CPSC 542: Assignment #2
Implement a Semantic Segmentation Neural Network

LEWIS, T.
TYLEWIS@CHAPMAN.EDU
ID# 002366930
03/19/24

Problem Statement

In my previous assignment, I used a convolutional architecture to classify handwritten Japanese characters. The problem comes when trying to find a real world application of such model. In practice, characters would more than likely be hidden within a more complex image, whether on a road sign being identified by a self driving vehicle, an automatic traffic camera reading license plates, transcribing physical texts from images, etc.

A potential solution for these problems may be achieved with the use of segmentation analysis. Isolating the character from background noise will allow for more accurate classification.

Using a Deep NN solution for this problem is particularly suitable. A convolutional neural network is strong at identifying spatial relationships in images and provides toward an effective image character segmentation model.

Dataset and Preprocessing

The assignment is to utilize aspects of the previous classification assignment to achieve segmentation. My greyscale dataset used for the classification task would truthfully be rather boring on its own for segmentation, so I adapted an idea from an online blog to generate a custom dataset by overlaying characters over images from the CIFAR database.

In my pre-processing approach, the handwritten character dataset is downsampled to be roughly half the size of a CIFAR image, and is then overlaid on a random location. The color channels of the characters are slightly randomized which introduces variability into the RGB channels.

This method allows for both the test image and valid segmentation map to be both generated at runtime.

CIFAR 32x32 RGB images
K-MNIST 28x28 BW images
-> downsampled to 14x14

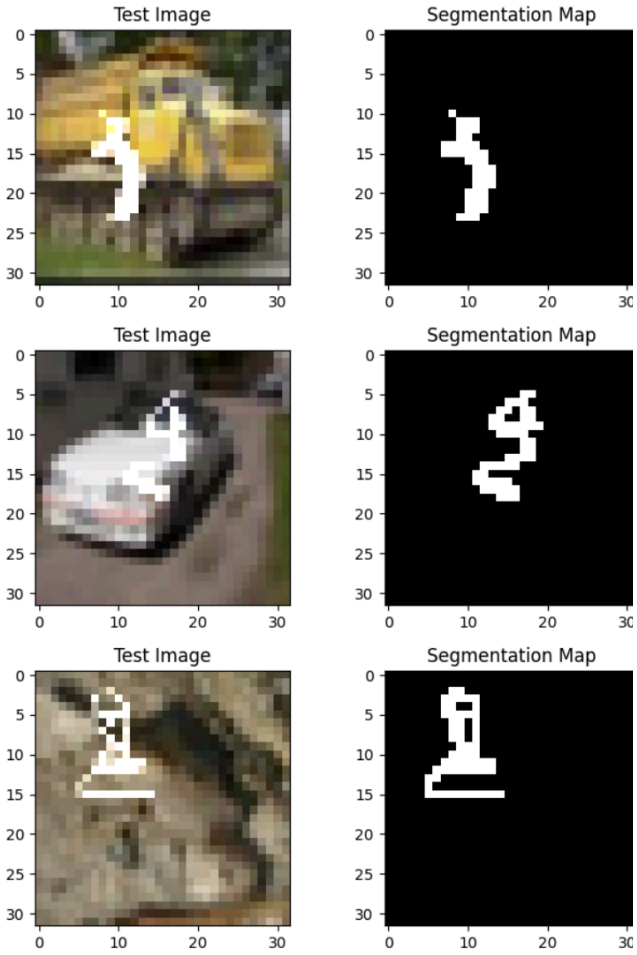


Figure 1: The generated dataset for training and testing

Methods

Model

The model utilizes the following NN components to achieve semantic segmentation:

- **Five Convolutional Layers:** These layers are responsible for capturing the handwritten character patterns. My arrangement of neurons per layer was 3(in), 12, 10, 8, 4, 1(out). Each convolutional layer uses stride = 1, and K = 5.
- **ReLU Activation:** Following the first three convolutional layers, a Rectified Linear Unit (ReLU) activation function is applied. It works by keeping positive values unchanged while setting negative values to zero. It's used here to add non-linear properties to the model and help combat the vanishing gradient problem.
- **Dropout:** Applied after each ReLU activation for the first three convolutional layers, dropout is a regularization technique used to prevent overfitting. During training, it randomly sets a fraction of input units to 0 at each update during training time, which helps to make the model more robust and less likely to rely on any small set of neurons.
- **Sigmoid Activation:** After the last convolutional layer, a sigmoid activation function is applied. Unlike ReLU, the sigmoid function maps the input values into a range between 0 and 1, making it suitable for binary classification tasks. In the context of segmentation, each pixel is classified as belonging to the foreground or background.

