Semantic segmentation with convolutional deep encoder-decoder networks on SUN RGB data

Mikołaj Antoni Barański
MSc. Data Science
Copenhagen Business School
Copenhagen, Denmark
mikba@itu.dk

Alisia Marianne Michel
MSc. Data Science
Copenhagen Business School
Copenhagen, Denmark
alis@itu.dk

***Abstract*—This paper explores the use of convolutional encoder-decoder networks to conduct semantic image segmentation on the 37 object classes in indoor scenes from the SUN RGB-D data set [1]–[4]. The performance of a U-Net [5] trained from scratch and a ConvNeXt [6] pre-trained in the UPerNet [7] method were tested with various hyperparameter settings. We investigate the impact of learning rate changes, cross-entropy loss weighting, augmentations on training data, and depths of fine-tuning (ConvNeXt only). We compare model performance by the mean Intersection over Union (mIoU) metric. Our research indicates optimal results on our data set can be achieved by fine-tuning the pre-trained ConvNeXt model on our data set including extensive image augmentations. We find optimal results are achieved when fine-tuning includes the decoder layers as well as the classifier layers. Additionally, in situations of resource limitations, only training the final classification layer provides sufficiently high mIoU. The findings of our work are limited by the size imbalance between the models (U-Net 31M & ConvNeXt 60M parameters), and the limited training capacity for models with weighted loss. Further research should explore methods of reducing the issue of class imbalance and diverse data sets.**

***Index Terms*—Semantic Image Segmentation, ConvNeXt, U-Net, SUN RGB-D**

I. Introduction

Semantic image segmentation is a computer vision task that has been approached with various methods including convolution and transformer-based approaches as well as multi-modal models capable of zero-shot learning. In the following paper, we return to models based on the architecture, which first revolutionized the computer vision field within neural networks - the convolutional neural network (ConvNet). We study optimal configurations of ConvNet-based encoder-decoder networks tasked with the semantic segmentation of 37 object classes in indoor scenes based on the SUN RGB-D data set [1]–[4]. Two model architectures were chosen for this task:

- Model 1: U-Net [5] convolutional neural network.
- Model 2: UPerNet [7] with a ConvNeXt [6] backbone.

Model 1 has been prepared and trained from scratch entirely on the SUN RGB-D data set, while Model 2 was pre-trained by the authors of its paper on a variety of indoor and outdoor scenes[1] [6]. We experiment with a variety of model hyperparameters including the learning rate, the loss function, training image augmentations, and the depth of trainable layers of Model 2. We then compare the output quality and computational intensity for the two models in their best-performing configurations. This study aims to propose configurations that improve the model model performance while comparing the use of a pre-trained model to one that is developed from scratch. We, therefore, introduce the following research questions:

RQ1: "What are hyperparameter settings that improve the performance of the U-Net and ConvNext for training on the SUN RGB-D data set?"

RQ2: "How does the performance of a fine-tuned pre-trained model compare to a model trained from scratch in semantic image segmentation?"

RQ3: "To what extent is the difference in performance between the models justifiable given their training costs?".

II. Methods

A. Model 1: U-Net

To allow for experimentation on a relatively low-resource, yet powerful state-of-the-art model, we have developed a custom implementation of the U-Net architecture [5][2]. This model has been set up and trained from scratch to allow for extensive experimentation regarding its optimal setup. The U-Net model was developed in 2015 with an intended use in biomedical image segmentation. It is based on multiple convolutional layers designed as an encoder and decoder. The encoder stage of the model uses convolutional layers to decrease the spatial resolution of the image while increasing its number of features from the initial three color channels to 1064. Again, the decoder stage uses convolutional layers, which increase the spatial resolution of the image and reduce the number of features. At each step, the output of each corresponding encoder stage is appended to the input of the decoder stage. This allows the network to both use the deep-learned feature of the image and understand the original structure of the image. Finally, a fully connected layer with the number of neurons equal to the number of classification classes is applied.

[1]The pre-training data sets have no overlap with the data set used within this report.

[2]The PyTorch implementation of the U-Net model was adapted from the code of the following GitHub repository [8].

B. Model 2: UPerNet with a ConvNeXt backbone

To compare U-Net with a newer ConvNet state-of-the-art model with extensive pre-training and limited need for fine-tuning, the UPerNet [7] with a ConvNeXt [6] backbone has been chosen. Specifically, the ConvNeXt-tiny[3] specification has been applied due to its manageable number of parameters (60M) allowing for fine-tuning with limited computational resources.

The ConvNeXt [6] model has been designed as a response to the rise of vision transformer models, which have begun to dominate the computer vision space. Models such as the Swin transformer [9], combined the power of transformers with the shifting-window logic of convolutional networks achieving SOTA performance on vision tasks previously difficult for transformers. The authors of the ConvNeXt paper took a parallel approach being inspired by elements of the transformers' architecture to produce the improved ConvNet model. The resulting model has a similar FLOP size to Swin-Transformers, but improved performance and a simpler efficient architecture.

