Implementing and Extending U-Net for Semantic Segmentation

Nico Fidalgo & Jade Nair
Harvard University
COMPSCI 2831
nfidalgo@college.harvard.edu, jnair@college.harvard.edu

Abstract

This work implements, extends, and evaluates the U-Net and Fully Convolutional Network (FCN) architectures for semantic segmentation, originally proposed by Ronneberger et al. and Shelhamer et al., respectively. We present several implementations: a baseline U-Net closely following the original design, an enhanced U-Net incorporating residual blocks, batch normalization, and improved training strategies, and multiple hybrid approaches combining U-Net and FCN components.

All models were trained and evaluated on the CamVid dataset, which differs from the original domains in both tasks and complexity, featuring 32 semantic classes. The enhanced U-Net outperformed the baseline U-Net in segmenting large, dominant classes such as "sky" and "road" with Dice scores of 0.954 and 0.928, respectively, but struggled with small classes like "sidewalk." Dilated U-Net architectures improved segmentation performance on smaller objects for the baseline, but results were inconsistent for optimized models. Hybrid FCN architectures with 256-pixel image resolution achieved Dice scores of 0.93 for "sky" and 0.91 for "road," outperforming the baseline FCN while maintaining efficiency.

Our experiments show the viability of adapting these segmentation methods to new domains, with optimized hybrid approaches demonstrating strong performance for urban scene understanding. These findings provide valuable insights for future work on enhancing segmentation accuracy in complex multi-class scenarios.

1. Introduction

Semantic segmentation aims to assign semantic labels to every pixel in an image, which is vital for applications such as autonomous driving, medical imaging, and robotics. Several methods have seen domain-specific success in segmenting images. The U-Net architecture, introduced by Ronneberger et al. [2], brought significant improvements to medical image segmentation by using a convolutional network with symmetric encoder-decoder paths and skip connections. Meanwhile, Shelhamer et al. have achieved high-quality results with fully convolutional network (FCN) adaptations to existing models such as AlexNet.

We set out to apply these techniques to another domain. The CamVid dataset primarily catalogues scenes on roads which feature a number of objects that could be useful to identify, from cars to nature to buildings to pedestrians. Applying semantic segmentation techniques to such a dataset could be used for systems that detect objects of importance on the road, such as partially or fully autonomous vehicles. It can sometimes be more complex than the original U-Net domain of binary medical segmentation, because there are 32 distinct classes that the network must learn to identify.

While U-Net excelled in binary medical segmentation tasks, applying it to more complex multi-class datasets such as **CamVid** introduces new challenges, such as:

- Increased variability in lighting, object scales, and classes.
- Handling small and irregular objects, e.g., pedestrians and traffic signs.

In this project, we experiment with a series of methods to apply semantic segmentation to this dataset accurately, despite the challenges. We take the following steps:

- Implement a baseline U-Net following the original architecture.
- Develop an enhanced U-Net with modern techniques such as residual connections and combined loss functions.
- Implement a basic Fully Convolutional Net.
- Experiment with dilation techniques on these methods.
- Experiment with new hybrid variants of U-Net techniques and FCN architectures.
- Finally, we evaluate all models' performance on the CamVid dataset through rigorous analysis.

2. Literature Review

2.1. Original U-Net Design

Ronneberger et al. [2] proposed U-Net for biomedical image segmentation. Key components of the U-Net architecture include:

- **Encoder Path:** Downsampling path with repeated convolutional layers and max-pooling to capture hierarchical features.
- **Decoder Path:** Upsampling path using transposed convolutions to recover spatial information.
- **Skip Connections:** Feature maps from the encoder are concatenated with corresponding decoder layers to retain fine-grained details.

2.2. Performance of Original U-Net

The original U-Net achieved state-of-the-art results on biomedical segmentation tasks, particularly on the ISBI challenge dataset, with a **Dice coefficient of 0.92**. Its ability to recover fine details made it highly effective for tasks requiring precision, such as segmenting cells and organs.

2.3. Application to Urban Scenes

Urban scene segmentation, as in the **CamVid dataset**, presents additional challenges:

- High class imbalance (e.g., large "sky" vs. small "pedestrian"), in both size and frequency of each class type.
- Multi-class segmentation instead of binary masks.
- Complex backgrounds and lighting variations.

2.4. Original FCN Design

Shelhamer et al. [3] proposed a fully convolutional design to improve the accuracy and performance of the models as compared to partially convolutional image processing methods. The authors modified AlexNet and other popular neural network architectures to keep the final layers convolutional, rather than concluding with linear layers. They also included skip connections, similar to those in the U-Net design. Skip connections are an important facet of semantic segmentation and were included in both papers because the process requires both high-level, sweeping features and lower-level, granular features.

