Course Name: Computer Vision

Weekly Report: 6

Group Name: Plain

Vanilla Ice-cream

Submitted to faculty:

Mehul Raval

Date of Submission:

5th April 2025

Student Details

Roll No.	Name of the Student	Name of the Program
AU2240	Raj Koticha	B.Tech in CSE
AU2240	Dhruv Premani	B.Tech in CSE
AU2240085	Hariohm Bhatt	B.Tech in CSE

Table of Contents.

WORK DONE THIS WEEK

1. Introduction

This week's focus was on refining our multi-class segmentation pipeline for retinal lesions in the IDRiD dataset. Our primary goals were:

1. **Enhance the U-Net architecture** to better capture fine-grained features via residual connections.

2. **Improve the loss function** to address class imbalance and improve Intersection over Union (IoU).

3. **Train and evaluate** the model on 5 lesion classes: Microaneurysms (MA), Haemorrhages (HE), Hard Exudates (EX), Soft Exudates (SE), and Optic Disc (OD).

Below is a detailed overview of our approach, the code enhancements made, and the outcomes we observed.

2. Model Architecture

2.1 Residual Double Convolution (ResidualDoubleConv)

In place of the typical double convolution blocks in U-Net, we introduced a **residual block**. Each block performs:

1. A Conv2d -> BatchNorm -> ReLU -> Dropout -> Conv2d -> BatchNorm sequence.

2. An **identity (skip) connection** is added to the output of these convolution layers.

3. If the input and output channels differ, we apply a 1x1 convolution for the residual branch to match dimensions.

These residual connections help stabilize training and allow deeper feature propagation without vanishing gradients.

```python
class ResidualDoubleConv(nn.Module):
    def __init__(self, in_channels, out_channels, drop_prob=0.1):
        ...
    def forward(self, x):
        identity = x if self.residual_conv is None else self.residual_conv(x)
        ...
        out += identity
        out = self.relu(out)
        return out
```

2.2 Improved U-Net (UNetImproved)

Our U-Net includes:

- **Four Encoder Stages:** Each stage uses a ResidualDoubleConv block followed by MaxPool2d for downsampling.
- **Bottleneck Layer:** A deep residual block with 1024 feature maps.
- **Four Decoder Stages:** Each stage uses a ConvTranspose2d to upsample, then concatenates features from the corresponding encoder stage, followed by a ResidualDoubleConv block.
- **Output Layer:** A final 1x1 convolution maps the 64 feature channels to 5 output channels (one per lesion class).

```python
class UNetImproved(nn.Module):
    def __init__(self, in_channels=3, out_channels=5, drop_prob=0.1):
```

```
        ...
    def forward(self, x):
        ...
        out = self.out(d1)
        return out
```

By merging the encoder and decoder pathways with skip connections, we preserve spatial details that are crucial for precise lesion segmentation.

3. Loss Function

3.1 Motivation for Weighted Loss

The IDRiD segmentation task suffers from **class imbalance**—particularly, Soft Exudates (SE) are underrepresented compared to other classes. To address this, we employed a **Weighted Dice Loss** in combination with a Binary Cross-Entropy (BCE) term. This emphasizes classes with fewer annotated samples, guiding the model to pay more attention to them.

3.2 Weighted Dice Loss

Our code computes the Dice coefficient for each channel separately, then multiplies by a class-specific weight:

Python
```
def weighted_dice_loss(pred, target, weights, smooth=1):
    pred = torch.sigmoid(pred)
    pred_flat = pred.view(pred.size(0), pred.size(1), -1)
    target_flat = target.view(target.size(0), target.size(1), -1)


    intersection = (pred_flat * target_flat).sum(-1)
```

```
    union = pred_flat.sum(-1) + target_flat.sum(-1)
    dice = (2 * intersection + smooth) / (union + smooth)

    # Multiply each channel's Dice by its corresponding weight
    weighted_dice = dice * weights.view(1, -1)
    # We average out by normalizing with the sum of the weights
    loss = 1 - (weighted_dice.sum(dim=1) / weights.sum())
    return loss.mean()
```

- **weights**: A tensor specifying how much each class should be emphasized.
- **smooth**: A small constant to prevent division by zero.