The strength of this pre-trained model can be attributed to its use of the UPerNet [7] framework for training.

C. Training set-up

Both models were trained in a similar training pipeline with minor alterations to accommodate the different structures of the models. The core loss function used throughout base training was the cross-entropy loss function (Eq. 1) with some experiments using the weighted version of the function to remedy class imbalance. The loss function was set to ignore pixels that were unlabelled in the ground truth mask.

$$H(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{1}$$

Models were trained with early stopping in case of clear convergence or to a limit of 100 epochs. Early stopping was evaluated based on the validation loss and validation mIoU scores of the models. The Adam optimizer was applied in training. We used a batch size of 64 except for larger models where 32 or 16 was used (for Model 2's unrestricted versions). All models were trained on three Nvidia A40 GPUs to ensure timely training and consistency among experiments.

D. Evaluation metrics

To evaluate the performance of our models, we have used applicable metrics in line with relevant literature. The main metric evaluating the quality of the image segmentation is the mIoU (mean Intersection over Union) score as seen on Eq. 2. The mIoU calculates the mean of the share of correctly classified pixels for each class over the number of pixels in the class in the true and predicted mask. This metric penalizes the model both for under and over-classifying a given class. For more fine-grained analysis we have also investigated the IoU values for each class individually to inspect for class imbalance.

$$mIoU = \frac{1}{N}\sum_{i=1}^{N} IoU_i = \frac{1}{N}\sum_{i=1}^{N} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{FN}_i} \tag{2}$$

E. Pre-processing & Augmentation

We have experimented with the effect of different augmentation techniques on the images to increase the robustness of the models. The specific methods applied to each model specification are shown in Table. I[4].

TABLE I: Image Augmentation

Methods		**Approach**	
Name	***Prob.***	*Raw*	*Augmented*
Resize to 512px	1.0	✓	✓
Normalize Pixels	1.0	✓	✓
Horizontal Flip	0.5		✓
Vertical Flip	0.5		✓
Random Crop	0.2		✓
Brightness/Contrast	0.2		✓
Rotate	0.1		✓

III. Experiments

A. Data Source

The data set chosen for this research is the SUN RGB-D data set [1]–[4]. It consists of 10,335 images of diverse indoor scenes in three color channels and additionally a depth measure of the scene. The images are extensively annotated with 2D polygons and 3D bounding boxes, covering circa six thousand unique object labels. Within our study, we have chosen to use the RGB images and 2D object polygons to train neural networks in classifying each pixel to one of 37 objects proposed by the authors of the data [1]. We have left out the depth dimension of the data to focus on models more generally applicable to the purely RGB-based image data.

The data set contains a mixture of high-quality annotation masks and ones lacking accuracy in either pixel coverage or the label itself. In Fig. 1, one can observe a high-quality example with the right variety of correct labels and good pixel coverage of the appropriate objects. On the other hand, in Fig. 2, the cabinets in the lower half of the photo are not labeled and the kitchen counter is mislabeled as a table. Through an exploration of the data, we have concluded that the majority of images have sufficiently good quality labels and no data cleaning was imposed. One drawback of the data set is the wide class imbalance among the 37 objects. This not only includes the expected background elements (floor and wall) but also items such as chair, table, bed, etc. As seen in Fig. 10, the less common items have a very low share of labels and by the example of Fig. 2, we can see images do not have these elements correctly labeled. However, we do observe that the majority of masks have at least 60% coverage of the image (see Fig. 11).

[3]Model implemented based on HuggingFace submission https://huggingface.co/openmmlab/upernet-convnext-tiny

[4]U-Net used standard normalization with mean 0 and standard deviation of 1, while ConvNeXt used values recommended by the model's authors

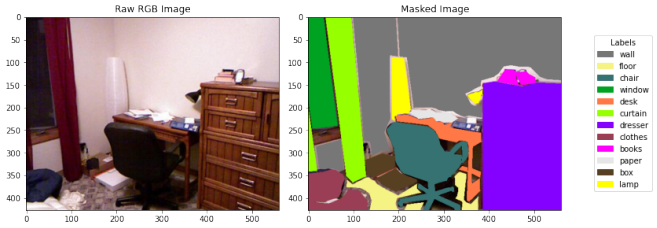


Fig. 1: Example of data set contents - good raw image and masks

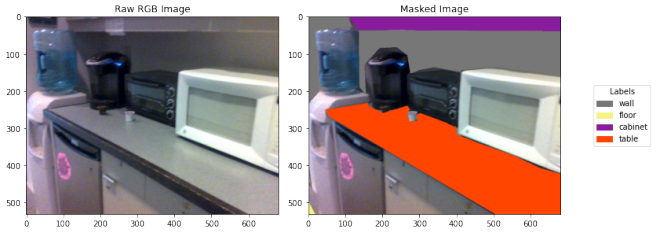


Fig. 2: Example of data set contents - bad raw image and masks

The data set was split into train, validation, and test subsets to allow for hyperparameter tuning and unbiased testing. The split ratio was 60%, 20%, and 20%, which in terms of the number of images was 6,213 and 2,071 respectively. Our model and data set were implemented in Python using the PyTorch framework[5].