2.5. Performance of Original FCN Design

The fully convolutional networks developed by Shelhamer et al. outperformed state-of-the-art models on the PASCAL VOC dataset and worked well on SIFT Flow and NYUD, with a maximum pixel accuracy of 90% and IU (intersection-over-union) measures ranging from around 40 to 60.

2.6. Takeaways and Applications

These methods were honed to the application domains they were meant for and the types of datasets they were tested on. Both received high accuracy in their respective domains, but the datasets in the FCN study were slightly more varied and required slightly more variety in classes and objects. We hypothesize that this will carry over into our implementation on the CamVid set.

3. Methods

3.1. Dataset: CamVid

The CamVid dataset [1] consists of high-resolution urban images with 32 annotated classes. Unlike the binary masks used in the original U-Net, this dataset introduces significant class imbalance.

Preprocessing:

- Resizing images to **256×256** for Model V1 and **384×384** for Model V2.
- Normalization to scale pixel values between [0, 1].
- Data augmentation: random flips, rotations, and brightness adjustments.

3.2. Baseline U-Net (Model V1)

The baseline closely follows the original U-Net:

- **Input size:** 256×256.
- **Architecture:** Standard encoder-decoder design with **skip connections**.
- **Loss function:** Cross-entropy loss.
- **Optimizer:** Adam with a learning rate of 3e-4.

3.3. Enhanced U-Net (Model V2)

Our enhanced model introduces the following improvements:

- **Residual Blocks:** Improve gradient flow and convergence stability.
- **Batch Normalization:** Normalizes intermediate activations for faster convergence.
- **Combined Loss:** Cross-entropy loss + Dice loss to handle class imbalance.
- **AdamW Optimizer:** Weight decay for regularization.
- **Input size:** 384×384 for higher resolution feature learning.

3.4. Training Details

- Both models were trained for **100 epochs** on the CamVid dataset.
- **Batch size:** 8.
- **Learning rate scheduling:** ReduceLROnPlateau with patience of 10 epochs.

3.5. Other Modifications

In addition to customizing our U-Net for our segmentation task and iterating on its specifications, we conducted new experiments that combined and iterated on other methods.

3.6. Dilation

First, we experimented with dilating the images by a factor of 2 because we wanted to increase the receptive field without losing resolution. This was particularly salient for our task as we needed to identify both large-scale (sky, road, buildings) and small-scale (people, signs, trees) segments of our images. We trained a dilated baseline and a dilated enhanced U-Net. These models were each trained for 30 epochs with a batch size of 4, and with the same loss and optimization parameters as above. Results are in the following section.

3.7. Baseline FCN Implementation

We trained a very basic FCN model as well, based on the AlexNet version of the architecture. This version did not utilize skip connections, upsampling, or downsampling, and utilized the same input size, loss and optimizer parameters when training.

3.8. Hybrid FCN Implementation

Additionally, to iterate on these existing methods and understand how to work with the unique challenges CamVid provided, we conducted several experiments with hybrid approaches. Specifically, we developed a hybrid method that uses the upsampling, downsampling, and skip connection strategies from the U-Net as well as the fully convolutional architecture from the FCNs, and set out to see how this new strategy would perform on the CamVid dataset. For our hybrid, fully convolutional method, we retained the upsampling and downsampling mechanisms across the network as well as the method of skip connections, while creating an architecture that approximated a fully convolutional AlexNet as seen in the FCN paper we read. Note that upsampling and skip connections are employed in the original paper's methodology; this version follows the UNet methods of encoder and decoder parts and uses more skip connections.

3.9. Iterations on Hybrid FCN

While reviewing all the results, we realized that batch normalization, carried over from the UNet, as well as dilation, may have been interfering with performance. We tried creating another version without this extra batch normalization layer between the convolutional and ReLU layers and without dilation. We also worked with a couple different image resolutions, as we balanced performance with computational limitations of running a more complex network.

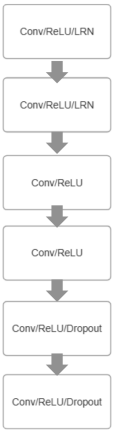

Figure 1: Baseline FCN Network Architecture.

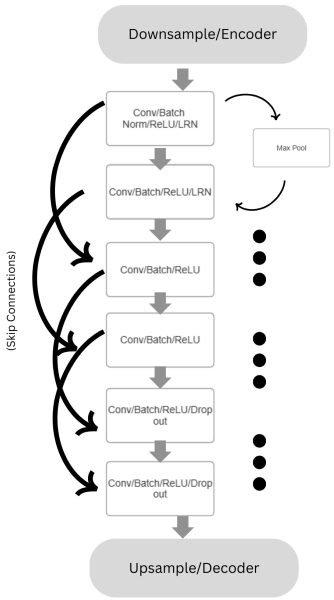

Figure 2: Hybrid FCN Network Architecture.

