Solving Pixel-level Semantic Segmentation with Fully Convolutional Network

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

In this study, we address the challenge of semantic segmentation using the PASCAL VOC-2007 dataset. Given the complexity of realistic scene representation, where images often contain multiple objects from various classes, our approach leverages deep convolutional neural networks (CNNs) to classify each pixel into one of 21 categories, including the background. Initial experiments with naive models demonstrated a tendency to achieve high pixel accuracy by predominantly classifying pixels as background, a result of the class imbalance inherent in the dataset. However, these models exhibited low Intersection-Over-Union (IoU) scores, underscoring their limited capability in distinguishing between different object classes. To address these limitations, we implemented advanced models that, despite sometimes lower pixel accuracy, achieved significantly higher IoU scores. Through visual inspection, these models showed an enhanced ability to differentiate objects from the background, albeit with occasional inaccuracies in precise class identification. This abstract summarizes our exploration into optimizing semantic segmentation performance, highlighting the trade-offs between pixel accuracy and IoU in models ranging from baseline to advanced CNN architectures.

1 Introduction

Semantic segmentation stands at the forefront of computer vision tasks, aiming to understand images at a pixel level by classifying each pixel into predefined categories. This task is fundamental for numerous applications, including autonomous driving, medical image analysis, and scene understanding, where the precise delineation of objects within their environments is crucial. Deep convolutional neural networks (CNNs) have emerged as a powerful tool in this domain, leveraging large datasets and GPU computing to learn complex representations of visual data.

The PASCAL Visual Object Classes Challenge 2007 (VOC2007) dataset serves as the benchmark for our exploration into semantic segmentation. It comprises 9,963 images, annotated with 24,640 objects across 20 different categories, plus an additional background category. This dataset is uniquely challenging due to its realistic scene composition, featuring multiple objects of various classes within a single image, often against complex backgrounds. Such diversity necessitates models that can not only recognize objects but also accurately delineate their boundaries at the pixel level.

In our study, we apply Xavier weight initialization and batch normalization to all models. Xavier weight initialization, also known as Glorot initialization, is designed to address the problem of vanishing and exploding gradients in deep neural networks. It aims to keep the variance of the outputs of each layer similar to the variance of its inputs, helping to stabilize the learning process across deep architectures. Batch normalization is a technique used to improve the speed, performance, and stability of artificial neural networks. It normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

2 Related Work

In the evolution of semantic segmentation, Fully Convolutional Networks (FCNs) have been pivotal, setting a foundation for models like ResNet and U-Net.

ResNet 6 introduced a novel approach to deep learning with its residual blocks, allowing for the training of much deeper networks by addressing the vanishing gradient problem. This architecture has been widely adopted in semantic segmentation tasks, serving as a backbone for various models due to its ability to capture rich, contextual information necessary for accurate pixel-wise classification.

U-Net 6 , originally designed for biomedical image segmentation, features a symmetric encoder-decoder structure with skip connections, facilitating precise localization and detailed segmentation from limited data. Its effectiveness extends beyond medical images to general scene segmentation, exemplified by its application in addressing the complexities of the PASCAL VOC-2007 dataset in our study.

By leveraging the strengths of ResNet and U-Net, our research aims to enhance semantic segmentation accuracy, particularly in challenging scenarios characterized by class imbalance and intricate scene compositions.

