

Pet Image Segmentation using Modified U-Net with MobileNetV2 Encoder

Devyn Miller
Chapman University
Orange, United States
devmiller@chapman.edu

Abstract—This work presents a comprehensive pipeline for the segmentation of images from the Oxford-IIIT Pet Dataset using a modified U-Net architecture with a pretrained MobileNetV2 encoder. The model achieves accurate pixel-level segmentation by distinguishing pets from backgrounds across diverse settings and poses. Key features include skip connections for preserving spatial information, data augmentation techniques, and transfer learning from ImageNet-pretrained weights. The model demonstrates improved segmentation capabilities with training accuracy reaching 0.8863 and validation accuracy of 0.8711 by epoch 12. While the model shows strong learning on training data, testing metrics reveal opportunities for further improvement in generalization. These findings provide insights into the application of U-Net architectures for pet image segmentation and highlight the importance of architectural choices and preprocessing strategies in achieving robust segmentation performance.

Keywords—image segmentation, U-Net, MobileNetV2, deep learning, computer vision, transfer learning

I. INTRODUCTION

The task at hand is to develop a comprehensive pipeline for the segmentation of images from the Oxford-IIIT Pet Dataset. The problem to be solved is the accurate segmentation of pet images, which involves distinguishing the pet from the background in a variety of settings and poses. This task is relevant due to its applications in various fields such as veterinary medicine, pet care, and animal welfare, where understanding the context and specifics of each animal can be crucial.

Segmentation is a complex problem in computer vision because it requires the model to understand and delineate the boundaries of objects at a pixel level. Traditional image processing techniques often fall short in handling the variability and complexity found in real-world images, such as those in the Oxford-IIIT Pet Dataset. This dataset presents challenges such as diverse appearances, poses, scales, and occlusions of pets, which necessitate a robust solution that can generalize well across different scenarios.

The plan to solve this problem involves using a deep learning approach, specifically a modified U-Net architecture with a pretrained MobileNetV2 as the encoder and a series of upsample blocks as the decoder. This architecture is well-suited for segmentation tasks due to its ability to capture both high-level features and fine-grained details through the use of skip connections and upsampling layers.

Deep learning is justified for this task because of its ability to learn hierarchical feature representations that are essential

for accurate segmentation. Unlike traditional methods that rely on handcrafted features, deep learning models can automatically learn to distinguish between the pet and the background from the data itself. This is particularly important for the Oxford-IIIT Pet Dataset, where the variability in the images requires a model that can adapt to new, unseen examples.

The pipeline will be facilitated by TensorFlow and Keras libraries and will include preprocessing steps such as normalization and resizing, data augmentation techniques to improve generalization, and training with a focus on optimizing the model's performance.

In summary, the problem of pet image segmentation is both relevant and challenging, and it requires a deep learning solution to achieve the desired level of accuracy and generalization. The proposed pipeline and model architecture are designed to address the complexities of the task, leveraging the power of U-Nets and transfer learning from pretrained models to create a robust segmentation model.

II. DATA

The Oxford-IIIT Pet Dataset [1], [2] is a comprehensive collection of pet images designed for fine-grained visual categorization tasks. It contains approximately 7,400 images of 37 distinct pet breeds, with around 200 images per breed. The dataset is well-balanced, ensuring that each class is adequately represented, which is crucial for training a robust segmentation model. However, the dataset is challenging due to the significant variation in scale, pose, and lighting conditions of the images. The dataset also includes XML files with head bounding box annotations in the PASCAL VOC format, which could be used for tasks that require precise localization of the pet's head. However, for the segmentation task, the primary focus is on the trimap annotations.

The trimaps are particularly valuable for segmentation tasks, as they provide a clear delineation between the pet (foreground) and the background, with three levels of annotation: foreground (1), background (2), and undefined (3). For convenience, the segmentation masks were adjusted to have labels in the set $\{0, 1, 2\}$, aligning with common practices in machine learning and simplifying the model's output interpretation. This normalization simplifies the model's output layer to predict three classes and aligns with the common practice of zero-indexing in machine learning tasks.

