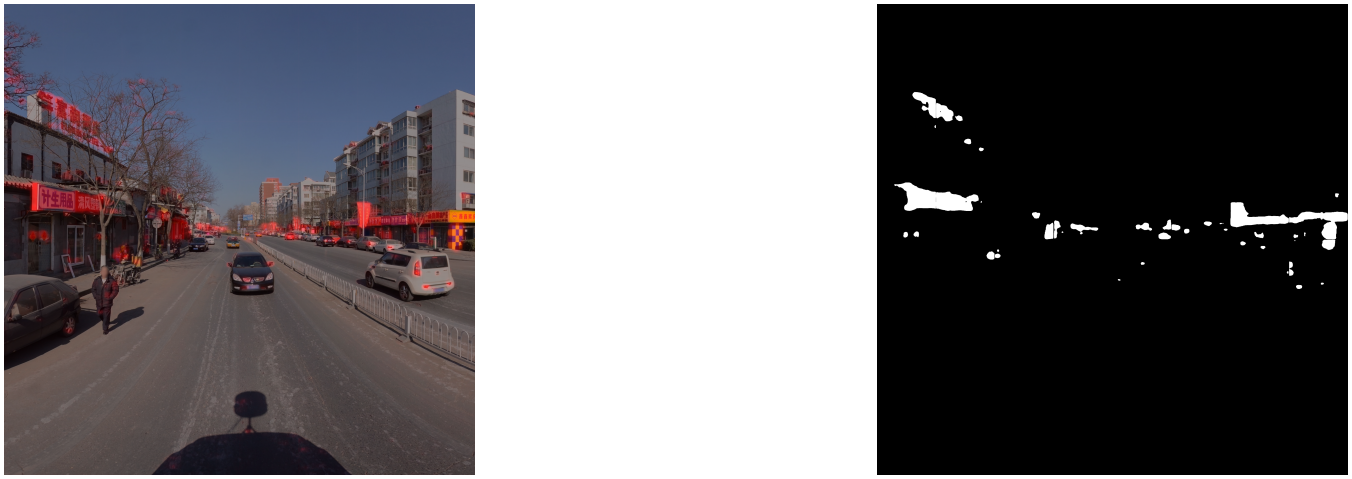


Figure 1: Validation IoU over epochs (best threshold per epoch).

Qualitative. Figure 2 shows example overlays: red heatmap for predicted probabilities and the thresholded mask (U-Net).



(a) Prob. heatmap

(b) Binary mask (t=0.80)

Figure 2: Qualitative predictions (U-Net).

7 Discussion

Data constraints. With < 900 images, strong generalization is unlikely. Nonetheless, the U-Net learned a useful signal (rising IoU). Patches from full resolution were essential; downsampling to 512×512 removed detail.

Additional ideas (future work):

- More extensive training & calibration: longer runs with early stopping on val IoU; iterate over different `pos_weight`, Dice weight, and minimum positive pixels; calibrate thresholds from PR curves.
- Box normalization for OCR: rectify/straighten candidate boxes (perspective correction) and feed crops to a potential character classifier.
- Input variants: grayscale training or per-channel inference; automatic contrast and simple per-channel selection/combination (e.g., max over RGB probabilities) to handle engravings or raised text.

AI Usage Disclaimer

I used an AI assistant (ChatGPT) in a responsible manner for: help with syntax issues, comparing model and loss options, managing the growing number of hyperarguments (with argument parser), and figuring out multi-worker/GPU usage on mltgpu. Final implementation and modeling decisions (splits, patching, architectures, evaluation) were mine.

Reproducibility

Code: Python/PyTorch. Run `python main.py make-splits --size 512`, then `train.py`. Visualizations via `viz.py` (sliding window). *(Exact terminal commands used are included in* ***`train.py`****)*