B. Hyperparameter Finetuning for Model 1: U-Net

For selecting appropriate hyperparameters, the model performance and computational feasibility were taken into account. The optimal epoch was selected based on model performance measured by the maximum validation mIoU.

1) Learning Rate: To choose an appropriate learning rate that determines the extent to which weights are updated, the U-Net was first trained for 20 epochs with four different learning rates between 0.01 and 0.0001 decreasing in powers of ten. A choice was made based on the training and validation loss graph. A learning rate of 0.01 did not converge which might be due to exploding gradients leading to a numerical overflow. The next lower learning rate of 0.001 showed the expected behavior of both loss functions decreasing even though the validation loss exhibits fluctuations from epoch six, which can be seen in Fig. 12a.

The validation loss function for a U-Net trained with a learning rate of 0.0001 in Fig. 12b displayed the smoothest decline. In comparison, a learning rate of 0.00001 exhibits stronger fluctuations and a slower decline. This can be seen in Fig. 12c. It was therefore decided to progress with a learning rate of 0.0001 for all the following U-Net model experiments.

Previous work on image segmentation in the medical field by [12] suggested that the application of a learning rate scheduler might increase the model performance. It was decided to train the model for 100 epochs with a plateau scheduler which reduces the learning rate once the validation loss stops decreasing. As seen in Table II, the model trained including a plateau scheduler outperformed the model without regarding validation mIoU. Therefore, further experiments were conducted on a U-Net with a learning rate of 0.0001 and a plateau scheduler.

TABLE II: U-Net learning rate experiments

Learning Rate	Epoch	Val Loss	Val mIoU
0.01	Nan	Nan	0.008
0.001	18	1.375	0.131
0.0001	**20**	**1.402**	**0.120**
0.00001	20	2.083	0.061
0.0001	75	1.071	0.255
0.0001 + scheduler	**73**	**0.986**	**0.279**

2) Weighted Cross Entropy Loss: As noted in Sec. III-A and shown in Fig. 10, the data set suffers from a large class imbalance. To counteract this imbalanced class distribution it was decided to experiment with weighted cross entropy loss. Prior research on weighted cross entropy loss in object detection suggests that it improves the performance of minority classes [13]. For this project, the loss for each class was weighted inversely proportional to the class frequency. In Fig 3, we can observe that the weighted cross-entropy loss led to only minor improvements in IoU for selected under-represented classes; however, this came at a significant drop in IoU for previously well-performing classes. This is further reflected in predicted masks (Fig. 4), where the model with weighted loss assigns a large number of incorrect classes to an image containing comparably few classes.[6]

3) Augmentation: To assess the utility of the applied augmentations, the best-performing hyperparameters were used to train a further model without using augmentations on the training data. As expected, this led to a lower performance of -0.063 mIoU compared to the model employing augmentations (see Table III). The model also converged in a lower number of epochs (40 vs. 98 with augmentation), indicating that it had explored all the variance in the available training data. A visual inspection of the mask indicated the model without augmentations creates less consistent object masks for classes covering large broad areas, e.g., tables where the mask does not cover the whole object or where other masks are added within the object area. Our results indicate, that augmentations improve model generalization for semantic image segmentation in convolutional networks.

C. Hyperparameter Fine-tuning for Model 2: UPerNet with a ConvNeXt backbone

For our second model, we have conducted similar experiments as in the section above and further experimented with

[5]Elements of the SUN RGB-D data set implementation were adapted from [10], their code implementation can be found in the GitHub [11]

[6]Validation images were captured in epochs with the lowest validation loss, not the highest validation mIoU.

TABLE III: Training Results

Model	Hyperparameters	Trainable parameters	Best Epoch	Val mIoU
U-Net	**Plateau + 0.0001LR**	**31,039,973**	**98**	**0.2835**
U-Net	Plateau + 0.0001LR + woAugment	31,039,973	40	0.2207
U-Net	Plateau + 0.0001LR + Weighted	31,039,973	99	0.1634
ConvNeXt	Classifer + 0.001LR	28,490	13	0.4476
ConvNeXt	Classifer + 0.0001LR	28,490	92	0.4464
ConvNeXt	Decoder + 0.001LR	32,334,154	25	0.4430
ConvNeXt	**Decoder + 0.0001LR**	**32,334,154**	**13**	**0.4482**
ConvNeXt	Decoder + 0.0001LR + woAugment	32,334,154	8	0.4394
ConvNeXt	Decoder + 0.0001LR + Weighted	32,334,154	98	0.4041
ConvNeXt	Encoder + Decoder + 0.0001LR	60,155,626	18	0.4272