Hyper/parameters

- **Learning Rate (INIT_LR = 0.001):** Determines the initial step size for updating the model's weights during training. A value of 0.001 is chosen to balance convergence speed and stability.
- **Dropout Rate (DROPOUT = 20%):** Applied to reduce overfitting by randomly omitting 20% of the neurons in the network during training, encouraging the model to learn more generalizable features.
- **Batch Size (BATCH_SIZE = 64):** The number of training samples processed before the model's internal parameters are updated. A batch size of 64 is used to leverage computational efficiency while maintaining a degree of stochastic training behavior.
- **Epochs (EPOCHS = 10):** Specifies the total number of complete passes through the training dataset. The model is trained for 10 epochs to ensure adequate learning without overfitting.

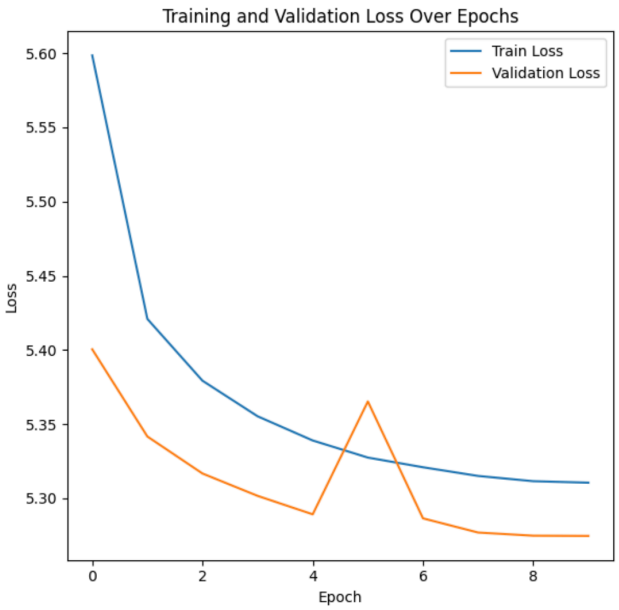
Evaluation

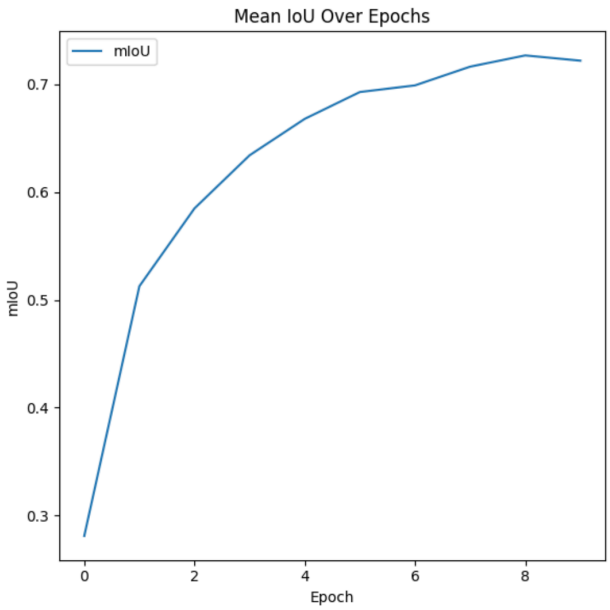
Accuracy was evaluated using Mean Intersection over Union (mIoU)

The mIoU is a metric for evaluating the performance of models on tasks such as segmentation. It measures the overlap between the predicted segmentation and the ground truth, normalized by the union of these two areas. For each class, the Intersection over Union (IoU) is calculated by dividing the area of overlap between the predicted segmentation and the ground truth by the area of their union. The "mean" in mIoU refers to the average of these IoU scores across all classes.

Results

Training Metrics:

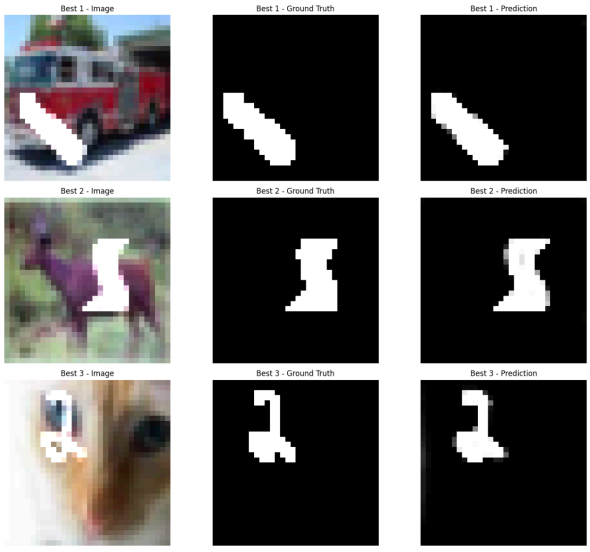




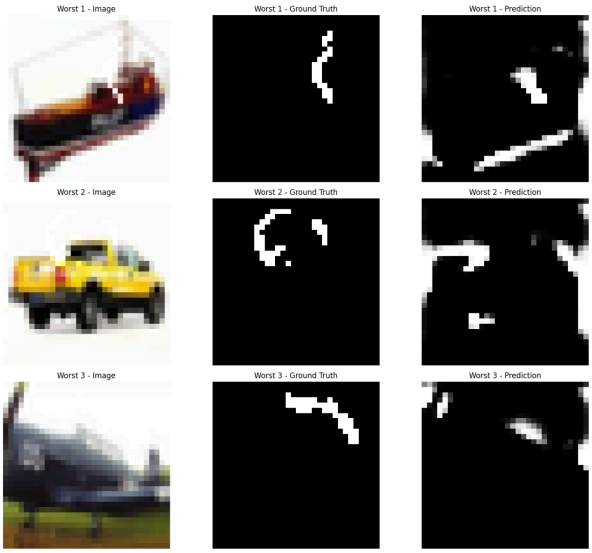
Epoch	Train Loss	Validation Loss	mIoU
1	5.639282	5.485751	0.2550
2	5.459481	5.382293	0.4583
3	5.412807	5.386549	0.5254
4	5.383757	5.310014	0.5695
5	5.361785	5.294540	0.6068
6	5.343482	5.312945	0.6374
7	5.331896	5.300787	0.6573
8	5.319019	5.275516	0.6878
9	5.314334	5.277806	0.6978
10	5.311616	5.273154	0.7046

3 best / 3 worst examples & occlusion prediction map generated from testing data:

Best



Worst



Report - Discussion/Results

In the evaluation of the segmentation model's performance, I primarily focused on two metrics: the mean Intersection over Union (mIoU) and the loss values (both training and validation). The model presented a training mIoU of 0.7046, with training and validation losses at 5.329562 and 5.278884, respectively.

The loss did not encounter significant improvement despite the improvement of mIoU, I anticipate this is because the model is becoming better at making more precise segmentations where it matters most for the mIoU metric, focusing on correctly predicting the boundaries and areas of interest within the images. However, the overall loss calculation, which is more sensitive to the sum of the errors across the entire image, may not reflect this improvement if there are still numerous small errors distributed throughout the image. This scenario can occur when the model optimizes for correctly classifying pixels that contribute significantly to the intersection over union for each class, thereby improving mIoU, without a corresponding large decrease in the overall pixel-wise loss. Additionally, it's possible that the loss function used may not be perfectly aligned with the mIoU metric, meaning improvements in one do not necessarily result in proportional improvements in the other.

The biggest issue was with images with high light color pixel content, as these interfered with the light colored nature of the overlapped characters.

Performance and Parameter Adjustments

Throughout the development process, I experimented with various model parameters and architectural changes to improve these metrics. Notably, adjustments included:

- Model Depth: Increasing the number of convolutional layers and filters allowed the model to capture more complex features, improving mIoU slightly.
- Dropout Rate: Incorporating the dropout helped in preventing overfitting, which was evident when comparing training and validation losses during development.
- Optimizer Selection: Switching between optimizers (from Adam to SGD with momentum) showed a variance in how quickly and effectively the model could minimize the loss function. Ended using Adam.

These adjustments led to incremental improvements in the model's performance. However, the extent of improvement indicated that further optimizations and explorations are warranted.

Future Enhancement

For future enhancements, several avenues are worth exploring:

- Implementing U-Net with skip connections.
- Hyperparameter Tuning: Could unearth a more optimal set of parameters.
- Data Augmentation: More aggressive data augmentation strategies could help the model generalize better to unseen data, improving both mIoU and loss metrics on the validation set.

Evaluation and Recommendation for Production Use

Considering the current state of the model, its performance is promising but not yet optimal for production use, especially in applications where high precision is critical. The mIoU of ~0.7, while decent, suggests there is room for improvement in how accurately the model can segment images. Production readiness requires a model to not only exhibit high accuracy but also to demonstrate robustness across diverse and challenging datasets.