Computer Vision Mini Project: Image Segmentation

(s1707068, s2737499)

Abstract

This report presents a comprehensive implementation and analysis of image segmentation techniques applied to the Oxford-IIIT Pet Dataset (Parkhi et al., 2012). We explore four distinct approaches: a UNet-based end-to-end network architecture, an autoencoder-based segmentation, CLIP feature-based segmentation, and prompt-based segmentation with point and text prompts. Our implementation includes extensive data preprocessing and augmentation techniques which significantly improved model performance across all architectures. Using CLIP features for segmentation improves over autoencoder-based features, which highlights the impact of pretraining on large-scale datasets. UNet achieves the highest performance with a mean IoU of 0.854 and a mean Dice score of 0.920 on the test set, significantly outperforming both the autoencoder-based and CLIP feature-based segmentation. We further extend our best-performing model with prompting capabilities, demonstrating that both point and text prompts can improve segmentation performance while allowing for interactive applications. Additionally, we conduct a comprehensive robustness analysis by subjecting our best model to eight different perturbation types of increasing intensity, revealing particular vulnerabilities to salt & pepper noise and contrast reduction, while demonstrating strong resilience to brightness variations, occlusion, and Gaussian perturbations. Through these experiments, we gain valuable insights into the strengths and limitations of each segmentation approach and the impact of data augmentation on model robustness.

1. Introduction

Image segmentation is a fundamental computer vision task that involves partitioning images into meaningful regions, a crucial step in many applications including medical imaging, autonomous driving, and scene understanding. In this work, we present the implementation and analysis of various image segmentation techniques applied to the Oxford-IIIT Pet Dataset (Parkhi et al., 2012). Our project explores four distinct approaches to image segmentation: UNet-based end-to-end network, Autoencoder-based segmentation, CLIP feature-based segmentation, and Prompt-based segmentation using point and text prompts.

For each approach, we detail the network architecture, training methodology, and performance metrics on the test set. Additionally, we conduct a comprehensive robustness analysis by subjecting our best-performing model to different perturbation types of increasing intensity. The objectives of this work are to compare the effectiveness of traditional and modern deep learning segmentation approaches, investigate how different pre-training and feature extraction strategies influence performance, and assess model robustness to various image perturbations. Through these experiments, we aim to gain insights into the strengths and limitations of each model architecture on the image segmentation task.

2. Dataset Preprocessing and Augmentation

The project leverages the Oxford-IIIT Pet Dataset (Parkhi et al., 2012), which consists of images of cats and dogs with corresponding segmentation masks. The dataset contains three classes: Background (class 0), Cat (class 1), and Dog (class 2), with an additional ignore index for border regions. The original dataset is provided with *TrainVal* and *Test* splits each containing separate directories for images (*color*) and masks (*label*), which we further processed into proper training, validation, and test sets.

2.1. Data Preprocessing

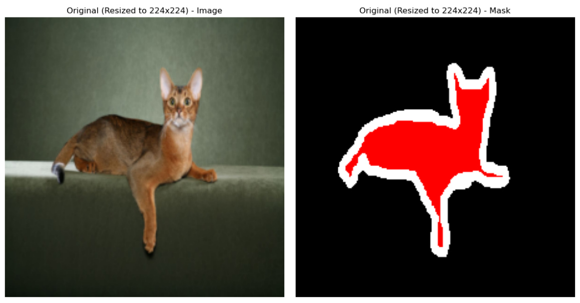
First, we implemented an 80/20 train/validation split on the original *TrainVal* set, ensuring a balanced distribution of classes. We standardised all images to a uniform size of 224×224 pixels. This specific resolution was chosen to align with CLIP's expected input size, as CLIP models were originally trained on 224×224 images and have fixed positional embeddings optimised for this resolution, which is crucial for our CLIP feature-based segmentation approach. For mask processing, the original masks are provided as RGB images where different colours represent different classes. We processed these masks to create single-channel segmentation maps with integer class indices. Black pixels (RGB: 0,0,0) were mapped to Background (class 0), red pixels (RGB: 128,0,0) to Cat (class 1), green pixels (RGB: 0,128,0) to Dog (class 2), and white pixels (RGB: 255,255,255) to Border which is ignored in the loss function.

2.2. Dataset Imbalance

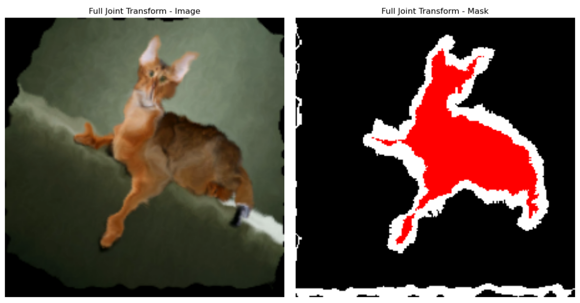
We observed a notable imbalance in our dataset, as there are approximately 2.1 times more dog instances that cat instances. Consequently, the models trained on this dataset could be biased and predict more dog pixels than cat pixels. To mitigate this imbalance, we employed data augmentation to provide the model with a more diverse training set of cat and dog images. Additionally, we later applied class weights to the cross-entropy loss function to assign a greater weight to the Cat class during model training.

2.3. Data Augmentation

To improve model generalisation, we implemented a comprehensive data augmentation strategy that applies consistent transformations to both images and their corresponding masks. Our augmentation pipeline incorporates several geometric transformations, which alter the geometrical structure of images without modifying pixel values (Mumuni & Mumuni, 2022). These include random horizontal flipping with 50% probability, random rotation between -30°and +30°with 30% probability, random cropping from 224×224 to 196×196 followed by resizing, random translation up to 10% of image dimensions, and elastic deformations with 30% probability using parameters α=25 and σ=3. For colour and intensity variations, we applied image-only transformations including random brightness, contrast, and saturation adjustments of ±20%. These transformations help the model become robust to lighting and exposure variations while preserving the semantic content of the images, as demonstrated by the consistent transformation of both images and masks in Figure 1. Our augmentation strategy generated 3 additional augmented versions for each original training image, effectively quadrupling the size of the training set. Validation and test sets were kept unaugmented to ensure fair evaluation of model performance.


Original (Resized to 224x224) - Image
Original (Resized to 224x224) - Mask

(a) Original (224×224)



(b) Full Joint Transform

Figure 1. Examples of joint transform applied consistently to both input image (left) and its segmentation masks (right)

2.4. Implementation Details

We implemented the preprocessing and augmentation pipeline using PyTorch's (Paszke et al., 2019) transformation utilities, with special attention to maintaining consistency between images and masks. Our implementation centres around a custom *JointTransform* class that applies the same geometric transformations to both image and mask, ensuring pixel-level correspondence critical for segmentation tasks. We extended PyTorch's Dataset class to create a *PetDataset* class for efficient data loading and preprocessing. Special consideration was given to mask transformations, where we used nearest-neighbour interpolation to preserve discrete class labels that would otherwise be corrupted by bilinear or bicubic methods. The pipeline is orchestrated by two main functions: `create_augmented_dataset`, which performs the train/val split and generates augmented samples, and `get_data_loaders`, which creates PyTorch DataLoader objects with appropriate batch sizes and worker configurations.

2.5. Final Dataset Summary

Our preprocessed and augmented dataset consists of 11776 training examples with augmentation (effectively 4× the original training set size), 736 validation images, and 3710 unchanged test images. This preprocessing and augmentation approach provided a robust foundation for training our segmentation models, improving model performance and generalisation by increasing the volume and diversity of the training data. The augmentation parameters were carefully selected to create realistic variations while preserving the semantic content of the images. Importantly, all transformations were applied jointly to both images and masks to maintain pixel-level correspondence, which is crucial for segmentation tasks.

3. Network Design and Implementation

The following networks have been designed to process RGB images (3 channels) of size 224×224 pixels and output segmentation masks with 3 channels corresponding to our target classes: background (class 0), cat (class 1), and dog (class 2). Cross-entropy is employed as the loss function for all segmentation models as it measures the difference in distribution between the true label and the predicted mask and outputs the class probability for each pixel in this multi-class setting. We ignored the white border pixels in the loss function and only considered the Cat, Dog, and Background classes.