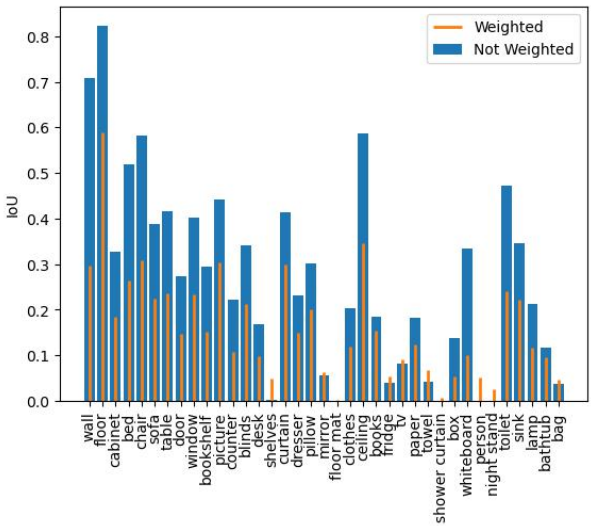


Fig. 3: U-Net IoU per class for weighted and not weighted cross-entropy loss

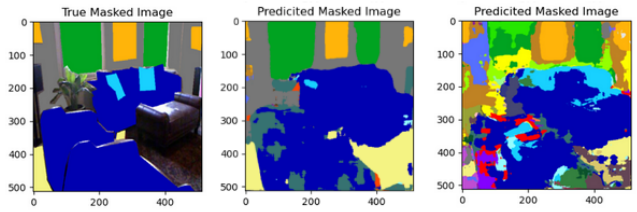


Fig. 4: Predicted mask for validation image with not weighted and weighted cross-entropy loss

the number of layers to be trained. This is possible as our ConvNeXt model is already pre-trained; therefore, we can choose the depth to which we adapt the parameters already learned by the model. We adopted the strategy of successively unfreezing layers for training to see the effect on model performance.

1) Training Classifier Layers Only: For the first experiment, we only train the final classifier layers with 28K trainable parameters. In the beginning, the model was fine-tuned with a learning rate of 0.001 where it already reached its maximum validation mIoU in epoch 13 out of 100 epochs. Then the performance was tested with a lower learning rate of 0.0001 which led to decreased performance by 0.001 mIoU within 100 epochs (see Table III).

2) Fine-tuning the Decoder: Next, the decoder and classifier were fine-tuned using 32M trainable parameters (a thousand times larger than training only the classifier layer)[7]. Two learning rates were tested at 0.001 and 0.0001. Here, the learning rate of 0.0001 led in fewer epochs to the best validation mIoU, outperforming the 0.001 by 0.005 mIoU. Additionally, as seen in Table III, training the decoder layers in addition to the final layers outperformed only training the classifier by 0.001 mIoU.

3) Fine-tuning all Layers: Finally, all layers (encoder, decoder, and classifier) were fine-tuned with a learning rate of 0.0001. This resulted in having 60M trainable parameters in the model[8]. We decided not to experiment with the learning rate due to the higher training duration per epoch. The results (Table III) show that fine-tuning all layers with these specific hyperparameter settings leads to a decrease in performance compared to the fine-tuning of the decoder & classifier by 0.021.

4) Weighted Cross Entropy Loss: Since in the previous experiment training the decoder and classifier layers with a learning rate of 0.0001 performed best these hyperparameter settings were tested with a weighted cross-entropy loss. The cross-entropy loss was set up in the same way as with the U-Net model. Here the weighted loss function also led to a significant decrease in model performance (-0.044 mIoU). The IoU improvements in Fig. 5, are less noticeable than in the case of U-Net with most under-represented classes experiencing a decline in IoU compared to the non-weighted model. This is also reflected in the predicted masks created by the model with weighted loss, which in the case of ConvNeXt produces comparable masks to the unweighted variant (see Fig. 13)[9].

5) Augmentation: The best-performing model was then re-trained without using augmentations on the training data. As is the case with U-Net, here we also observed a decrease in model performance and convergence at a much lower epoch compared to the model with augmentations. This indicates the

[7]Due to the GPU's limited memory, the batch size was decreased to 32.
[8]Due to the GPU's limited memory, the batch size was decreased to 16.
[9]Validation images were captured in epochs with the lowest validation loss, not the highest validation mIoU.