3 Method

3.1 Baseline Model

We designed Fully Convolutional Network (FCN) for semantic segmentation tasks. Out baseline model is structured into two main parts: an encoder and a decoder. The encoder consists of 5 convolutional layers that progressively downsample the input image to capture feature representations at multiple levels of abstraction. Each layer has kernel size of 3, stride of 2, padding of 1 and dilation of 1. The last layer in the encoder part has 512 output channels. The decoder part comprises 5 transposed convolutional (also known as deconvolutional) layers that upsample these features to construct a pixel-wise prediction map corresponding to the original image dimensions. Each layer has kernel size of 3, stride of 2, padding of 1, dilation of 1 and output padding of 1. The last layer in the decoder part has 32 output channels. For all layers in encoder and decoder we use ReLU as the activation function. After each layer we apply batch normalization again to ensure there are negative numbers in features. The last layer in the baseline model, also the 11th layer, we apply a 2D convolution with kernel size of 1 which takes 32 input channels and outputs 21 channels, mapping 21 classes. In the training part, we use Xavier weight initialization and unweighted cross entropy loss. We set 20 epochs, 0.001 constant learning rate and batch size of 16. In each epoch, we track the pixel-accuracy and intersection-over-union (IoU) of the validation set. We use Adam optimizer as our gradient descent optimizer. Also, in our whole investigation process, we enable GPU computation to accelerate our training process, when cuda is available.

3.2 Improvement

Model 4a is a modified version of baseline model. We add 10 more epochs and decrease the learning rate to 0.0005. We also add a cosine annealing learning rate scheduler with minimum learning rate of 0.0001, so that learning rate changes among epochs

Model 4b is a modified version of the baseline model. Since adding a learning rate scheduler doesn't improve the performance, we are not adding one here. Since there is an r imbalanced class problem, which background dominates 70% among 21 classes. So here we decide to add a weight loss function to encourage our model to predict the pixel other than background. We use 2 minus the inverse number of occurrences of each class in the training set. For convenience, we mutually set the weight of the background category to 0.15. Now false negatives of the background get less penalty, and false negatives of other classes get almost double the penalty.

Model 4c is a modified version of model 4b. Beside batch normalization, we additionally apply a horizontal flip to every picture and target.

For both model 4b and 4c, we use 50 epochs and 0.0005 learning rate.

3.3 Experimentation

For all three models in this part below we will use a weighted cross entropy loss function to solve the imbalance class problem. The weight tensor is a ones tensor with length 21, except the weight for background class is 0.1. We also save the model at a certain epoch when IoU on validation set is currently the highest, and load the best model after all epochs. With this strategy, our model won't overfit the training set. We use Xavier weight initialization and Adam gradient descent optimizer. We are not using any learning rate scheduler here.

Model 5a is a modified version of the baseline model. We add one layer each to encoder and decoder. We change kernel size and dilation of some layers. We also use Tanh as our activation function. During training, we use 50 epochs. The architecture of Model 5a is shown in the table below.

Table 1: Model 5a Architecture

Layer#	in-channel/out-channel	Kernel size/stride/ padding/dilation/ (output_padding)	Activation function/other
Encoder 1 (Conv2d)	21/32	3/1/1/1	Tanh/batchNorm
Encoder 2 (Conv2d)	32/32	3/1/1/1	Tanh/batchNorm
Encoder 3 (Conv2d)	32/64	3/1/1/1	Tanh/batchNorm
Encoder 4 (Conv2d)	64/128	3/2/1/2	Tanh/batchNorm
Encoder 5 (Conv2d)	128/256	5/2/1/2	Tanh/batchNorm
Encoder 6 (Conv2d)	256/512	5/2/1/2	Tanh/batchNorm
Decoder 1	512/512	5/2/1/2/0	Tanh/batchNorm
Decoder 2 (ConvTranspose2d)	512/256	5/2/1/2/0	Tanh/batchNorm
Decoder 3 (ConvTranspose2d)	256/128	3/2/1/2/1	Tanh/batchNorm
Decoder 4 (ConvTranspose2d)	128/64	3/1/1/1/0	Tanh/batchNorm
Decoder 5 (ConvTranspose2d)	64/32	3/1/1/1/0	Tanh/batchNorm
Decoder 6 (ConvTranspose2d)	32/32	3/1/1/1/0	Tanh/batchNorm
Classifier (Conv2d)	32/21	Kernel size=1	N/A

Model 5b is a modified version of the baseline model. We apply transfer learning on this model. We replace the encoder part with a trained model, in our study we use Resnet34 in pytorch library. In the training process, we use 50 epochs and learning of 0.0005. Notice that the Resnet34 model uses similar architecture as our encoder. Resnet34 contains a series layer of 2D convolution and max pool layer. It also uses ReLU activation function, but uses different kernel size, stride and padding in Conv2d. The architecture of Model 5b is shown in the table below.