3.1. UNet

Our UNet implementation follows the well-established encoder-decoder architecture with skip connections originally proposed by (Ronneberger et al., 2015), adapted specifically for the pet segmentation task. Figure 2 illustrates the overall architecture, showing the symmetric U-shaped structure that gives the network its name.

The foundational building block of the design is the **DoubleConv** module, comprising two sequential 3×3 convolutional layers, each followed by batch normalisation and ReLU activation. This module appears throughout the network and serves to extract and refine features at each resolution level. We included batch normalisation as an enhancement to the original UNet design based on the work of (Ioffe & Szegedy, 2015), which stabilises training and accelerates convergence. Following (He et al., 2015), we omitted bias terms from convolutional layers when followed by batch normalisation, as they become redundant. ReLU activations are applied with the inplace parameter to reduce memory consumption during training, an important consideration given the large feature maps in the UNet architecture.

3.1.1. UNet - Encoder Component

Our network's encoder path progressively downsamples the input image while increasing feature channel depth. Starting with the input image (224×224×3), the initial level processes it through a DoubleConv block to produce 32 feature maps at the same spatial resolution. We deliberately chose 32 as our initial feature dimension, rather than 64 as in the original UNet, to balance model capacity with computational

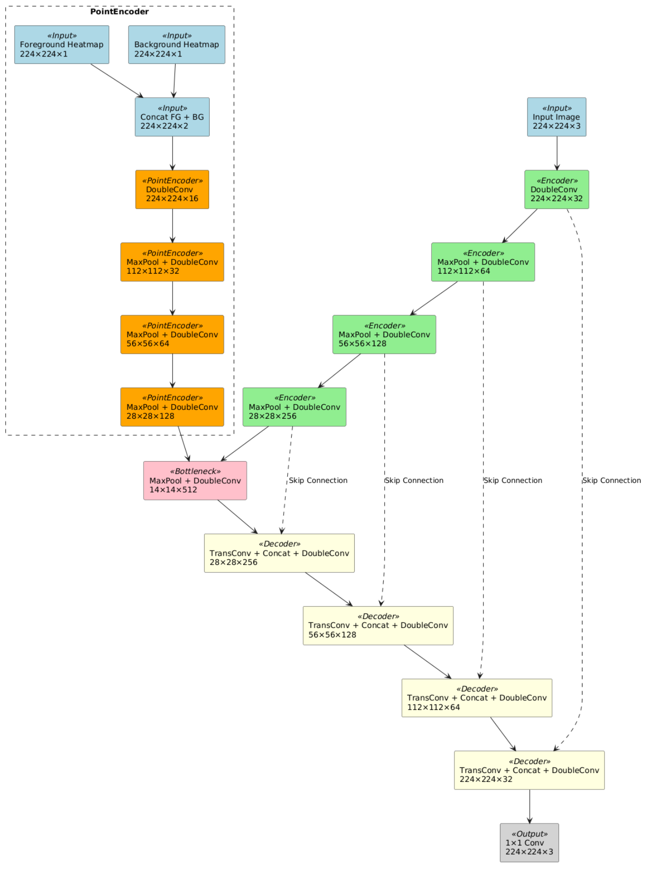

PointEncoder
«Input» Foreground Heatmap 224×224×1
«Input» Background Heatmap 224×224×1
«Input» Concat FG + BG 224×224×2
«PointEncoder» DoubleConv 224×224×16
«PointEncoder» MaxPool + DoubleConv 112×112×32
«PointEncoder» MaxPool + DoubleConv 56×56×64
«PointEncoder» MaxPool + DoubleConv 28×28×128
«Input» Input Image 224×224×3
«Encoder» DoubleConv 224×224×32
«Encoder» MaxPool + DoubleConv 112×112×64
«Encoder» MaxPool + DoubleConv 56×56×128
«Encoder» MaxPool + DoubleConv 28×28×256
«Bottleneck» MaxPool + DoubleConv 14×14×512
Skip Connection
Skip Connection
Skip Connection
Skip Connection
«Decoder» TransConv + Concat + DoubleConv 28×28×256
«Decoder» TransConv + Concat + DoubleConv 56×56×128
«Decoder» TransConv + Concat + DoubleConv 112×112×64
«Decoder» TransConv + Concat + DoubleConv 224×224×32
«Output» 1×1 Conv 224×224×3

Figure 2. UNet Architecture Diagram. The dashed rectangle is the additional point encoder for the point-based model in §5.1.

efficiency. This design choice follows modern efficient UNet variants (Zhou et al., 2018), allowing us to achieve good performance with fewer parameters. The encoder continues with four sequential downsampling operations using max pooling with stride 2, reducing spatial dimensions by half at each level. We selected max pooling over strided convolutions based on empirical evidence (Badrinarayanan et al., 2017) suggesting it better preserves important features for segmentation tasks. Each downsampling operation is followed by a DoubleConv block that doubles the number of feature channels, compensating for spatial information loss with richer feature representations (Long et al., 2015). The encoder's spatial dimensions and feature channels progress as follows: from 224×224×32→ 112×112×64→ 56×56×128→ 28×28×256→ 14×14×512, reaching the bottleneck. This bottleneck creates an information constraint that forces the network to extract the most salient features (Tishby & Zaslavsky, 2015), while the high channel dimension ensures sufficient capacity for representing complex boundaries between cats, dogs, and background regions.

3.1.2. UNet - Decoder Component

The decoder path mirrors the encoder in reverse, using transposed convolutions with kernel size 2 and stride 2 for upsampling. These transposed convolutions learn the optimal upsampling parameters during training, making them more effective than fixed interpolation methods for preserving fine details in pet fur and boundaries (Zeiler et al., 2010; Noh et al., 2015). Following each upsampling operation, feature maps from the corresponding encoder level are concatenated with the upsampled features via skip connections, allowing the decoder to recover spatial details that would otherwise be lost during downsampling. The decoder progresses through feature maps of increasing spatial resolution and decreasing channel depth: from 14×14×512→28×28×256→56×56×128→112×112×64→224×224×32, the original image size. At each level, a DoubleConv block processes the concatenated features. The final stage applies a 1×1 convolution that maps the 32 feature channels to the 3 output channels, essentially making a class prediction for each pixel location. This 1×1 convolution functions as a pixel-wise classifier, determining for each spatial position whether it belongs to background, cat, or dog class (Long et al., 2015).

3.2. Autoencoder

Originally designed for dimensionality reduction, autoencoders have since been adapted for various tasks such as image compression, image denoising, and image generation. The objective of the autoencoder is to reconstruct its own input; in other words, the target output of the network is simply the original input. This is particularly advantageous as it allows the model to learn a representation of the image without requiring any annotated labels.

An autoencoder consists of two components: an encoder which constructs a low-level representation of the input $h = encoder(x)$ and a decoder which reconstructs the original input from the latent representation $\tilde{x} = decoder(h)$. During training, the model aims to align the output $\tilde{x}$ to the input x and learns useful properties of the input images in the process (Goodfellow et al., 2016).

As illustrated in Figure 3, we leveraged the autoencoder to obtain low-level and mid-level features of the image, which can then be used to train a separate segmentation decoder. Our autoencoder architecture closely resembles the design in (Bagheri & Mohsenzadeh, 2024). The latent dimension of 256 is used to allow sufficient dimensionality reduction while still capturing complex features in the image. Mean Squared Error (MSE) is employed as the loss function for the autoencoder to directly compute the squared difference between an original image and its reconstruction, while cross-entropy loss is used to train the segmentation decoder for pixel-wise classification.

3.2.1. Autoencoder - Encoder Component

The encoder has five blocks, and each block consists of a convolutional layer with batch normalisation, followed by the ReLU activation function and max pooling. Within each block, the convolutional layer extracts spatial features from the image by applying a kernel to 3×3 regions in the image. Earlier layers learn simple features such as edges, while later layers learn more complex patterns in the input. Across the five encoder blocks, the convolutional layers increase the number of channels from 3 (RGB)→32→64→128→256.