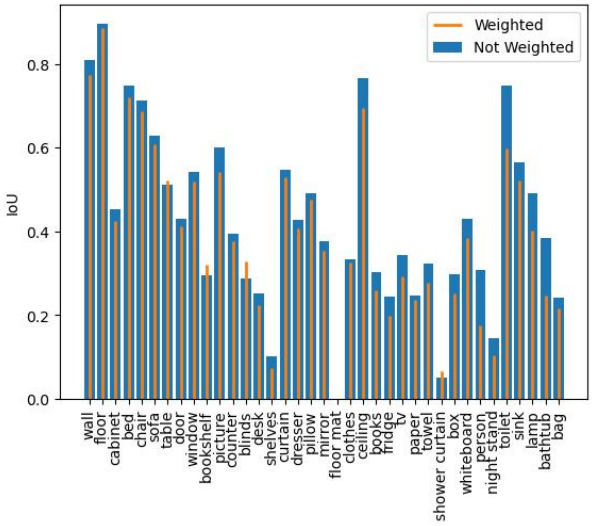


Fig. 5: ConvNeXt IoU per class for weighted and not weighted cross-entropy loss

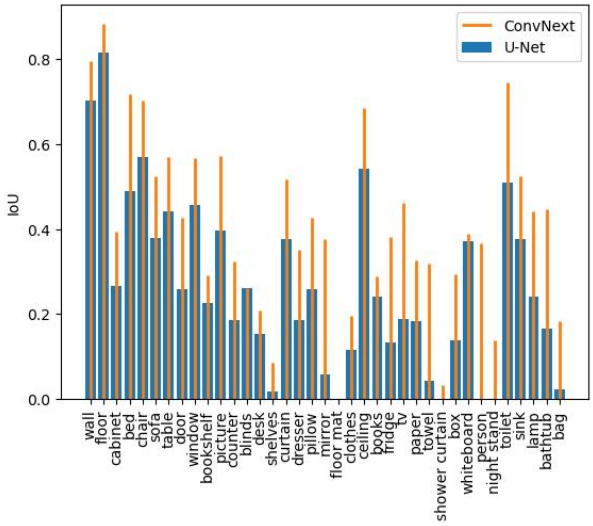


Fig. 6: IoU per class on the test set for U-Net and ConvNext

value of using augmentations both in model training and fine-tuning a pre-trained model in our research context.

IV. Results of pre-trained and fully trained model

The final model performance was obtained by evaluating the previously best-performing U-Net and ConvNext on the test data set. The results in Table IV show that the pre-trained ConvNext model with a mIoU of 0.430 outperforms the U-Net model with a mIoU of 0.278.

In Fig. 6 the IoU achieved during segmentation of the test set for both models is compared. The results show that the ConvNext model outperforms the U-Net model in nearly every class. Only for segmenting blinds, both seem to do equally well.

Fig. 7 shows the development of validation IoU for selected classes over the training epochs for the U-Net. Interestingly, some classes are recognized only later in training (bathtub and fridge), while very well-represented classes (wall and floor) achieved a high IoU from the beginning of training. Some classes never picked up (shower curtain). The patterns exhibited are similar to those observed in the validation loss function in Fig. 14. The development of validation IoU per class over epochs looked more stable for the ConvNext model and is well summarized by the validation mIoU in Fig. 15.

TABLE IV: Best U-Net and ConvNeXt performance on test data

Model	Test mIoU
UNet: 0.0001LR + Plateau	0.278
ConvNext: Decoder + 0.0001LR	**0.430**
Difference	*Δ0.152*

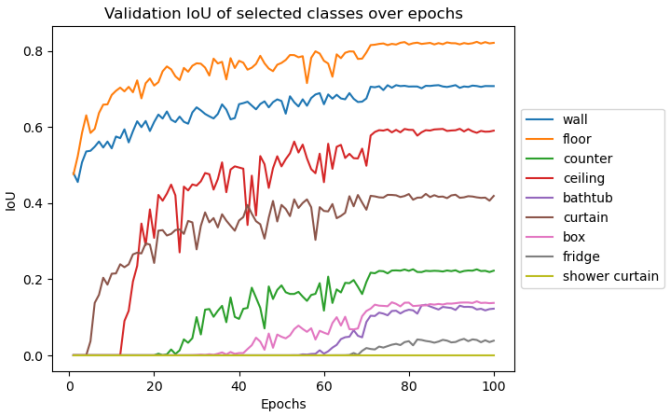


Fig. 7: U-Net validation IoU of selected classes over epochs

V. Discussion

Answering our research question *RQ1*, we have found that both our models perform best with training image augmentation and without class imbalance weighting of the loss function. The augmentation of training images was shown to allow the model to train for a further number of epochs before overfitting while achieving a higher performance. Class imbalance remains an issue with the models we have trained; however, the loss function weighting that we tested was not a successful solution. This may have been caused by the 100 Epoch training limit set in our experiments, where both the weighted U-Net and ConvNeXt did not appear to converge. For both models, a learning rate of 0.0001 achieved the highest performance. Additionally, the U-Net performed better with a Plateau learning rate scheduler. The highest score of the ConvNeXt fine-tuning was achieved by setting the decoder and classifier as trainable layers.