Exploratory Data Analysis (EDA) was conducted to gain insights into the dataset’s characteristics and to inform the preprocessing and model development stages. The EDA involved visualizing image examples alongside their corresponding segmentation masks. This visualization was facilitated by a custom `display` function, which plotted the input images and their true masks, providing a clear view of the expected outputs of the model.

Additional EDA work included printing details about the features and number of images in the dataset as well as the test and training set split.

In summary, the Oxford-IIIT Pet Dataset is well-suited for segmentation tasks due to its variety and complexity. The EDA provided valuable insights into the data, which will inform the preprocessing steps and the design of the deep learning model for segmentation.

III. METHODS: PREPROCESSING

In the development of the segmentation pipeline for the Oxford-IIIT Pet Dataset, preprocessing and augmentation play pivotal roles in preparing the data for input into the deep learning model. The first step in the preprocessing stage is data normalization. In this particular case, the input images are converted from integers to floating-point values within the range of 0 to 1 by dividing each pixel value by 255, the maximum possible value for an 8-bit color channel. This normalization process ensures consistency in pixel value ranges across the dataset, which can aid in the convergence and stability of the training process. The original images from the Oxford-IIIT Pet Dataset are resized to a fixed resolution of 128×128 pixels using the `tf.image.ResizeMethod.NEAREST_NEIGHBOR` method, which preserves the sharp edges and boundaries of the mask, preventing any blurring or interpolation artifacts.

Following preprocessing, the data enters the augmentation stage, which is designed to artificially increase the diversity of the training data and improve the model’s ability to generalize to unseen examples. The augmentation is implemented as a custom layer within the TensorFlow framework, which applies random horizontal flips to both the input images and their corresponding masks. This stochastic transformation enriches the dataset with variations in orientation, allowing the model to learn to recognize pets regardless of their horizontal alignment in the image.

In this pipeline, a custom layer is implemented, which applies random horizontal flipping to both the input images and their corresponding segmentation masks. This augmentation step introduces variations in the orientation of the objects within the images, forcing the model to learn features that are invariant to horizontal flipping. This augmentation layer is applied after batching the input data, which means that each batch of images may contain a mix of original and horizontally flipped images, further increasing the diversity of the training samples.

By carefully considering the preprocessing and augmentation requirements of the input data, this pipeline ensures

that the deep learning model receives high-quality, diverse, and properly formatted training samples. These steps are essential for achieving optimal performance and generalization capabilities, ultimately leading to more accurate and robust segmentation results.

IV. METHODS: MODEL

The model employed for the segmentation task is a modified U-Net architecture, which is well-suited for image segmentation due to its ability to capture context from the input image while preserving spatial information. The U-Net architecture (detailed figure found [here](#)) consists of an encoder, which downsamples the image and extracts features, and a decoder, which upsamples the feature maps to generate a segmentation map matching the input image’s resolution.

For the encoder, we utilize the MobileNetV2 architecture [3], pretrained on ImageNet, due to its balance between performance and computational cost. The pretrained network aids in faster convergence and improved performance by leveraging a robust set of features learned from a large and diverse dataset.

The outputs from several intermediate layers of the MobileNetV2 model (`'block_1_expand_relu'`, `'block_3_expand_relu'`, `'block_6_expand_relu'`, `'block_13_expand_relu'`, and `'block_16_project'`) are extracted and used as input to the decoder. These multi-scale features capture both high-level semantic information and low-level details, which is crucial for accurate segmentation.

The decoder consists of a series of upsample blocks implemented using transposed convolutions from the `pix2pix` library in the TensorFlow Examples repository. These blocks progressively recover the spatial resolution while combining the upsampled features with the corresponding encoder feature maps via skip connections. The skip connections allow for the network to use features from multiple resolutions which facilitates the propagation of low-level details, enabling the model to preserve spatial information and produce sharper segmentation masks.