Table 2: Model 5b Architecture

Layer#	in-channel/out-channel	Kernel size/stride/ padding/dilation/ (output_padding)	Activation function/other
Pre-trained Resnet34	3/512	N/A	Batch normalization after layer
Decoder 1 (ConvTranspose2d)	512/512	3/2/1/1/1	ReLU/batchNorm
Decoder 2 (ConvTranspose2d)	512/256	3/2/1/1/1	ReLU/batchNorm
Decoder 3 (ConvTranspose2d)	256/128	3/2/1/1/1	ReLU/batchNorm
Decoder 4 (ConvTranspose2d)	128/64	3/2/1/1/1	ReLU/batchNorm
Decoder 5 (ConvTranspose2d)	64/32	3/2/1/1/1	ReLU/batchNorm
Classifier (Conv2d)	32/21	Kernel size=1	N/A

Model 5c uses U-Net architecture, which is completely different from our baseline model. It also contains encoder and decoder, but with more max pool layers. The training process is similar to the two models above, it uses 50 epochs and a learning rate of 0.0001, with similar weighted loss and regularization techniques. The architecture of Model 5c is shown in the table 3 at the end of document.

Table 3: Model 5c Architecture

Layer#	in-channel/out-channel	Kernel size/stride/ padding/dilation/ (output_padding)	Activation function/other
Encoder 1 (Conv2d)	21/64	3/1/0/1	ReLU/batchNorm
Encoder 2 (Conv2d)	64/64	3/1/0/1	ReLU/batchNorm
Encoder 3 (Max pool)	64/64	2/2/0/1	ReLU/batchNorm
Encoder 4 (Conv2d)	64/128	3/1/0/1	ReLU/batchNorm
Encoder 5 (Conv2d)	128/128	3/1/0/1	ReLU/batchNorm
Encoder 6 (Max pool)	128/128	2/2/0/1	ReLU/batchNorm
Encoder 7 (Conv2d)	128/256	3/1/0/1	ReLU/batchNorm
Encoder 8 (Conv2d)	256/256	3/1/0/1	ReLU/batchNorm
Encoder 9 (Max pool)	256/256	2/2/0/1	ReLU/batchNorm
Encoder 10 (Conv2d)	256/512	3/1/0/1	ReLU/batchNorm
Encoder 11 (Conv2d)	512/512	3/1/0/1	ReLU/batchNorm
Encoder 12 (Max pool)	512/512	2/2/0/1	ReLU/batchNorm
Encoder 13 (Conv2d)	512/1024	3/1/0/1	ReLU/batchNorm
Encoder 14 (Conv2d)	1024/1024	3/1/0/1	ReLU/batchNorm
Decoder 1 (ConvTranspose2d)	1024/1024	2/2/0/1/0	ReLU/batchNorm
Decoder 2 (Conv2d)	1024/512	3/1/2/1	ReLU/batchNorm
Decoder 3 (Conv2d)	512/512	3/1/2/1	ReLU/batchNorm
Decoder 4 (ConvTranspose2d)	512/512	2/2/0/1/1	ReLU/batchNorm
Decoder 5 (Conv2d)	512/256	3/1/2/1	ReLU/batchNorm
Decoder 6 (Conv2d)	256/256	3/1/2/1	ReLU/batchNorm
Decoder 7 (ConvTranspose2d)	256/256	2/2/0/1/0	ReLU/batchNorm
Decoder 8 (Conv2d)	256/128	3/1/2/1	ReLU/batchNorm
Decoder 9 (Conv2d)	128/128	3/1/2/1	ReLU/batchNorm
Decoder 10 (ConvTranspose2d)	128/128	2/2/0/1/0	ReLU/batchNorm
Decoder 11 (Conv2d)	128/64	3/1/2/1	ReLU/batchNorm
Decoder 12 (Conv2d)	64/64	3/1/2/1	ReLU/batchNorm
Classifier (Conv2d)	64/21	Kernel size=1	N/A

4 Result

This section contains figures of training loss curve, validation loss curve, validation IoU curve and validation pixel accuracy curve for each model. Mask visualization on one image predicted by each model, and mask visualization on image. The table shows IoU and pixel accuracy of the best model among all epochs of each model.