4. Results

4.1. Feature Representation

Figure 3 and Figure 4 show the t-SNE visualizations of learned features for Model V1 and Model V2. Enhanced separation is observed in V2.

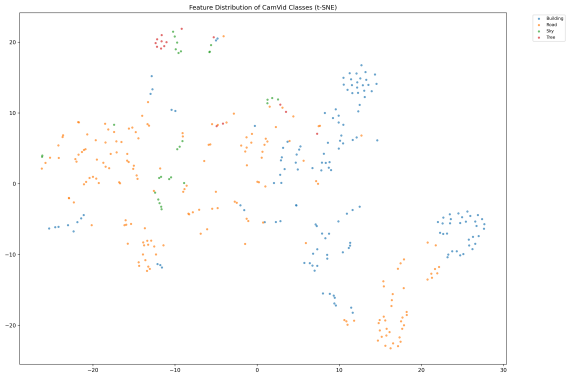

Figure 3: t-SNE feature distribution for Model V1.

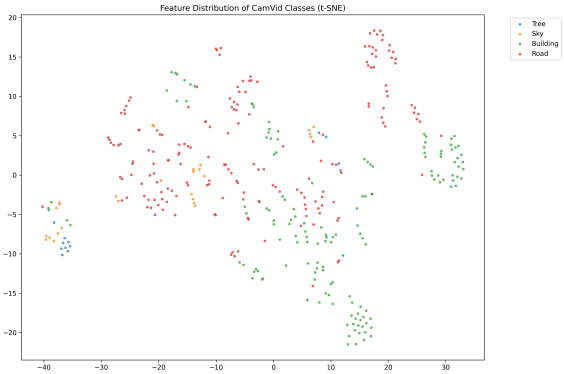

Figure 4: t-SNE feature distribution for Model V2.

4.2. Training Curves

Figures 5 and 6 show the training loss and Dice scores for both models.

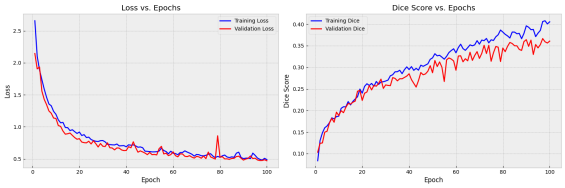

Figure 5: Training loss and Dice scores for Model V1.

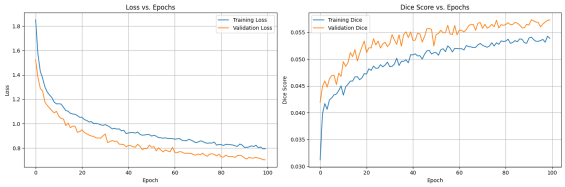

Figure 6: Training loss and Dice scores for Model V2.

4.3. Segmentation Examples

Figures 7 and 8 compare predictions for both models on sample images.

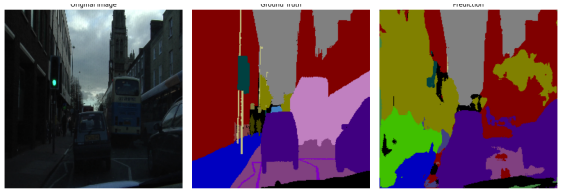

Figure 7: Model V1 segmentation results. Left: Original image. Middle: Ground truth. Right: Prediction.

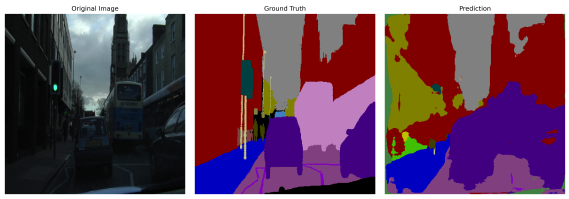

Figure 8: Model V2 segmentation results. Left: Original image. Middle: Ground truth. Right: Prediction.

4.4. Quantitative Results

Table 2 shows the Dice scores for key classes.

Class	Model V1	Model V2	Dilated
Sky	0.954	0.921	0.9451
Road	0.928	0.895	0.9153
Building	0.768	0.742	0.7879
Sidewalk	0.765	0.731	0.6508

Table 1: Dice scores for key classes.

4.5. Dilation Results

First, we compare the original and optimized UNets to dilated versions of each.