Batch normalisation is then applied to the outputs of the convolutional layers, which normalises the activations in each mini-batch to zero mean and unit variance. This effectively mitigates the *internal covariate shift* and acts as regularisation to the network which enables more stable training (Ioffe & Szegedy, 2015).

The ReLU activation function $f(x) = max(0, x)$ introduces non-linearity in the network which enables it to learn complex

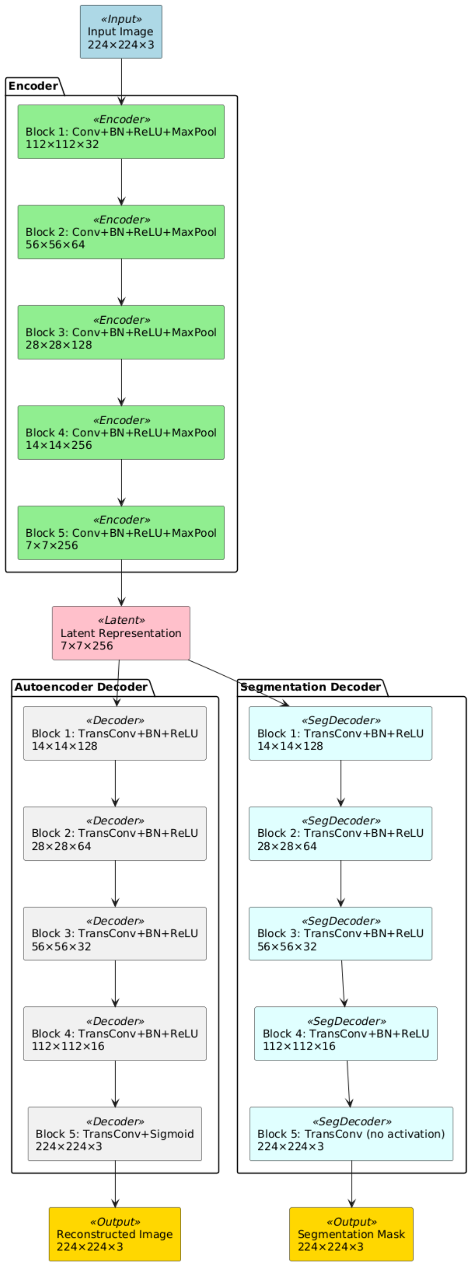


Figure 3. Autoencoder Architecture with Segmentation Decoder

relationships. We chose ReLU over other activation functions because it is most commonly used in the literature.

Finally, max pooling layers are applied to iteratively downsample the feature maps. In our network, max pooling outputs the maximum value in each 2×2 region with a stride of 2 to avoid overlapping regions. This effectively reduces the dimensions by half, thus over the five encoder blocks, the original image size is downsampled from 224×224→112×112 →56×56→28×28→14×14→7×7.

The encoder component results in a latent representation with the shape 7×7×256, where 256 is the latent dimension and 7 is the downsampled image size, which is 12 times smaller than the original input size of 224×224×3.

3.2.2. Autoencoder - Decoder Component

The decoder mirrors the encoder with five sequential blocks, however it applies upsampling starting from the latent representation of shape 7×7×256 to reconstruct the original input shape of 224×224×3. Each decoder block consists of a transposed convolution, batch normalisation, and ReLU, except for the last block which consists of a transposed convolution and a sigmoid layer.

The transposed convolution upsamples the feature maps by reversing the downsampling effect of the convolutional and max pooling layers in the encoder. Using a 4×4 kernel and stride of 2 doubles the spatial dimensions in each decoder block, which iteratively increases the dimensions from 7×7→14×14→28×28→56×56→112×112→224×224 to restore the original image size. At the same time, the number of channels is decreased from 256→128→64→32→3.

Similar to the encoder, batch normalisation is used to ensure activation values stay within a reasonable range and stabilise training, while ReLU is used to allow non-linear transformations. At the end of the decoder, the sigmoid function is used as the final activation function to ensure output pixel values are between 0 and 1. Figure 4 illustrates the reconstructed images as the output of the autoencoder.

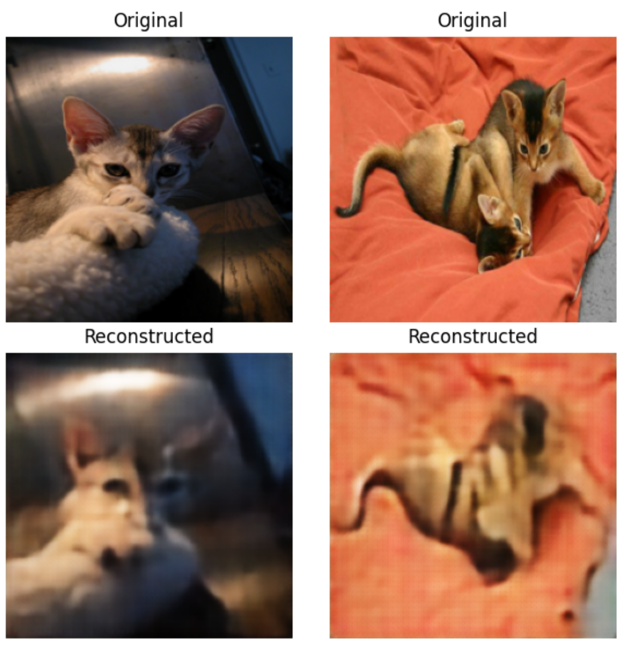


Figure 4. Reconstructed images by the Autoencoder

3.2.3. Segmentation Decoder

The segmentation decoder takes the latent representation of the frozen autoencoder's encoder as input to train for the image segmentation task. The architecture of the segmentation decoder is similar to the autoencoder's decoder, with the only difference being the absence of the final sigmoid activation as the cross-entropy loss function expects unnormalised logits as input. There are five upsampling blocks in the segmentation decoder, each doubling the spatial dimensions and reducing the number of channels by half. These blocks iteratively upsample the latent representation of size 7×7×256 to the output of size 224×224×3. Unlike in the autoencoder's decoder, here 3 is the number of output classes and not the number of input channels.

3.3. CLIP

The third segmentation approach leverages OpenAI's CLIP (Contrastive Language-Image Pre-training) model (Radford

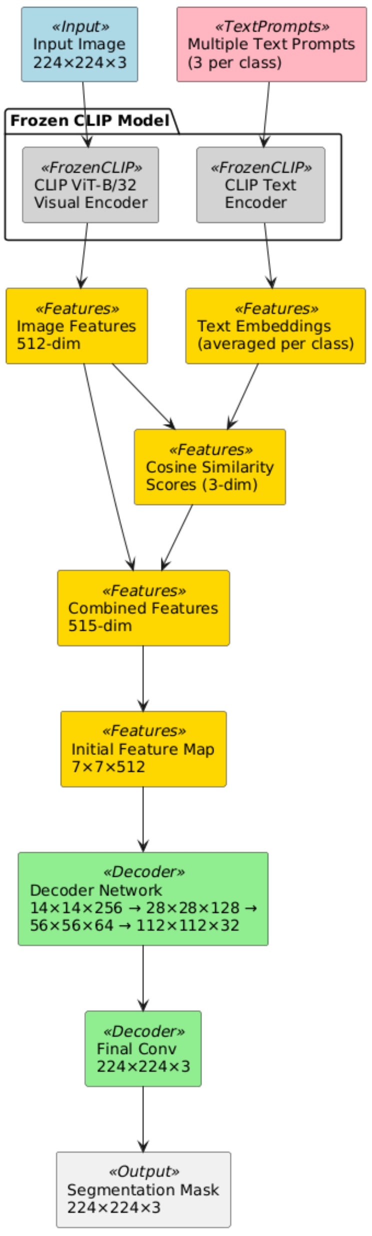


Figure 5. CLIP Architecture Diagram

et al., 2021) as a foundation for segmentation. CLIP was originally designed for image-text matching and zero-shot image classification tasks rather than dense prediction, making our adaptation of its features for segmentation an exploration of its versatility beyond its intended use case. The overall architecture in Figure 5 shows the hybrid approach that keeps the CLIP encoder frozen while training a custom decoder network to produce pixel-wise segmentation from CLIP's semantic features.