The final layer is a 1×1 convolutional layer that maps the learned features to the desired number of output channels, which corresponds to the number of classes in the segmentation mask (3 in this case: pet, background, and undefined).

The model is compiled using the Adam optimizer which helps in converging to a solution more efficiently than traditional stochastic gradient descent. The sparse categorical cross-entropy loss is well-suited for this task as it directly compares the predicted class probabilities with the ground truth labels for each pixel (penalizing predictions that diverge from the actual labels), without the need for one-hot encoding. Additionally, the model’s performance is evaluated using the accuracy metric, which measures the proportion of correctly classified pixels.

During training, data augmentation techniques are employed to increase data diversity and improve generalization. The model is trained using a combination of training and validation datasets, with the validation dataset used to monitor performance and prevent overfitting. The training process is

initiated for up to 50 epochs, with the model’s performance being evaluated at the end of each epoch using the validation dataset. Early stopping is implemented, allowing the training to terminate before reaching the maximum specified epochs if the model’s performance ceases to improve on the validation dataset.

Overall, the model architecture combines the strengths of the U-Net architecture with the efficient feature extraction capabilities of the MobileNetV2 model, enabling accurate and robust segmentation of pet images from the Oxford-IIIT Pet Dataset.

V. RESULTS

The evaluation of the model’s performance on the Oxford-IIIT Pet Dataset is depicted through several metrics, complemented by visual representations such as loss and accuracy plots. The data is organized to distinctly separate the training, validation, and testing phases, with each metric associated with the respective phase.

During the training phase, the model consistently improved its ability to segment images from the dataset. The training loss steadily decreased from an initial value of 0.7944 to 0.2920 by the 12th epoch, indicating that the model’s predictions increasingly matched the actual segmentation masks. Simultaneously, the training accuracy rose from 0.6767 to 0.8863, demonstrating the model’s improved pixel-wise classification capability.

However, the validation phase presented a more nuanced picture. Initially, the validation loss was exceptionally high at 2213.6946, likely due to the model’s lack of tuning for the validation data. This suggests initial model inadequacies or possible inconsistencies within the validation data itself. This validation loss significantly decreased as the model started to generalize, but fluctuations in later epochs hinted at potential overfitting. Nevertheless, the validation accuracy exhibited a promising increase, reaching 0.8711 by the 12th epoch, indicating successful learning on the validation set. However, the fluctuations in validation loss and the high initial loss value highlight areas for potential refinement.

Furthermore, the significant gap in segmentation reliability is illustrated by the disparity between training and testing performance metrics, with a precision of 0.3965, recall of 0.3293, F1 score of 0.1876, IoU of 0.1124, accuracy of 0.2064, and loss of 1.4191 on the testing set. This disparity suggests that while the model has learned to identify segments within images, it falls short of dependable segmentation accuracy on unseen data. This is highlighted by the incongruently high training accuracy when compared to the low IoU and F1 score on the testing set, raising flags about the model’s true predictive power.

To elucidate the model’s operational focus and potential shortcomings, supplementary visual aids including 3 best 3 worst and Grad-CAM heatmaps are integrated. The best and worst predictions are determined by their accuracy scores, reflecting the model’s ability to match the predicted segmentation masks with the ground truth. By examining the best cases,

we can learn about the scenarios where the model excels, which helps in identifying the features and patterns the model is leveraging effectively. Conversely, the worst cases highlight the model’s limitations and the challenges it faces, such as complex backgrounds or varying animal poses, offering a direct line of sight into potential areas of improvement.

Grad-CAM heatmaps are used to visualize which regions of the input image are most influential for predicting a specific class. This is particularly useful to verify whether the model is focusing on relevant areas of the image for making its decisions or if it is being misled by irrelevant patterns or noise. By overlaying these heatmaps on the input images, we can interpret the model’s attention and ensure it aligns with the salient features for accurate segmentation. These insights are paramount in refining the model further, as they can inform adjustments to the architecture, training procedure, or data preprocessing steps.