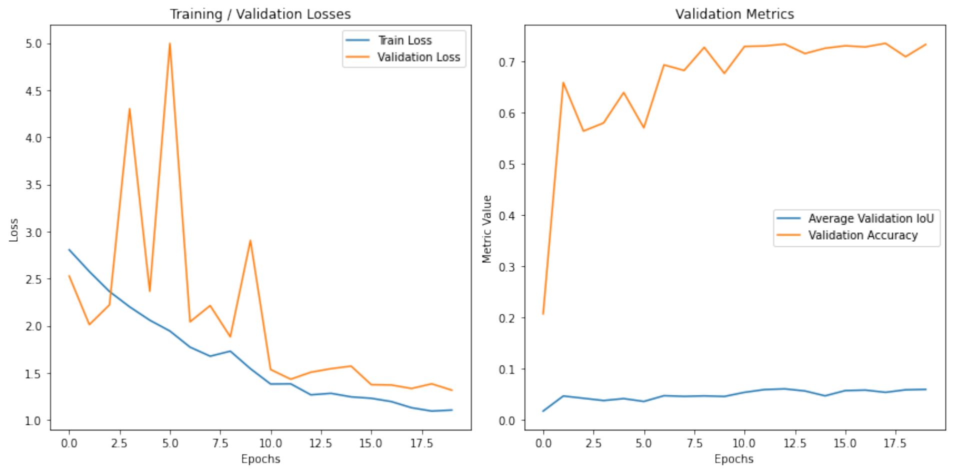

Figure 1: Loss, IoU and accuracy curve of baseline model

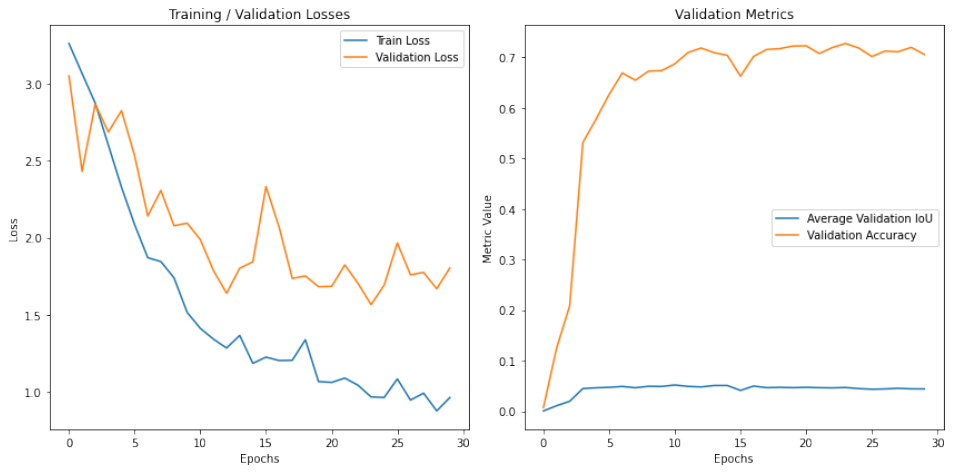

Figure 2: Loss, IoU and accuracy curve of model 4a

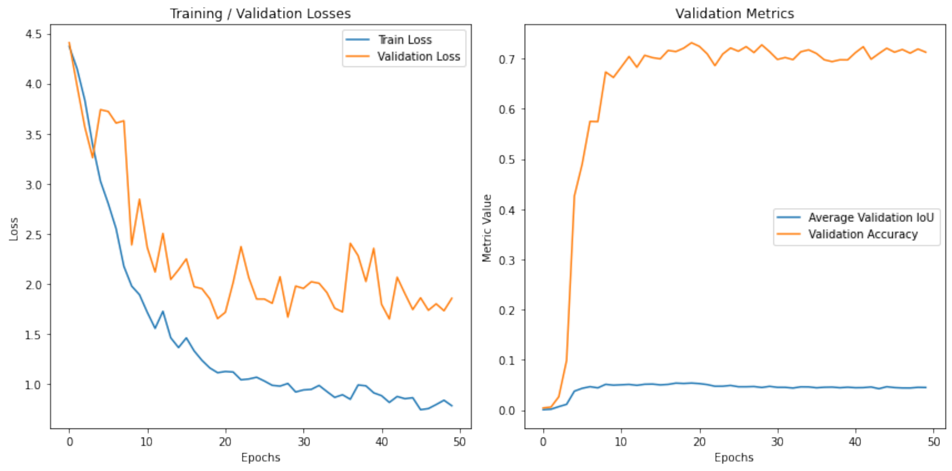

Figure 3: Loss, IoU and accuracy curve of model 4b

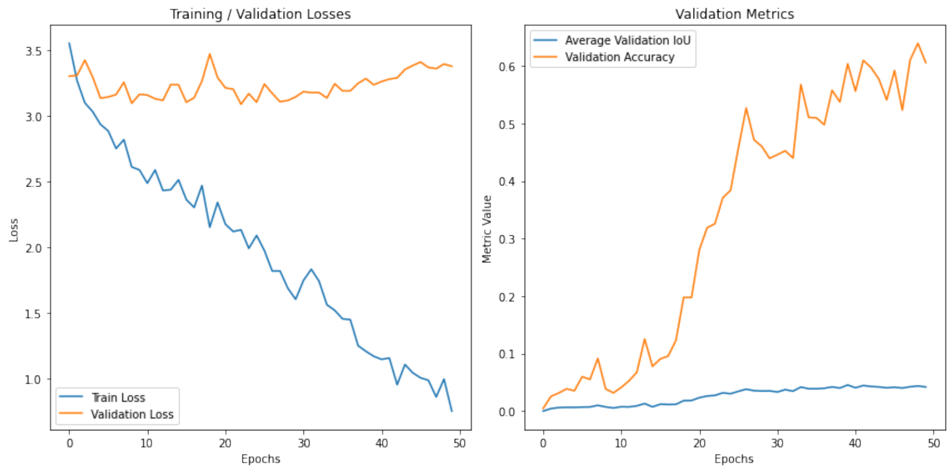

Figure 4: Loss, IoU and accuracy curve of model 4c

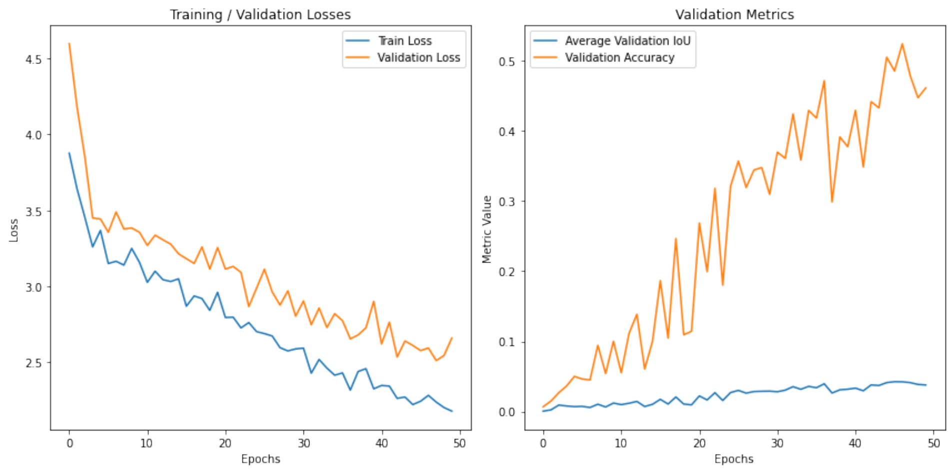

Figure 5: Loss, IoU and accuracy curve of model 5a

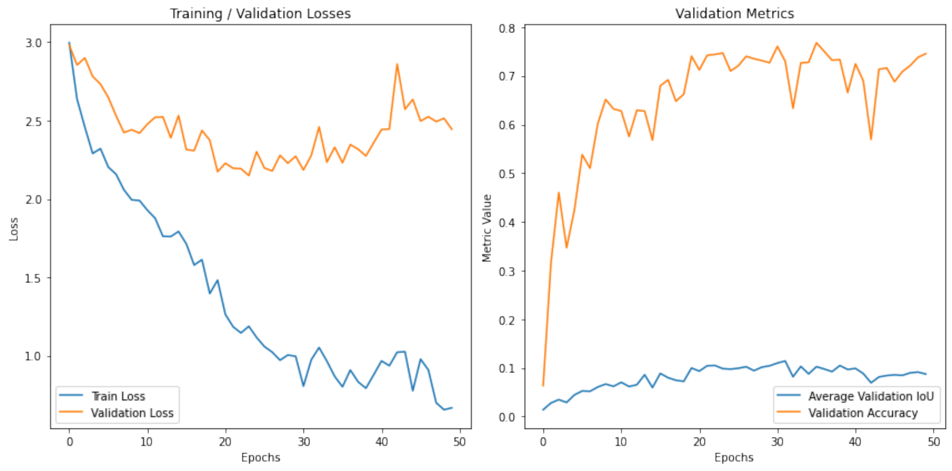

Figure 6: Loss, IoU and accuracy curve of model 5b

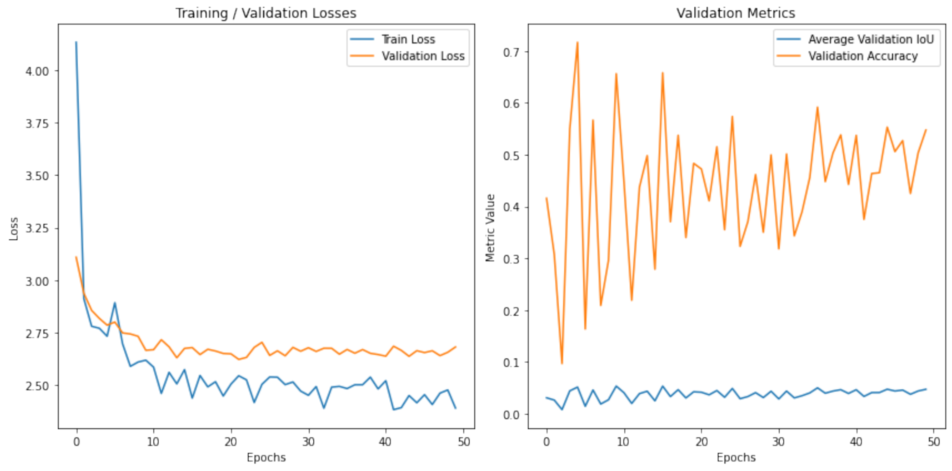

Figure 7: Loss, IoU and accuracy curve of model 5c

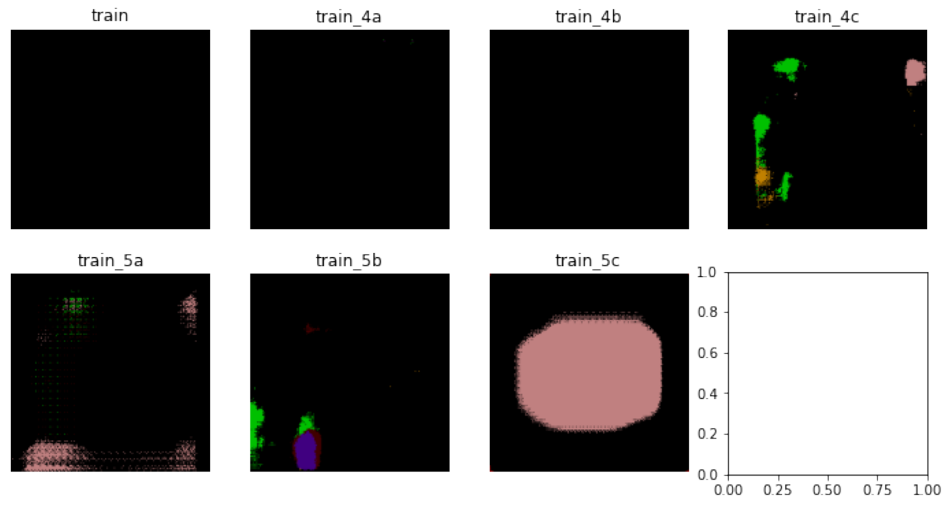