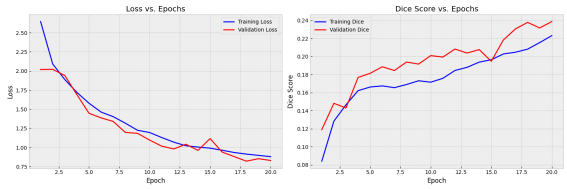

Figure 9: Dilated training and Dice score curves.

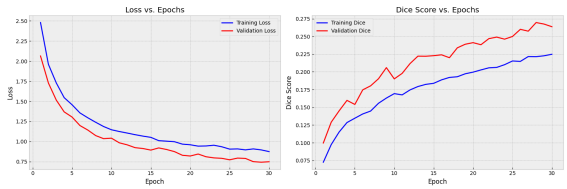

Figure 10: Optimized Dilated training and Dice score curves.

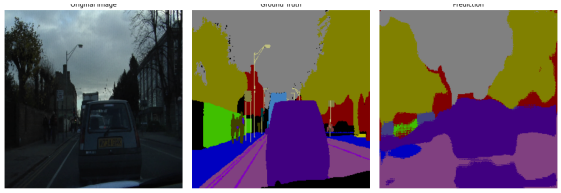

Figure 11: Dilated segmentation results. Left: Original. Middle: Ground truth. Right: Prediction.

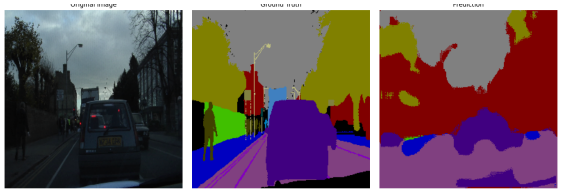

Figure 12: Optimized Dilated segmentation results. Left: Original. Middle: Ground truth. Right: Prediction.

4.6. FCN Results

Dice Scores:

Class	FCN256	FCN128	Baseline
Sky	0.9274	0.9209	0.9328
Road	0.9110	0.8837	0.8276
Building	0.7226	0.6913	0.5779
Sidewalk	0.6186	0.5260	0.3279

Table 2: Dice scores for key classes.

Prediction Images:

Hybrid FCN with 128-pixel image resolution:

Hybrid FCN with 256-pixel image resolution:

Hybrid FCN without batch normalization or dilation:

Baseline (non-hybrid) FCN:

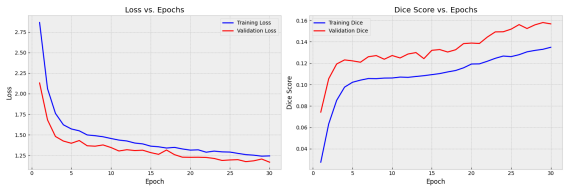

Figure 13: Baseline (non-hybrid) FCN training and Dice score curves.

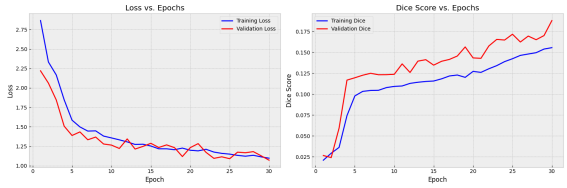

Figure 14: No BatchNorm training and Dice score curves.

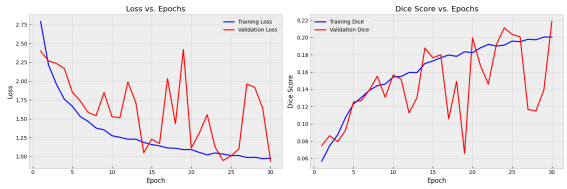

Figure 15: Hybrid FCN with 128-pixel image resolution training and Dice score curves.

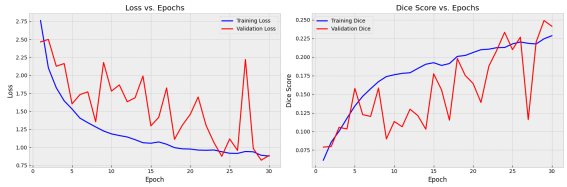

Figure 16: Hybrid FCN with 256-pixel image resolution training and Dice score curves.

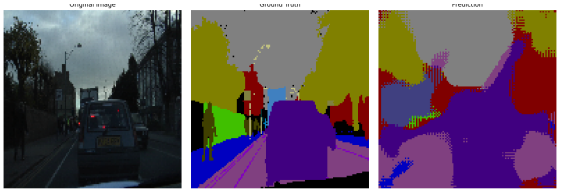

Figure 17: Hybrid FCN with 128-pixel image resolution segmentation results. Left: Original image. Middle: Ground truth. Right: Prediction.