VI. PLOTS AND FIGURES

The following tables summarize the key performance metrics across training, validation, and testing phases:

Training Phase	
Initial Training Loss	0.7944
Final Training Loss by Epoch 12	0.2920
Initial Training Accuracy	0.6767
Training Accuracy by Epoch 12	0.8863

Validation Phase	
Initial Validation Loss	2213.69
Final Validation Loss by Epoch 12	0.5007
Initial Validation Accuracy	0.3067
Validation Accuracy by Epoch 12	0.8711

Testing Phase	
Precision	0.3965
Recall	0.3293
F1 Score	0.1876
IoU	0.1124
Accuracy	0.2064
Loss	1.4191

Table 1: Performance metrics across training, validation, and testing phases

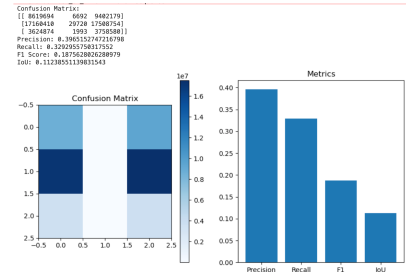


Fig. 1: Confusion matrix for the U-Net model

Note: the model accuracy and loss graphs in my [Jupyter Notebook](#) look a little different but contain the same data. I am just using the graphs I generated in the original non-modular Jupyter Notebook.

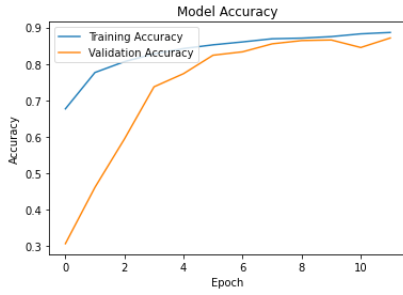


Fig. 2: Training and validation accuracy over epochs

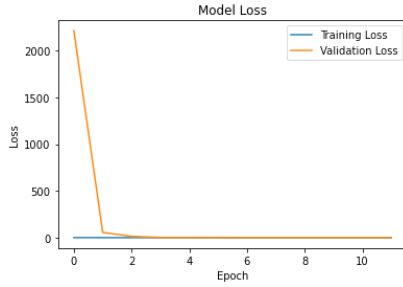


Fig. 3: Training and validation loss over epochs

VII. DISCUSSION

The performance metrics of the model, as observed from the training and validation results, provide a comprehensive view of its capabilities in addressing the image segmentation problem within the Oxford-IIIT Pet Dataset. The consistent improvement in training accuracy and loss indicates that the model is learning the segmentation task effectively. However, the fluctuations in validation loss suggest that the model may be sensitive to the validation dataset's variability or may be beginning to overfit the training data.

In addition to the base model, three other, simpler models were tested, each yielding different results. The first model structure variation, a conventional convolutional neural network (CNN), achieved a loss of 1.7101 and an accuracy of 0.5074 on the testing data. This model's relatively high loss and modest accuracy suggest that it may lack the necessary depth or complexity to capture the intricate patterns required for precise segmentation. The CNN model's performance is inferior compared to the original model, which utilized a more complex U-Net architecture with a pretrained MobileNetV2 encoder. The original model's skip connections and upsampling layers, which are characteristic of the U-Net architecture, likely contribute to its superior ability to localize and delineate the segmentation boundaries more accurately.

The second model structure variation (found [here](#)), a streamlined version of the U-Net architecture, achieved an accuracy of 0.5566 and a loss of 0.1304, indicating a performance improvement over the basic CNN model. The simplified U-Net retains the core structure that makes U-Net architectures effective for segmentation tasks, such as skip connections that help recover spatial information during upsampling. However,

when compared to the original model, the simplified U-Net may not capture as much contextual information due to its reduced complexity, which could explain its lower accuracy and higher loss.