Our results indicate a superior performance of the pre-train ConvNeXt model against a U-Net trained from scratch. This result was expected, given the extensive pre-training

methods used in the UPerNet [7]. This is especially visible on the less represented object classes in the data set, which are poorly covered by the U-Net (see Fig. 3&5). In Fig. 8, we can observe the ConvNeXt model makes a much more accurate classification of the room scene, with the precise classification of classes like lamp, pillow, or books that are under-represented in the data. U-Net struggles with the more busy parts of the image misclassifying many pixels. Similarly in Fig 9, the U-Net struggles to classify a bed in a dark scene. Noteably in both scenarios, ConvNeXt produces masks of almost higher quality than the ground truth. ConvNeXt only appears to struggle in more complex texture combinations, such as a wooden cabinet by a wooden desk wall (Fig. 16), or with ambiguous objects, such as a window through which you can see another class of object (Fig. 17).

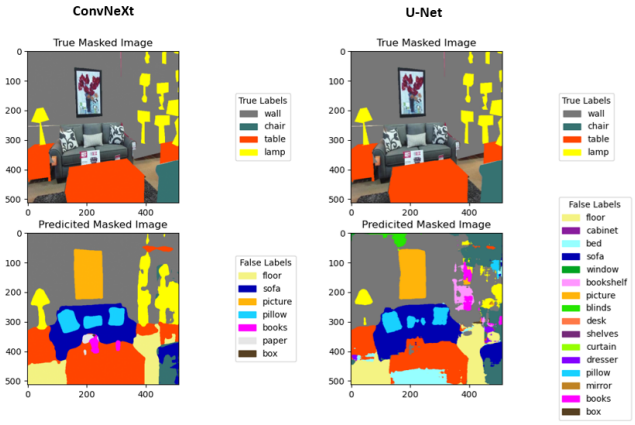


Fig. 8: Classification of general scene

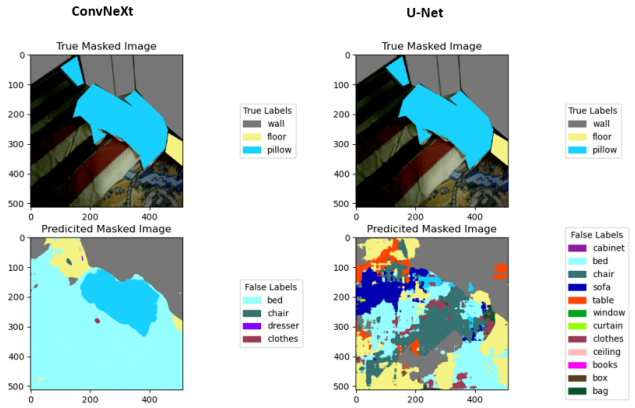


Fig. 9: Classification performance in ambiguous scene

With this, we can answer our research question *RQ2*, stating that there is a significant performance improvement when using a fine-tuned pre-trained network. Moreover, in the context of *RQ3* this performance is fully justifiable in terms of training resources, the fine-tuned ConvNeXt with 30M trainable parameters over only 13 epochs vastly outperformed the best U-Net with 31M parameters, which needed over 98 epochs. In terms of inference, the fine-tuned ConvNeXt model consists of 60M parameters; therefore, inference is slower and its performance gain must be weighed against the available computational resources in deployment.

One limitation of our study is the difference in size between the compared models. We used the smallest ConvNeXt pre-trained version *tiny*; however, even this model has double the parameters of the U-Net (60M vs 31M). Research including the original ConvNeXt paper [6] indicates that model size has a direct influence on higher model performance. Another limitation is the different levels of training afforded to the models. The ConvNeXt is pre-trained not only on a diverse data set but also on a variety of tasks within UperNet including object, scene, texture, material, and part detection [7]. Our U-Net is trained only on the semantic segmentation and only using the SUN RGB-D data set. The data set in itself may cause lower training performance due to the noise in the images and the large proportions of objects being unlabelled.

Further research could be done including additional data sets from different sources or reducing the noise and unlabelled areas in the current data set to circumvent problems caused by low-quality labeling. Furthermore, the depth component of the data could be included in further architecture experiments, since currently well-performing[10] encoder-decoder models like [14] and [15] do so. Additionally, one could explore the performance of different weighted loss strategies. [13] suggest besides the inverse class frequency other weighted loss strategies like balanced cross-entropy or focal loss. The imbalanced data could also be considered during the data set splitting and data augmentation allowing for a more balanced representation of classes in the training data. Another aspect that could benefit from further fine-tuning is the layer architecture which was not changed for the experiments in this project. Last, a learning rate scheduler could also be applied to the ConvNext model in hopes of improving performance as it did so for the U-Net.

VI. Conclusions

We explored the use of a U-Net trained from scratch and a fine-tuned pre-trained ConvNeXt for semantic image segmentation of the SUN RGB-D dataset. Through our hyper-parameter experiments and ablation study we demonstrated the importance of applying augmentations to training data, found no performance gain from loss function weighting, and for fine-tuning observed that freezing the encoder of the model leads to improved performance. On our data set, the ConvNeXt model with a fine-tuned decoder over 13 epochs, performed best among all model setups. It achieved a mIoU of 0.430 on the test set, while the top U-Net achieved 0.278. Future work should focus on exploring more diverse training approaches such as multi-task training, which could harness the depth data available within the SUN RGB-D data and experiment with diverse training data.