Our CLIP-based segmentation model processes images through CLIP's ViT-B/32 visual encoder to extract 512-dimensional feature vectors. These image features are compared with text embeddings representing each class (background, cat, dog) using cosine similarity. Both the image features and similarity scores are fed to a trainable decoder network that progressively upsamples them into a full-resolution segmentation mask. This approach leverages CLIP's pre-trained visual-textual representations while adapting them for dense prediction.

We initially experimented with simple, single prompts per class ("this is the background", "this is a cat", "this is a dog"). While this approach achieved reasonable results (validation Mean IoU: 0.6988), we found that using multiple complementary prompts per class improved performance slightly (validation Mean IoU: 0.7054). Following (Zhou et al., 2022), our multiple-prompt approach uses three descriptions per class that capture different aspects: general identity, distinctive features, and overall shape.

- Background Prompts
  - Background region with no pets
  - The area surrounding
  - Scenery, furniture, floors, or walls
- Cat Prompts
  - A domestic cat with fur and whiskers
  - A feline pet with pointed ears and a tail
  - The complete shape and outline of a cat
- Dog Prompts
  - A domestic dog with fur and a snout
  - A canine pet with distinctive ears
  - The complete shape and outline of a dog

For each class, we compute the average of normalised embeddings from these prompts to create a more robust representation, similar to the ensemble approach described by (Li et al., 2022).
CLIP's ViT-B/32 encoder produces global feature vectors without inherent spatial information, creating a unique challenge for segmentation tasks. Unlike traditional CNNs designed for dense prediction, CLIP lacks the spatial feature maps that typically feed into segmentation decoders. Our decoder architecture addresses this fundamental challenge through a carefully designed reconstruction and upsampling pathway. We begin by combining CLIP's 512-dimensional image features with the 3-dimensional class similarity scores to create a rich 515-dimensional semantic representation that captures both visual content and class relationships. This combined representation is then projected into a 7×7×512 feature map through a learned upscaling operation, where the 7×7 spatial dimension is not arbitrary but corresponds to CLIP ViT-B/32's effective receptive field when processing 224×224 images, as identified by (Amir et al., 2021). From this initial feature map, we implemented a progressive upsampling strategy through a series of transposed convolutional blocks that systematically increase spatial resolution while decreasing channel depth: from 7×7×512→ 14×14×256→ 28×28×128→ 56×56×64→ 112×112×32→224×224×3, the original image size. Throughout this upsampling chain, each block incorporates batch normalisation and ReLU activation (Ioffe & Szegedy, 2015; He et al., 2015) to maintain stable training dynamics. The relatively high initial channel count preserves CLIP's rich semantic information during the early upsampling stages, while later stages gradually reduce dimensionality as spatial resolution increases. This architecture effectively bridges the semantic-rich but spatially-poor CLIP features to the dense spatial predictions required for segmentation, following principles established by (Xu et al., 2022) for adapting transformer features to dense prediction tasks.

4. Results and Analysis

For each model architecture, we ran each experiment for either 20 or 50 epochs and saved the checkpoint with the lowest validation loss. We fixed the seed for `numpy`, `random`, and `torch` to ensure reproducibility and fairness across the runs for each model.

Throughout training, we monitored the pixel-wise accuracy, IoU (Intersection over Union) score, and Dice score on the validation set for each class. Mean IoU is used as the primary evaluation metric due to its effectiveness in capturing segmentation quality for imbalanced classes (Csurka et al., 2013).

4.1. UNet

UNet models were trained for 50 epochs using the `Adam` optimiser with an initial learning rate of 1e-4, chosen for its faster convergence in deep convolutional networks compared to traditional SGD (Kingma & Ba, 2017). We implemented a learning rate scheduler that reduced the rate by a factor of 0.1 when validation performance plateaued for 5 consecutive epochs, following best practices for training deep networks (Smith, 2017). We used a batch size of 16 to balance between computational efficiency and optimisation stability, noting that excessively large batch sizes can negatively impact generalisation (Keskar et al., 2016).

To address the class imbalance in our dataset described in §2.2, we hypothesised that a weighted loss function might improve segmentation performance. There were approximately 2.1 times more dog instances that cat instances in the dataset, leading us to implement a weighted cross-entropy loss that applied a weight of 2.1 to the cat class while maintaining default weights of 1.0 for dog and background classes, following approaches suggested for handling class imbalance (Sudre et al., 2017). However, as shown in Table 1, our experiments revealed that data augmentation has a more substantial impact on addressing class imbalance that the weighted loss function. The augmented models consistently outperformed the unaugmented models across all metrics. This suggests that diverse training examples through augmentations were more effective in helping the model learn under-represented classes than explicitly weighting the loss function, as the augmentation strategy likely helped the model develop more robust features for identifying cats despite their lower representation in the original dataset.

Table 1. Validation performance comparison for UNet models

Run	Pixel	IoU				Dice			
	Acc	BG	Cat	Dog	Mean	BG	Cat	Dog	Mean
Unaug	0.926	0.940	0.656	0.770	0.789	0.969	0.792	0.870	0.877
Aug	0.949	0.952	0.788	0.831	**0.857**	0.976	0.882	0.908	**0.922**
Aug+Weighted	0.945	0.948	0.778	0.819	0.848	0.973	0.875	0.900	0.916

4.2. Autoencoder

In all of our experiment results with autoencoder-based segmentation, we trained the autoencoder for 20 epochs, froze its parameters, and used its features to train the segmentation decoder for another 20 epochs. For both training steps, `Adam` optimiser is employed with a learning rate of 1e-3. We experimented with longer training for 50 epochs and a smaller learning rate of 1e-4 but results did not improve.

We observed that even when trained on a diverse set of augmented images as described in §2.3, the segmentation model trained with autoencoder features showed a great disparity in performance between cat and dog images, indicated by the class-wise IoU and DICE scores. To further mitigate this imbalance, we explored two approaches: (1) Biased augmentation: augment 4 examples for each cat image and 2 for each dog image, and (2) Weighted loss: apply a weight of 2.1 to the Cat class in the loss function of the segmentation decoder.

The results in Table 2 indicate that the autoencoder-based segmentation model is highly sensitive to the class imbalance. Both the default augmentation and biased augmentation strategies alone failed to handle the class imbalance. Using a class-weighted loss function, we were able to mitigate the performance gap between the cat and dog classes but not completely. Overall, the autoencoder-based segmentation model with the default augmentation described in §2.3 combined with class-weighted loss led to the best results.

Table 2. Validation performance comparison for Autoencoder models (Aug: default augmentation, Biased Aug: augment more cats than dogs, Weighted: apply weight to Cat class).

Run	Pixel	IoU				Dice			
	Acc	BG	Cat	Dog	Mean	BG	Cat	Dog	Mean
Aug	0.832	0.865	0.328	0.550	0.581	0.928	0.494	0.709	0.711
Biased Aug	0.820	0.851	0.349	0.527	0.576	0.920	0.517	0.690	0.709
Weighted	0.780	0.817	0.306	0.433	0.519	0.900	0.469	0.604	0.658
Aug+Weighted	0.827	0.861	0.393	0.517	**0.590**	0.925	0.564	0.682	**0.724**

4.3. CLIP

For CLIP feature-based segmentation, we trained only the decoder network while keeping all CLIP parameters frozen, following the transfer learning approach recommended by (Ranftl et al., 2021). This strategy preserves CLIP's pre-trained knowledge, reduces training time, and prevents overfitting on our relatively small dataset. We used the `Adam` optimiser with an initial learning rate of 1e-4, a weighted cross-entropy loss with cat class weight of 2.1, and batch size of 16. The model was trained for 20 epochs with a `ReduceLROnPlateau` scheduler. Despite using the same initial learning rate as our UNet implementation, we observed faster convergence with the CLIP-based model, suggesting the pre-trained features provide a more stable learning signal, a phenomenon also noted by (Mensink et al., 2021).