The third model structure variation (found [here](#)) integrates the MobileNetV2 architecture within a U-Net framework. The training loss decreased to around 0.15 and the training accuracy exceeded 0.95. The validation accuracy also increases to approximately 0.92, with some fluctuations in the validation loss. These results indicate that the MobileNetV2-based U-Net model is highly effective at learning the segmentation task and generalizes well to validation data. When compared to the original model, the MobileNetV2-based U-Net shows a similar or slightly improved performance, likely due to the efficient feature extraction capabilities of MobileNetV2 and the depth of the U-Net architecture.

In summary, the original model with the pretrained MobileNetV2 encoder within a U-Net architecture appears to provide a good balance between complexity and performance, as evidenced by its accuracy and loss metrics. The basic CNN model lacks the architectural sophistication required for high performance in segmentation tasks, while the simplified U-Net offers a middle ground, retaining some of the U-Net's beneficial properties but with reduced complexity. The MobileNetV2-based U-Net variation demonstrates that a well-designed integration of a powerful feature extractor with a robust segmentation architecture can lead to high performance, potentially matching or exceeding that of the original model.

Looking forward, several scenarios can be considered to understand how the model's performance might be affected. Adjusting the learning rate of the Adam optimizer could potentially lead to different convergence behaviors. A lower learning rate might result in slower but more stable convergence, possibly leading to better generalization on the validation set. Conversely, a higher learning rate could accelerate the initial learning phase but might also cause the model to overshoot the optimal solution, resulting in higher validation loss or instability.

Another parameter that could be modified is the batch size. The current model uses a batch size of 32, which is a balance between computational efficiency and the ability to generalize. Increasing the batch size could lead to faster training times and more stable gradient estimates, but it might also require more memory and could potentially lead to poorer generalization if the batch size is too large. Decreasing the batch size could improve generalization by providing noisier gradient updates, but it would also slow down the training process and might lead to convergence issues if the batch size is too small.

Furthermore, introducing additional augmentation techniques such as random rotations, scaling, or color jittering could potentially increase the robustness of the model to various transformations and perturbations in the input data. However, excessive augmentation might also lead to a model that is too invariant to the features that are actually important for the segmentation task, thereby reducing its performance.

Several challenges were encountered when building the

pipeline for the main model, such as the need to modularize the pipeline, which required changes to the model's structure and dependencies. This experience furthered my understanding of best practices when structuring deep learning projects. Additionally, I have continued to learn that deep learning models for image segmentation can achieve high accuracy, but careful attention must be paid to the risk of overfitting, as evidenced by the validation loss fluctuations. The choice of parameters, data augmentation strategies, and the overall model can significantly impact the model's performance, and these choices must be made judiciously to balance the trade-offs between training efficiency, model accuracy, and generalization.

Overall, my main model's computational efficiency was quite low. Therefore, I would recommend improvement through parameter tuning and augmentation strategies as well as more thorough investigation into its generalization across different datasets and under various conditions. While the model's architecture leverages a pretrained MobileNetV2 encoder and a U-Net-like decoder for the segmentation task, I do not think it should be used as-is. I would instead recommend building off of the architecture from the MobileNetV2-based U-Net variation due to its greater computational efficiency. Future work could involve experimenting with the overall model architecture as well as the aforementioned parameter adjustments, implementing additional regularization techniques, and conducting extensive testing to ensure the model's robustness and reliability in real-world applications. The exploration of different model architectures for the segmentation of the Oxford-IIIT Pet Dataset has yielded varied results, each providing insights into the impact of architectural choices on model performance.

REFERENCES

- [1] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," <https://www.robots.ox.ac.uk/~vgg/data/pets/>, 2012, accessed: 2024.
- [2] TensorFlow, "Oxford-iiit pet dataset," https://www.tensorflow.org/datasets/catalog/oxford_iiit_pet, 2024, accessed: 2024.
- [3] Keras, "Mobilenetv2," <https://keras.io/api/applications/mobilenet/>, 2024, accessed: 2024.