[10]Best performing models for semantic segmentation on SUN-RGBD on paperswithcode.com, accessed at 08.01.2024

REFERENCES

[1] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.

[2] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3d object dataset: Putting the kinect to work," *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*, pp. 141–165, 2013.

[3] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*. Springer, 2012, pp. 746–760.

[4] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1625–1632.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[6] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022.

[7] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," 2018.

[8] A. Persson, "Semantic segmentation w. u-net," 2021, adapted from the code available at the GitHub repository. [Online]. Available: https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/image_segmentation

[9] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[10] D. Seichter, M. Kohler, B. Lewandowski, T. Wengefeld, and H.-M. Gross, "Efficient rgb-d semantic segmentation for indoor scene analysis," *arXiv preprint arXiv:2011.06961*, 2020.

[11] M. Koehler and D. Seichter. (2021) Rgbd-seg. [Online]. Available: https://github.com/Barchid/RGBD-Seg/tree/master

[12] A. Al-Kababji, F. Bensaali, and S. P. Dakua, "Scheduling techniques for liver segmentation: Reducelronplateau vs onecyclelr," in *International Conference on Intelligent Systems and Pattern Recognition*. Springer, 2022, pp. 204–212.

[13] T. H. Phan and K. Yamamoto, "Resolving class imbalance in object detection with weighted cross entropy losses," *arXiv preprint arXiv:2006.01413*, 2020.

[14] S. Dong, Y. Feng, Q. Yang, Y. Huang, D. Liu, and H. Fan, "Efficient multimodal semantic segmentation via dual-prompt learning," *arXiv preprint arXiv:2312.00360*, 2023.

[15] Y. Wang, X. Chen, L. Cao, W. Huang, F. Sun, and Y. Wang, "Multimodal token fusion for vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 186–12 195.

APPENDIX

A. Dataset Exploration

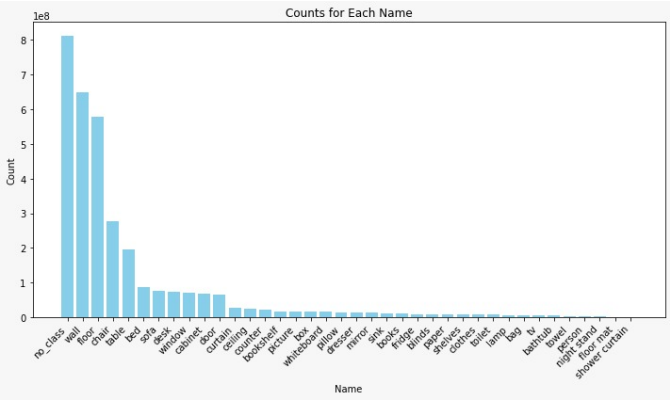


Fig. 10: Class Imbalance in Data Set

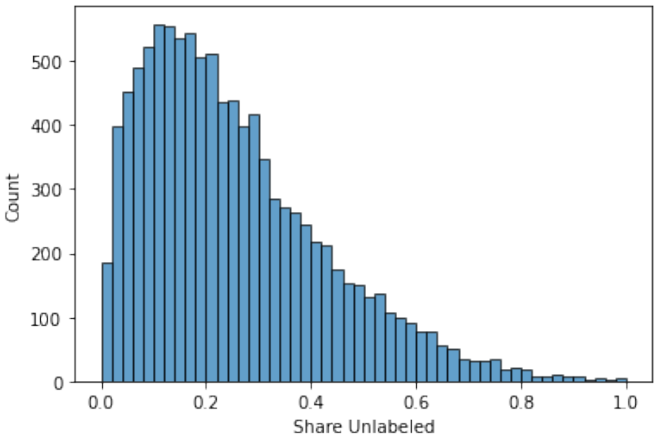
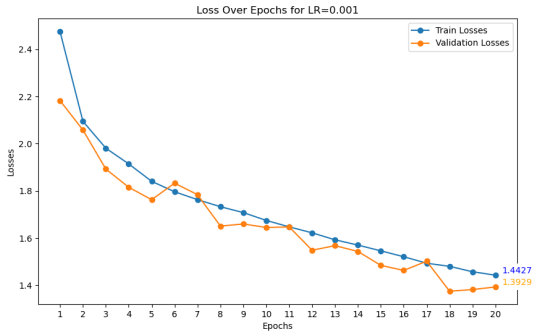
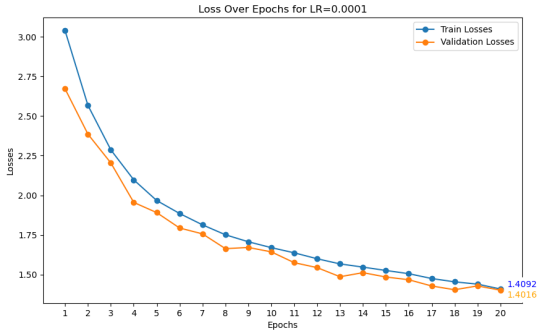