Table 3 shows the performance of our CLIP-based segmentation approaches on the validation set using the simple prompt and the multiple prompt approaches. The multiple prompt strategy showed a slight improvement over the simple prompt approach when trained for 20 epochs, particularly evident in the IoU and DICE scores. However, extending training to 50 epochs resulted in slightly diminished performance. This confirms our experimental observation that performance typical plateaued after 15 epochs with the generalisation gap widening beyond this point as the model began to overfit to the training data. These results demonstrate that despite CLIP not being originally designed for dense prediction tasks, its pre-trained visual-textual representations can be effectively adapted for semantic segmentation through our decoder architecture. The performance stability across different prompt strategies further suggests that CLIP's robust feature representations provide a solid foundation for transfer learning in segmentation applications.

Table 3. Validation performance comparison for CLIP models

Prompts	Pixel Acc	IoU BG	IoU Cat	IoU Dog	IoU Mean	Dice BG	Dice Cat	Dice Dog	Dice Mean
Simple	0.850	0.800	0.665	0.653	0.699	0.889	0.799	0.790	0.821
Multi 20 epoch	0.850	0.798	0.665	0.653	**0.701**	0.888	0.799	0.790	**0.826**
Multi 50 epoch	0.849	0.797	0.662	0.651	0.697	0.887	0.797	0.789	0.820

4.4. Best Models

Having selected the best UNet, autoencoder, and CLIP models based on the validation set, we report their performance on the held-out test set in Table 4. The autoencoder-based segmentation model fails to handle the class imbalance and performs worst overall, likely because the latent representation in the autoencoder has insufficient dimensionality to capture complex patterns in the image.

Using CLIP features to train the segmentation decoder improved the mean IoU and DICE scores by over 10% compared to the autoencoder's features. Moreover, the performance gap between cat and dog segmentation was significantly reduced. This result demonstrates the benefits of using features from a pretrained model over a model trained from scratch. However, CLIP was not able to outperform UNet as CLIP remains optimised for classification rather than dense prediction tasks.

UNet performed best overall on the test set, significantly outperforming CLIP and autoencoder. UNet's strong performance can be attributed to its architectural advantages: skip connections effectively preserve spatial information throughout the encoding-decoding process, allowing for precise boundary delineation, while multi-scale feature extraction enables the model to capture both low-level details and high-level semantic information. The model also shows balanced performance across all classes. UNet's segmentation specific architecture with its symmetrical encoder-decoder structure and preservation of spatial information at multiple resolutions enables it to achieve significantly better results. Hence, we used UNet as the base architecture to construct the prompt-based segmentation models in §5.

Table 4. Test performance comparison of the best of each segmentation model. BG represents background class.

Model	Pixel Acc	IoU BG	IoU Cat	IoU Dog	IoU Mean	Dice BG	Dice Cat	Dice Dog	Dice Mean
UNet	0.949	0.954	0.771	0.838	**0.854**	0.976	0.870	0.912	**0.920**
AutoEnc	0.838	0.870	0.341	0.562	0.591	0.931	0.509	0.720	0.720
CLIP	0.856	0.798	0.665	0.653	0.705	0.888	0.799	0.790	0.826

5. Prompt-based Models

Inspired by prompting techniques in LLMs, (Kirillov et al., 2023) proposed promptable segmentation models which enable real-time capabilities. The prompt can be a point, a bounding box, or free-form text which provides additional guidance to the model for the segmentation task. Given that our best model from §4 is UNet, we extended the model architecture to incorporate point prompts and free-form text prompts as auxiliary inputs. For both prompt-based approaches, we trained the model for 20 epochs and saved the checkpoint with the lowest validation loss.

5.1. Point Prompt

The point-based model allows the user to select any points on the image as additional input for the segmentation model. In our implementation, we used foreground and background points to help the model distinguish between cat/dog pixels and background pixels. First, we randomly sampled one or more foreground (cat/dog) points and background points, excluding points on the white outline. These points are used to create normalised Gaussian foreground and background heatmaps with the same dimensions as the image. As illustrated in Figure 2, the foreground and background heatmaps are concatenated and processed by a point encoder, starting from a 224×224×2 combined heatmap and resulting in a 14x14x128 heatmap encoding, which is then concatenated with the image features at the bottleneck layer.

During training, we sampled one foreground point and one background point for each image. For evaluation, we compared using 1, 5, and 10 foreground and background points for each image in the test set. As shown in Table 5, using one foreground and background point worsened segmentation performance, as a single point prompt introduces *ambiguity* to the model (Kirillov et al., 2023). However, using 5 and 10 foreground and background points outperformed the best UNet model, which indicate that point prompts are useful for segmentation when sufficient points are provided.

Table 5. Test performance comparison for point-prompted models

Points	Pixel Acc	IoU BG	IoU Cat	IoU Dog	IoU Mean	Dice BG	Dice Cat	Dice Dog	Dice Mean
1	0.945	0.953	0.736	0.829	0.839	0.976	0.848	0.907	0.910
5	0.955	0.967	0.762	0.850	0.859	0.983	0.865	0.919	0.922
10	0.960	0.975	0.775	0.859	**0.870**	0.987	0.873	0.924	**0.928**

Figure 6 illustrates how the model more accurately distinguishes between the dog and the background when provided with 5 foreground and background points compared to when given only one point of each type. Aside from the number of points, their locations should be considered as point prompts in ambiguous regions are more challenging for the model.

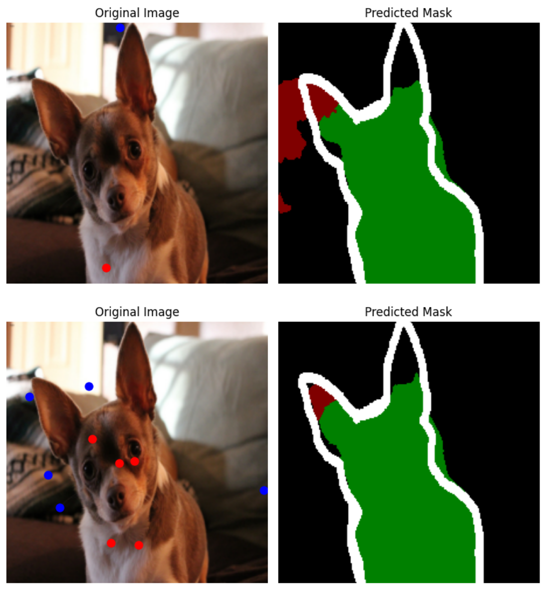
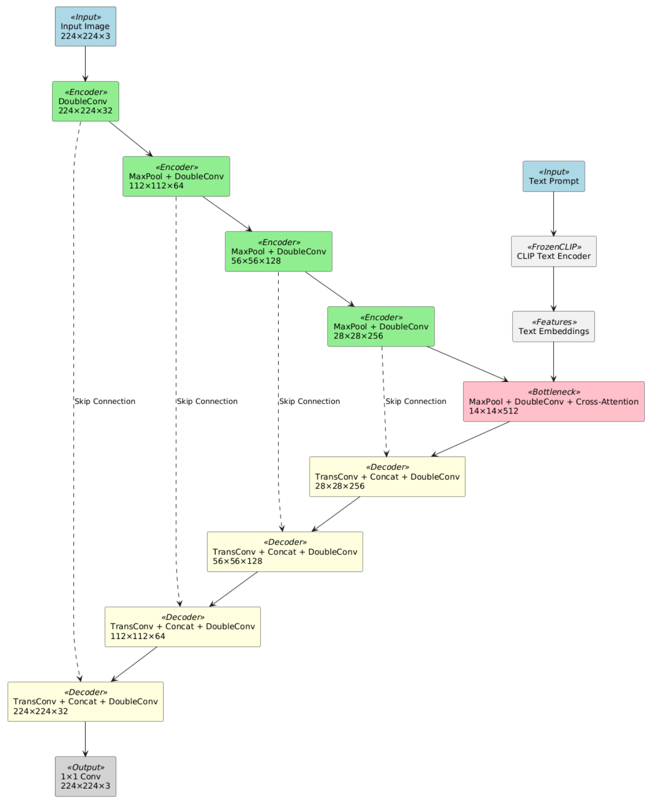


Figure 6. Impact of using 1 versus 5 foreground (red) and background (blue) points in the point-based segmentation model

5.2. Text Prompt

In addition to the point-based segmentation model, we explored incorporating a free-form text prompt into our best UNet model. At inference time, the user can provide a text to guide the segmentation task. To add the text prompt to the model, we closely followed the approach in (Kirillov et al., 2023). Text prompts are converted into embeddings using the text encoder from CLIP, and cross-attention is applied between the image features and the text embeddings at the bottleneck layer as shown in Figure 7.