Figure 8: Mask visualization of each model

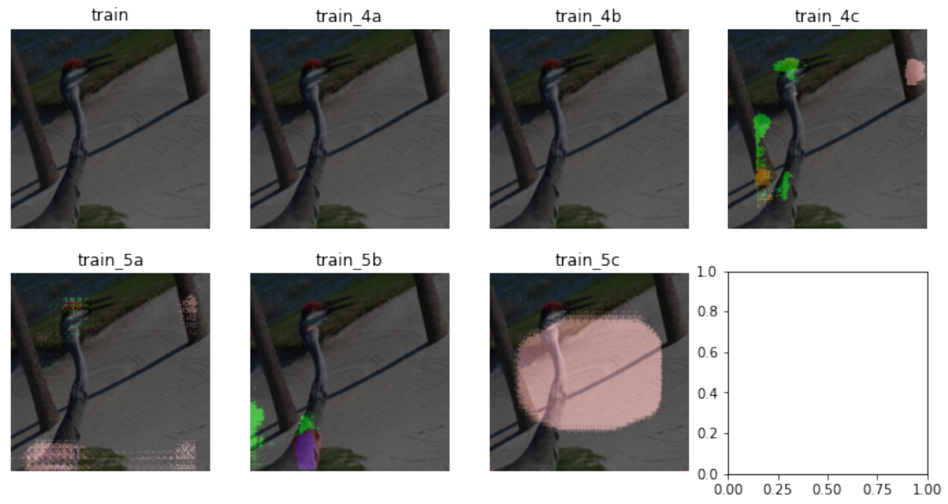

Figure 9: Mask visualization with image

Table 4: Comparison of Model Performances

Model	Best validation IoU	Best validation Pixel accuracy	IoU on test	Pixel accuracy on test
Baseline	0.05486	0.73823	0.04800	0.70578
4a	0.04985	0.72074	0.04811	0.69542
4b	0.04497	0.71236	0.04174	0.69448
4c	0.04271	0.60852	0.04214	0.59277
5a	0.04245	0.48558	0.04275	0.46131
5b	0.11415	0.72988	0.10563	0.70622
5c	0.04887	0.51972	0.05137	0.54130

5 Discussion

5.1 Baseline Model

The baseline model is built with 5 layer encoder and 5 layer decoder. This model reached a relatively IoU and pixel accuracy on both validation and test set. The high accuracy is due to the class imbalance problem. In the mask prediction image, we can clearly see that all pixels are black. Since background is dominated by class, if we simply predict every pixel as background, we will roughly get about 70-75% accuracy. The learning rate is relatively big and we only have epochs here. According to the loss image, though both loss curves are decreasing, the validation loss is not smooth. Seeing the accuracy and IoU image of the validation set, we can see accuracy stop increasing around epoch 5, and IoU also keeps unchanged at this point.

5.2 Improvement

Model 4a has more epochs and lower learning rate than the baseline mode, and we also added a learning rate schedule. So according to the loss image, the curve for both training and validation become smoother. The accuracy and IoU curve are smoother too, but they also stop increasing around 5 epochs. For this model, we still haven't solved the class imbalance problem, in which the prediction mask is still black. The best IoU number on the validation set is 0.005 lower than the baseline model, which is due to randomness of batch and initialization of weight. For both model 4b and 4c, we used a weighted cross entropy loss to deal with the class imbalance problem. The derivation of weight was explained in the method section. And we also use more epochs and lower learning rate.