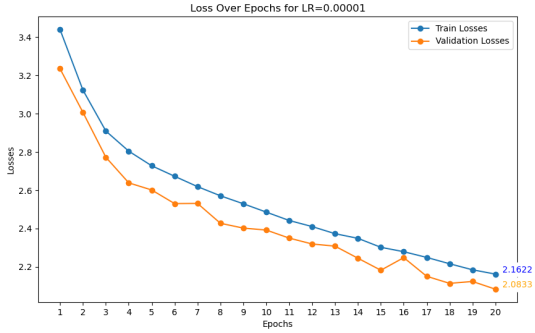
Fig. 11: Histogram of the share of unlabeled pixels in images



(a) Learning rate of 0.001



(b) Learning rate of 0.0001



(c) Learning rate of 0.00001

Fig. 12: Loss functions of a U-Net trained with different learning rates

B. Experiments

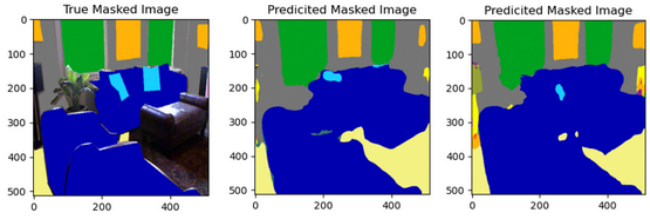


Fig. 13: Predicted mask for validation image with not weighted and weighted cross-entropy loss

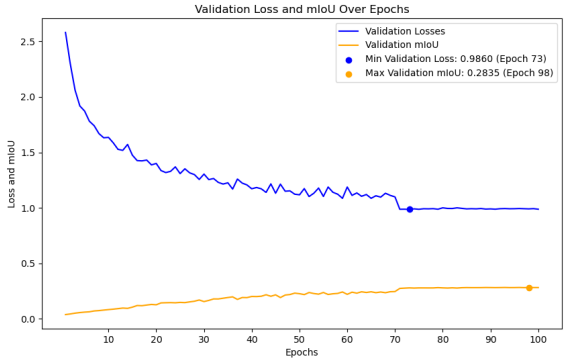


Fig. 14: Development of validation mIoU and loss over epochs for U-Net

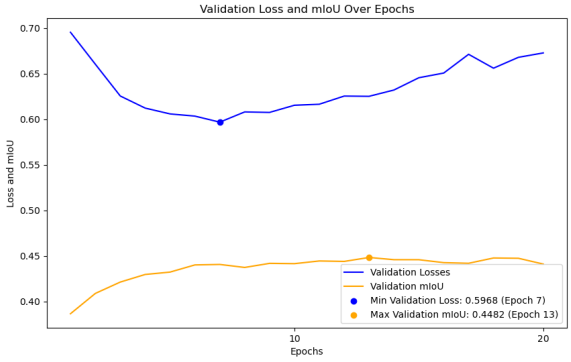


Fig. 15: Development of validation mIoU and loss over epochs for ConvNext

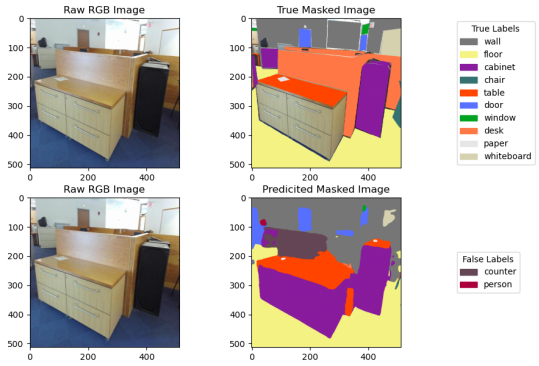


Fig. 16: Segmentation performance of ConvNeXt on a scene with difficult textures

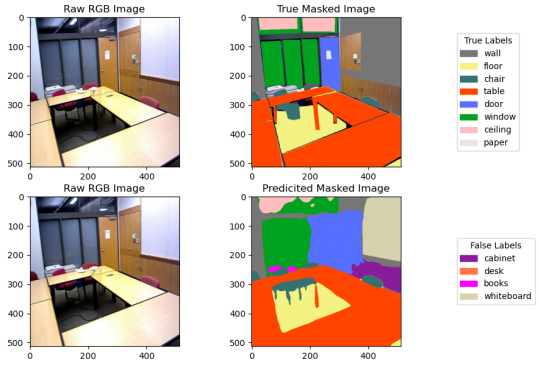


Fig. 17: Segmentation performance of ConvNeXt on a scene with ambiguous window