«Input» Input Image 224×224×3
«Encoder» DoubleConv 224×224×32
«Encoder» MaxPool + DoubleConv 112×112×64
«Encoder» MaxPool + DoubleConv 56×56×128
«Encoder» MaxPool + DoubleConv 28×28×256
«Input» Text Prompt
«FrozenCLIP» CLIP Text Encoder
«Features» Text Embeddings
«Bottleneck» MaxPool + DoubleConv + Cross-Attention 14×14×512
Skip Connection
Skip Connection
Skip Connection
Skip Connection
«Decoder» TransConv + Concat + DoubleConv 28×28×256
«Decoder» TransConv + Concat + DoubleConv 56×56×128
«Decoder» TransConv + Concat + DoubleConv 112×112×64
«Decoder» TransConv + Concat + DoubleConv 224×224×32
«Output» 1×1 Conv 224×224×3

Figure 7. UNet with CLIP Text Encoder Architecture Diagram

During training, we determined the main class of each image based on pixel count per class, then randomly sampled from a list of short text prompts for that class as follows:

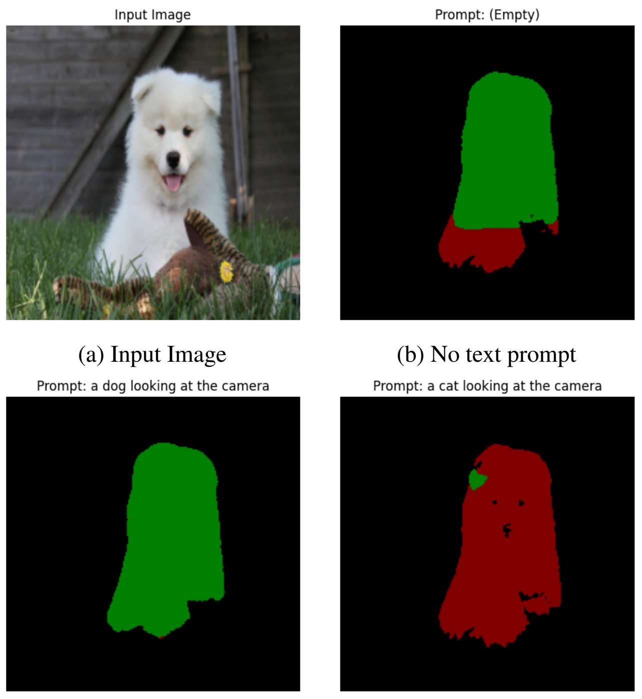
- Background prompts: ["the background", "background region with no pets", "the area surrounding", "scenery, furniture, floors, or walls", "The backdrop of the image."]
- Cat prompts: ["a cat", "a domestic cat with fur and whiskers", "a feline pet with pointed ears and a tail", "the complete shape and outline of a cat", "A cat staring at the camera.", "the quiet cat"]
- Dog prompts: ["a dog", "a domestic dog with fur and a snout", "a canine pet with distinctive ears", "the complete shape and outline of a dog", "A dog sitting.", "the furry dog"]

At test time, we evaluated the model using a different set of text prompts to ensure that it can generalise to unseen prompts and images. We compared the text-prompted model trained with and without class weighting in the loss function and recorded the results in Table 6. The results revealed that as with the base UNet model, the text-prompted model is able to handle the class imbalance without the need for class weighting. Furthermore, the best text-prompted model outperformed the best UNet model by 1.7% in mean IoU and 1.0% in mean Dice, indicating that providing a relevant prompt is helpful for the segmentation model.

Table 6. Test performance comparison for text-prompted models

Model	Pixel	IoU				Dice			
	Acc	BG	Cat	Dog	Mean	BG	Cat	Dog	Mean
Base	0.954	0.955	0.800	0.858	**0.871**	0.977	0.889	0.924	**0.930**
Weighted	0.933	0.941	0.717	0.788	0.816	0.970	0.835	0.882	0.896

We performed more extensive testing with different text prompts using a dog image as input and illustrated our findings in Figure 8. When no text prompt is given (8b), the model generally predicts the dog pixels (green) in the image correctly but with some incorrect cat pixels (red). Providing a consistent text prompt (8c) improves the segmentation mask as the model no longer predicts the incorrect cat pixels. However, a misleading prompt (8d) greatly confuses the model and causes it to incorrectly predict mostly cat pixels. This aligns with the recent literature which states that when the image and text inputs are inconsistent, vision-language models tend to strongly prefer the textual input even if it is factually incorrect (Deng et al., 2025).



Input Image
Prompt: (Empty)
Prompt: a dog looking at the camera
Prompt: a cat looking at the camera

(a) Input Image (b) No text prompt

(c) Consistent text prompt (d) Inconsistent text prompt

Figure 8. Impact of varying input texts in the text-prompted model

6. Robustness Testing

As shown in Figure 9, we systematically evaluated our best-performing model, UNet, against eight distinct image perturbations at varying intensity levels. This allows us to assess the model's resilience to different types of image corruption that might occur in real-world scenarios. Figure 12 provides a ranking of perturbation impacts on segmentation performance (blue bars), and Figure 10 shows the results for each. Salt & pepper noise emerged as the most destructive perturbation, causing a 44.3% decrease in Dice score. This is followed closely by contrast decrease with a 38.6% reduction. The severe impact of salt & pepper noise can be attributed to how it introduces sharp, isolated artifacts that disrupt true object boundaries and create false edges. These random high-contrast pixels particularly challenge the

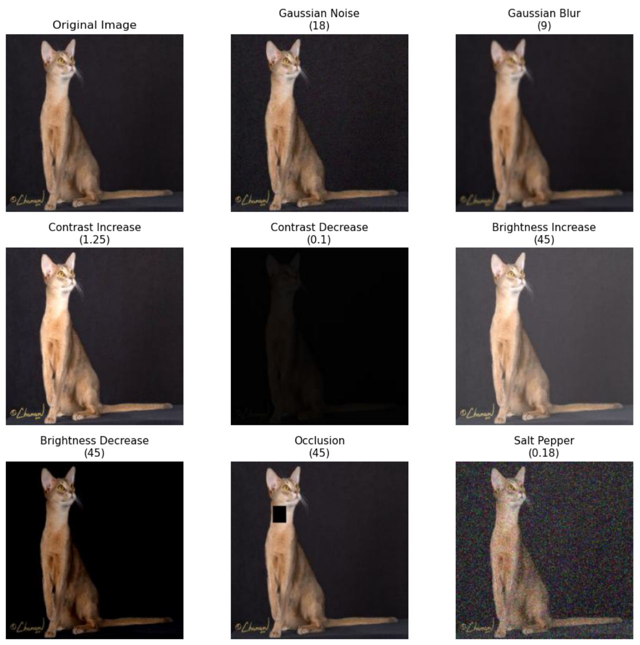


Figure 9. Image perturbation types at maximum intensity (level 9) compared to the original image

model's ability to maintain coherent segmentation boundaries. Other perturbations had significantly less impact on model performance. Gaussian blur, brightness decrease, and Gaussian noise showed moderate effects (4.0%, 3.6%, and 2.5% decreases respectively). Occlusion and contrast increase had minimal impact (0.5% and 0.1% decreases), while brightness increase actually improved performance slightly by 0.11%.