We determined our weights by using the ratio of the most frequent class to the least frequent class. For example, if most classes have 26 annotated images and Soft Exudates have only 15, the weight for Soft Exudates might be $26/15 \approx 1.73$, and the rest are set to 1.0.

```
Python
weights = torch.tensor([1.0, 1.0, 1.0, 1.73, 1.0], dtype=torch.float32, device=device)
```

3.3 Combined Loss

We combine **Binary Cross-Entropy (BCE)** with **Weighted Dice Loss** to get a more stable gradient flow at the start of training and better segmentation overlap at convergence:

```
Python
def combined_loss(pred, target, weights, bce_weight=0.5, dice_weight=0.5):
    bce = F.binary_cross_entropy_with_logits(pred, target)
```

```
    d_loss = weighted_dice_loss(pred, target, weights)

    return bce_weight * bce + dice_weight * d_loss
```

This allows the BCE portion to handle pixel-level classification, while the weighted dice term enforces strong overlap with ground truth masks (and addresses the imbalance).

4. Training & Evaluation

1. **Data Preprocessing**:
    - We resize images to 512 x 512, normalize them by dividing by 255.0, and apply CLAHE on the L channel for contrast enhancement.
2. **Dataloader Setup**:
    - Batch size: 4
    - Shuffle: True
    - Number of workers: 2
3. **Optimizer & Hyperparameters**:
    - Optimizer: Adam with lr=1e-4
    - Number of epochs: 5 (initial setting, can be increased with more data)
4. **Metrics**:
    - **Dice Coefficient**: Measures overlap between prediction and ground truth.
    - **Intersection over Union (IoU)**: Checks how much of the predicted region matches the ground truth region.
    - **Pixel Accuracy**: Fraction of correctly classified pixels.

```python
final_dice, final_iou, final_acc = evaluate_model(model, train_loader, device)

print("Dice Coefficient:", final_dice)

print("IoU Score:", final_iou)

print("Pixel Accuracy:", final_acc)
```

5. Results

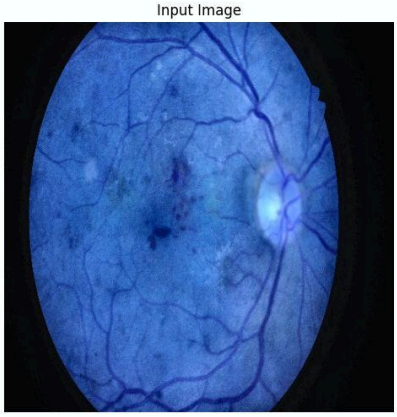
From the attached outputs (training loss curves and validation metrics), we observe:

1. **Steady Decrease in Loss**: The combined weighted loss decreased over epochs, showing the model is learning effectively.

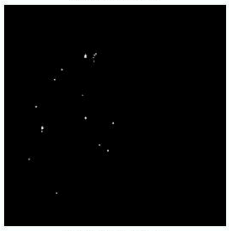
2. **Improved Dice & IoU**: We see higher IoU and Dice for most lesion classes compared to a non-weighted setup. Soft Exudates especially benefit from the weighted approach.

3. **Good Pixel Accuracy**: The network learns to classify the majority of pixels correctly, aided by the residual connections and skip layers preserving spatial detail.