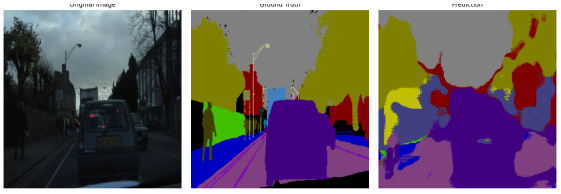

Figure 18: Hybrid FCN with 256-pixel image resolution segmentation results. Left: Original image. Middle: Ground truth. Right: Prediction.

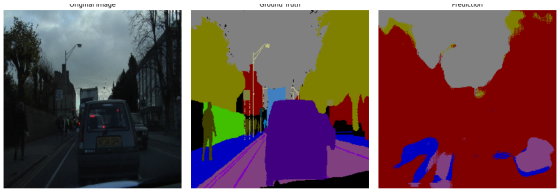

Figure 19: Baseline FCN sample segmentation result. Left: Original image. Middle: Ground truth. Right: Prediction.

5. Discussion

5.1. Limitations

Some of our models experienced overfitting, but others may have still benefited from training for more epochs. For the more complex hybrid model in particular, we began to run into computational limitations of the resources we had available, as we needed more compute to run for longer. In the future, we would extend our resources to train the more complex hybrid models for longer. For the time being, comparing the initial performance of a sizable group of models helps us understand which have potential and should be given more resources in the future.

5.2. Insights

From the eight network architectures we experimented with, it seemed that both the optimized U-Net and the FCN-256 hybrid model helped complete the task more accurately. They both improved on the U-Net and FCN baselines significantly for our dataset and each reached dice scores of above 0.9 for key classes.

The U-Net method and the FCN method both transferred well onto our dataset, particularly after we made modifications from the original methodology to enhance performance and incorporated hybrid versions.

Dilation was somewhat variable in success. It occasionally lowered the accuracy, especially for the optimized U-Net. The dilated regular U-Net appeared to improve on the non-dilated regular U-Net, but the optimized U-Net predictions without dilation more closely resembled the image than the dilated optimized U-Net predictions.

However, the opposite effect happened with FCN - dilation and batch normalization appeared to improve the performance significantly as compared to the model without either. These results suggest that dilation interacts better with some forms or architectures than with others.
Image size had a noticeable impact; the size-128 version of FCN performed considerably worse than the size-256 version.
Due to the outperformance of both baselines on the CamVid

dataset by two of our enhanced models, we conclude that our iterations were successful and corrected for the domain shift accurately, at least in this preliminary context. In the future, we would improve on our results by focusing in on the two best models, fine-tuning these two models further and training for longer.

6. Conclusion

We implemented and extended U-Net for urban scene segmentation using the CamVid dataset, and extended and modified the FCN architecture. Despite challenges, we provided enhancements such as residual connections, combined loss functions, dilation, and batch normalization, and saw that these improved feature learning and segmentation detail significantly. These experiments show that iterating on these current methods can make them workable and accurate in new domains.

6.1. Replicating the Results

The **results** folder that accompanies this report contains all eight of our experiments, their trained models, and their respective results.

To replicate the results for FCN256, FCN128, both dilated versions, the baseline FCN, and the no-batch-norm FCN directly in Colab, follow these steps:

1. Select the `colab_train` file corresponding to the model you want to replicate.
2. Upload the `camvid.zip` dataset to your Google Drive. *(This only needs to be done once, even if you run all eight models.)*
3. Upload the following files directly into Colab:
    - `dataset.py`
    - The intended network architecture file
4. Select the applicable import statement from the options.
5. Unzip the `camvid.zip` dataset and rename the folder to `data`.
6. Run the code in the Colab notebook.
7. When ready, run `colab_eval` on the appropriate resulting best model.

To replicate the results from Models V1 and V2, follow these steps:

1. **Setup Environment:** To set up the environment, install the required dependencies using the following command:

```
pip install -r requirements.txt
```

2. **Train Models:** To train the models, use the following commands:

```
# Train V1 model
python src/train.py

# Train V2 model
python src/train_v2.py
```

3. **Evaluate Models:** To evaluate the trained models, use the commands below:

```
# Evaluate V1 model
python src/evaluate.py

# Evaluate V2 model
python src/evaluate_v2.py
```

4. **Visualize Feature Distributions:** To visualize the feature distributions for each model, use the following commands:

```
# V1 features
python src/visualize_features.py

# V2 features
python src/visualize_features_v2.py
```

References

[1] G. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[2] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[3] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017.