Figure 11 demonstrates the progressive impact of salt and pepper noise on segmentation quality. At low noise levels (Level 0), the model maintains relatively accurate segmentation with the cat's silhouette largely preserved, though with some misclassification in the lower body. As intensity increases to Level 3 and Level 6, we observed increased misclassification where cat pixels (red) are incorrectly classified as dog pixels (green). The model struggles to maintain the coherent shape of the cat, with the lower body and tail regions most severely affected by class confusion. At the maximum intensity (Level 9), the lower half of the cat is almost completely unrecognisable to the model and is misclassified as background, with only portions of the upper body still correctly identified. This dramatic degradation occurs because salt and pepper noise introduces random high-contrast pixels that disrupt the texture and edge patterns the model relies on for class discrimination. Unlike Gaussian noise, which affects images more uniformly, salt and pepper noise creates sharp artifacts that interfere with the distinctive visual features that separate cat and dog classes, explaining why this perturbation type proved most challenging for our segmentation model.

Decreasing contrast was the second most impactful perturbation. This suggests that the model heavily relies on contrast differences to distinguish object boundaries. Conversely, contrast increase had almost no effect (0.1% decrease), indicating the model maintains performance when contrast is enhanced. Brightness modifications showed asymmetric effects: brightness decrease caused moderate performance degradation (3.6% decrease), while brightness increase actually improved performance slightly. This asymmetry suggests the model

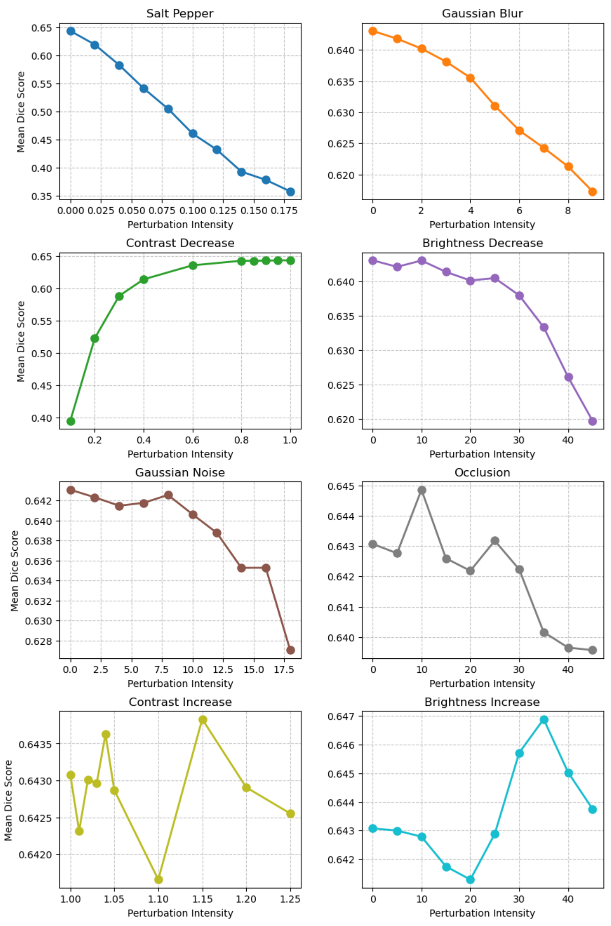


Figure 10. Impact of various image perturbations on segmentation performance. Each plot shows the Mean Dice Score as a function of perturbation intensity

may have been optimised slightly toward brighter images during training. Occlusion testing revealed strong robustness with only a minimal performance decline of 0.5%, suggesting the model effectively infers missing parts based on surrounding context, a valuable property for real-world applications where objects are often partially hidden.

The clear vulnerabilities to salt & pepper noise and contrast decrease highlight areas requiring preprocessing attention in real-world applications. Implementing noise filtering or contrast normalisation would significantly improve robustness. The model's resilience to brightness variations, occlusion, Gaussian perturbations, and contrast increases demonstrates successful data augmentation during training.

To further investigate the impact of our data augmentation strategy, we compared the robustness of models trained with and without augmentation across all perturbation types. As shown in Figure 12, data augmentation significantly improved model resilience for a lot of the perturbation types, with the largest improvement being for contrast decrease. An exception was salt & pepper noise, where both models struggled considerably, with the augmented model showing higher sensitivity. Even though our model was not explicitly trained on any of the tested perturbations, the varied augmentations implemented (rotations, flips, elastic deformations, and colour jitters) improved resilience to most perturbations while unexpectedly increasing sensitivity to salt & pepper noise. This suggests that augmentation strategies create indirect robust-

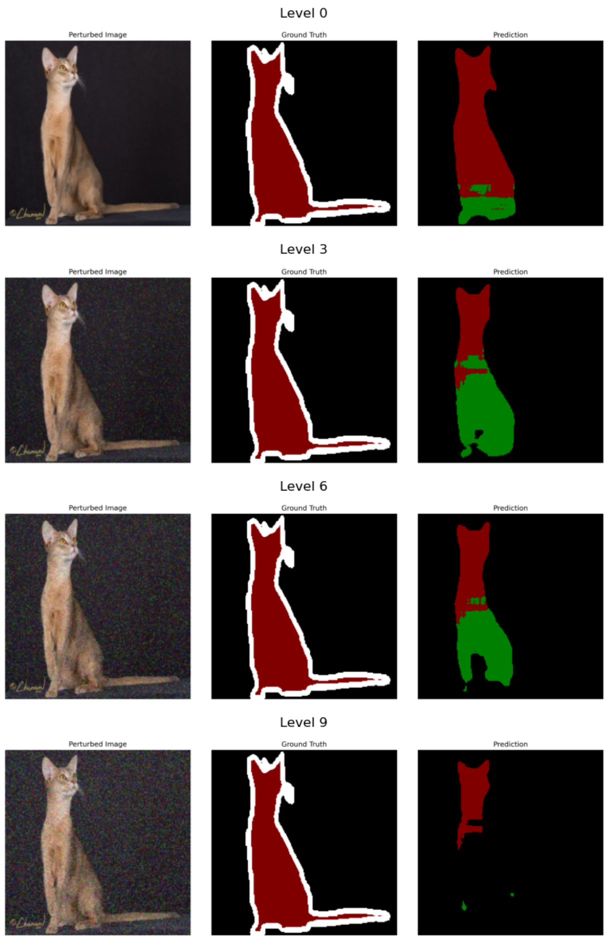


Figure 11. Impact of varying levels of salt and pepper noise perturbation on model predictions.

ness benefits through feature generalisation, but may also introduce specific vulnerabilities through optimisation trade-offs.

These results confirm that our augmentation strategy effectively enhanced model robustness against most real-world image corruptions, particularly for high-frequency noise patterns that are most disruptive to segmentation quality.

7. Conclusions

Our comprehensive exploration of image segmentation techniques on the Oxford-IIIT Pet Dataset has yielded several important findings. The UNet architecture consistently outperformed other approaches, achieving a mean IoU of 0.854 and a Dice score of 0.920 on the test set. This superior performance can be attributed to its skip connections that effectively preserve spatial information throughout the encoding-decoding process, allowing for precise boundary delineation and robust multi-scale feature extraction. While the CLIP-based approach demonstrated the potential of leveraging pre-trained visual-textual representations for segmentation, achieving a mean IoU of 0.705, it could not match UNet's performance due to CLIP's optimisation for classification rather than dense prediction tasks. The autoencoder-based segmentation model performed notably worse, with a mean IoU of 0.591 and high discrepancy in performance between Cat and Dog classes, suggesting that the latent representation lacked sufficient dimensionality to capture complex spatial patterns necessary for accurate segmentation.