After using a weighted loss function on Model 4c, the mask visualization shows that our model starts to mark some object, such as bird and pillar, other than the background. Though both IoU and accuracy decrease a lot, it's better than just predicting everything as background. However, the model doesn't correctly predict an object as a bird, it may be caused by insufficient amount of training images and classes. IoU and accuracy curve are increased first, then kept unchanged when near local optimum. The train loss keeps decreasing, but validation loss doesn't decrease. It may be caused by overfitting to the training set and appropriate weight parameters.

Model 4b was built based on 4c, except we added a horizontal flip to training images and targets. It has higher IoU and accuracy than model 4b, and totally different shape of, loss, accuracy and IoU shape. Our best explanation of this phenomenon is that randomness of initialization and batch cause this problem.

5.3 Experiment

For all three models in this section, we use weighted loss function, more epochs and lower learning rate than the baseline model. Specific information can be found in the method section.

For Model 5a, we changed the architecture of the baseline model. We use more stride 1, so that each layer contains more information. We added two more 2D convolution layers and used the Tanh activation function instead. Both validation and train loss curve decrease along epochs. Iou and accuracy curve on the validation set keep increasing. Though we save the best model during training to avoid overfit to the training set, overfitting to the validation set problem still exsit. The result on the test set shows that problem. The IoU on the test set is lower than the baseline model. And accuracy is 20 percent lower than the baseline model, which is really bad. However, according to the mask visualization, the model distinguishes the body and head of bord and pillar from the background, though the classification is wrong.

For model 5b, we use a pretrained Resnet34 to replace the encoder. This transfer learning strategy can save some training time and increase accuracy. The IoU is almost double than the baseline model on both validation and test set. Furthermore, it keeps a good accuracy, above 70%, on both validation and training. This is because we used a pre-trained model. The train loss, IoU and accuracy curve increase along epochs. The validation loss doesn't decrease a lot. Thus, here, we don't have an overfitting problem to both train and validation set. Numerical statistics looks good, but the prediction mask is not quite good. We can see, it predicts 3 classes on the bird's body, and all 3 classes are wrong.

For model 5c, we use a different architecture, U-Net, which also includes encoder and decoder, but includes more layers and complex. Details of architecture can be found in the method section. The U-Net architecture provides benefits for pixel level segmentation tasks, particularly due to its ability to capture both context and localization information effectively. Both loss curves smoothly decrease, but IoU keeps unchanged and accuracy dramatically goes up and down. The IoU on validation and test set is better than the baseline model. However it only has near 50% pixel accuracy. When inputting various pictures to the model, the mask visualization always shows a pink square in the middle. Trying to figure out the reason, we delved into architecture. We discover that after 14 layers of encoder, the output is an 8x8 feature map with 1024 channels. Since out input image size is small, 224*224, after all the encoder layers, 8*8 feature maps might not contain enough information about the original image. To solve such problems, we should use higher resolution images.

5.4 Summary

All above models don't reach our desired outcome. Though some models solve class imbalance problems and distinguish between object and background, they fail to classify the object. We believe if we have a larger train set and try better weight parameters, we can get a relatively better outcome.

6 Authors' Contributions and References

This is a one person group. Thus Lian has 100% contribution on this PA.

[1] Praveen, S.P., Srinivasu, P.N., Shafi, J. et al. ResNet-32 and FastAI for diagnoses of ductal carcinoma from 2D tissue slides. Sci Rep 12, 20804 (2022). https://doi.org/10.1038/s41598-022-25089-2

[2] Olaf Ronneberger, Philipp Fischer, Thomas Brox (2015) *U-Net: Convolutional Networks for Biomedical Image Segmentation* https://arxiv.org/abs/1505.04597