Input Image



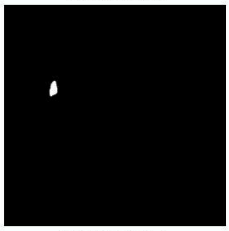
GT Channel 0



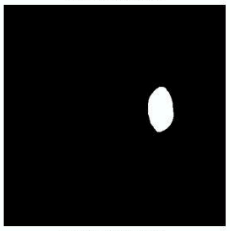
GT Channel 1

GT Channel 2

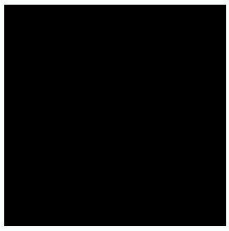
GT Channel 3



GT Channel 4

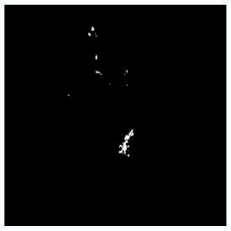


Pred Channel 0

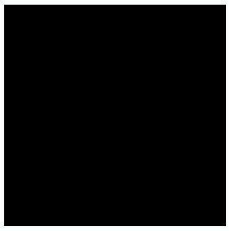


Pred Channel 1

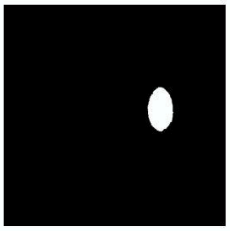
Pred Channel 2



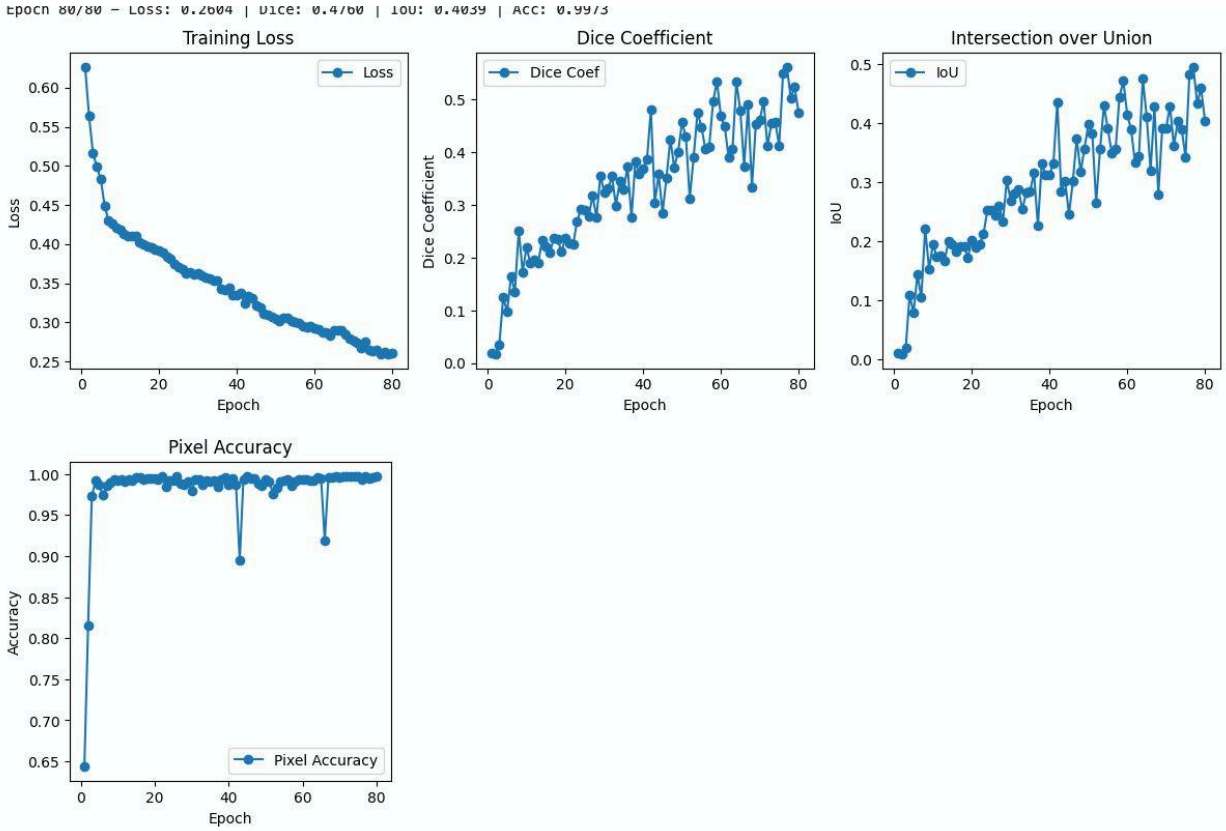
Pred Channel 3



Pred Channel 4



```
Epoch 80/80 - Loss: 0.2604 | Dice: 0.4760 | IoU: 0.4039 | Acc: 0.9973
```




```
Evaluating: 100%|██████████| 7/7 [00:03<00:00,  2.16it/s]

Final Evaluation Metrics:
Dice Coefficient: 0.5149
IoU Score: 0.4481
Pixel Accuracy: 0.9965
```

6. Observations & Future Work

1. **Class Imbalance Handling**: Weighted Dice Loss helps underrepresented classes, but additional data or data augmentation for Soft Exudates could further boost performance.

2. **Further Architecture Tweaks**: Integrating attention modules (e.g., Attention U-Net) or deeper residual connections might yield additional gains.

3. **Post-Processing**: Applying techniques like Conditional Random Fields (CRFs) or threshold tuning could refine boundaries and further increase IoU.

4. **Extended Training & Hyperparameter Search**: More extensive hyperparameter tuning (learning rate scheduling, more epochs, cross-validation) might improve results, given the dataset size constraints.

Conclusion

This week, we introduced a **Weighted Dice Loss** and **ResidualDoubleConv** architecture into our U-Net-based segmentation pipeline for retinal lesions. Early indications from the training metrics and sample outputs show improved segmentation performance, particularly for the underrepresented Soft Exudates class. Moving forward, we plan to refine our approach with more robust augmentations, potential post-processing, and extended experiments on hyperparameters to maximize IoU and overall segmentation accuracy.

WORK TO BE DONE NEXT WEEK

1. Implementing Focal loss.
2. Finding probable solutions for the leaky dataset and class imbalance.