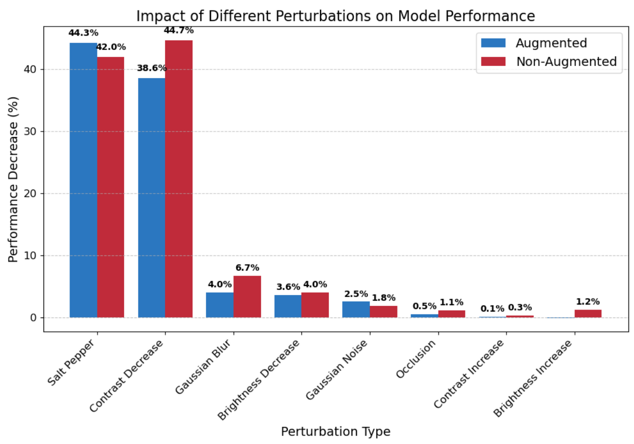


Figure 12. Impact of data augmentation on model robustness.

Extending the best UNet model to use additional foreground and background points or a free-form text prompt improves segmentation performance and allows for interactive use cases. The best point-based and text-prompted models achieved a mean IoU of 0.870 and 0.871, respectively. However, it is important to consider the number and locations of point prompts in the point-based model, as well as the consistency of the input text in the text-prompted model, as inconsistent prompts can introduce ambiguity and confusion to the model.

Our robustness analysis revealed that salt & pepper noise posed the greatest challenge to our best model, causing a 44.3% decrease in Dice score at maximum intensity, followed by contrast decrease with a 38.6% reduction. This indicates that the model heavily relies on edge patterns and contrast differences for segmentation. Conversely, the model demonstrated remarkable resilience to brightness variations, occlusion, and Gaussian perturbations, with minimal performance degradation. Data augmentation proved crucial for model performance and robustness, significantly improving resilience against most perturbation types, particularly for high-frequency noise patterns.

Future work could leverage salt & pepper and contrast decrease as additional data augmentation methods, since the model was most vulnerable to these perturbations. In addition, we could explore more complex augmentation methods which have proven to be effective for image segmentation such as Copy-Paste (Ghiasi et al., 2021) and Cutmix (Yun et al., 2019). To further improve autoencoder-based segmentation, future experiments could investigate adding skip connections to the autoencoder to increase the sharpness of the reconstructed images according to (Collin & De Vleeschouwer, 2021). Given sufficient computational resources, we could explore using larger variants of CLIP or greater feature dimensions in UNet and the autoencoder to have increased model capacity. Dropout could also be applied as additional regularisation to the networks, however we did not prioritise applying this technique as batch normalisation can remove the need for Dropout in certain cases (Ioffe & Szegedy, 2015). For prompt-based segmentation, evaluating other prompts such as scribbles and bounding boxes, along with combinations of multiple prompts, could further enable interactive capabilities and potentially enhance segmentation performance (Kirillov et al., 2023).

References

Amir, Shir, Gandelsman, Yossi, Bagon, Shai, and Dekel, Tali. Deep vit features as dense visual descriptors. *CoRR*, abs/2112.05814, 2021. URL https://arxiv.org/abs/2112.05814.

Badrinarayanan, Vijay, Kendall, Alex, and Cipolla, Roberto. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615.

Bagheri, Elham and Mohsenzadeh, Yalda. Modeling visual memorability assessment with autoencoders reveals characteristics of memorable images. *arXiv preprint arXiv:2410.15235*, 2024.

Collin, Anne-Sophie and De Vleeschouwer, Christophe. Improved anomaly detection by training an autoencoder with skip connections on images corrupted with stain-shaped noise. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 7915–7922. IEEE, 2021.

Csurka, Gabriela, Larlus, Diane, and Perronnin, Florent. What is a good evaluation measure for semantic segmentation? In *Proceedings of the British Machine Vision Conference*, pp. 32.1–32.11. BMVA Press, 2013. doi: 10.5244/C.27.32.

Deng, Ailin, Cao, Tri, Chen, Zhirui, and Hooi, Bryan. Words or vision: Do vision-language models have blind faith in text? *arXiv preprint arXiv:2503.02199*, 2025.

Ghiasi, Golnaz, Cui, Yin, Srinivas, Aravind, Qian, Rui, Lin, Tsung-Yi, Cubuk, Ekin D, Le, Quoc V, and Zoph, Barret. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2918–2928, 2021.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

Keskar, Nitish Shirish, Mudigere, Dheevatsa, Nocedal, Jorge, Smelyanskiy, Mikhail, and Tang, Ping Tak Peter. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. URL https://arxiv.org/abs/1609.04836.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Kirillov, Alexander, Mintun, Eric, Ravi, Nikhila, Mao, Hanzi, Rolland, Chloe, Gustafson, Laura, Xiao, Tete, Whitehead, Spencer, Berg, Alexander C, Lo, Wan-Yen, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Li, Boyi, Weinberger, Kilian Q., Belongie, Serge, Koltun, Vladlen, and Ranftl, René. Language-driven semantic segmentation, 2022. URL https://arxiv.org/abs/2201.03546.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation, 2015. URL https://arxiv.org/abs/1411.4038.

Mensink, Thomas, Uijlings, Jasper, Kuznetsova, Alina, Gygli, Michael, and Ferrari, Vittorio. Factors of influence for transfer learning across diverse appearance domains and task types, 2021. URL https://arxiv.org/abs/2103.13318.

Mumuni, Alhassan and Mumuni, Fuseini. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16, 2022. doi: 10.1016/j.array.2022.100258.

Noh, Hyeonwoo, Hong, Seunghoon, and Han, Bohyung. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1520–1528, 2015. doi: 10.1109/ICCV.2015.178.

Parkhi, Omkar M., Vedaldi, Andrea, Zisserman, Andrew, and Jawahar, C. V. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3498–3505, 2012. doi: 10.1109/CVPR.2012.6248092.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, Desmaison, Alban, Köpf, Andreas, Yang, Edward, DeVito, Zach, Raison, Martin, Tejani, Alykhan, Chilamkurthy, Sasank, Steiner, Benoit, Fang, Lu, Bai, Junjie, and Chintala, Soumith. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Radford, Alec, Kim, Jong Wook, Hallacy, Chris, Ramesh, Aditya, Goh, Gabriel, Agarwal, Sandhini, Sastry, Girish, Askell, Amanda, Mishkin, Pamela, Clark, Jack, Krueger, Gretchen, and Sutskever, Ilya. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Ranftl, René, Bochkovskiy, Alexey, and Koltun, Vladlen. Vision transformers for dense prediction, 2021. URL https://arxiv.org/abs/2103.13413.

Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Smith, Leslie N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017. doi: 10.1109/WACV.2017.58.

Sudre, Carole H, Li, Wenqi, Vercauteren, Tom, Ourselin, Sebastien, and Cardoso, M Jorge. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, volume 10553 of *Lecture Notes in Computer Science*, pp. 240–248. Springer, 2017. doi: 10.1007/978-3-319-67558-9_28.

Tishby, Naftali and Zaslavsky, Noga. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2015. doi: 10.1109/ITW.2015.7133169.

Xu, Mengde, Zhang, Zheng, Wei, Fangyun, Lin, Yutong, Cao, Yue, Hu, Han, and Bai, Xiang. A simple baseline for open-vocabulary semantic segmentation with pre-trained vision-language model, 2022. URL https://arxiv.org/abs/2112.14757.

Yun, Sangdoo, Han, Dongyoon, Oh, Seong Joon, Chun, Sanghyuk, Choe, Junsuk, and Yoo, Youngjoon. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Zeiler, Matthew D., Krishnan, Dilip, Taylor, Graham W., and Fergus, Rob. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2528–2535, 2010. doi: 10.1109/CVPR.2010.5539957.

Zhou, Kaiyang, Yang, Jingkang, Loy, Chen Change, and Liu, Ziwei. Conditional prompt learning for vision-language models, 2022. URL https://arxiv.org/abs/2203.05557.

Zhou, Zongwei, Siddiquee, Md Mahfuzur Rahman, Tajbakhsh, Nima, and Liang, Jianming. Unet++: A nested u-net architecture for medical image segmentation, 2018. URL https://arxiv.org/abs/1807.10165.

A. Statement of Contribution

Table 7. Task Distribution

Task	s1707068	s2737499
Data augmentation	x	x
UNet	x	
Autoencoder		x
CLIP-based	x	
Prompt-based		x
Text-prompt		x
Evaluation	x	x
Robustness analysis	x	