CPSC 542 - Assignment 2

Spencer Au

Problem Statement

The goal of this project is to use a U-Net Model in order to perform image segmentation on a dataset of various images of food. The dataset contains images of food, and the goal is to be able to segment the images of food into the pixels that are the food in question or the pixels that are just the "background". The relevance of this problem is that it can be used to help automate the process of identifying and classifying food in images, which can be useful for a variety of applications such as food recognition, dietary analysis, and food waste reduction. The task of segmentating (food) images involves understanding complex patterns, which traditional machine learning models may struggle to handle efficiently due to their limited capacity for feature extraction, etc. Deep learning models, particularly convolutional neural networks (CNNs) and specifically the U-Net architecture, excel in image segmentation tasks across many datasets and benchmarks. This capability to learn directly from data and improve with scale makes deep learning the de facto standard for image segmentation problems. The plan to solve the problem is to first preprocess the data, and then train a U-Net model on the dataset. The U-Net model is a popular architecture for image segmentation tasks, and is known for its ability to effectively segment images of varying sizes and shapes. The U-Net model consists of a contracting path, which captures context, and an expansive path, which enables precise localization. The model is trained using a combination of the binary crossentropy loss function and the Adam optimizer. The model is trained on the Chapman University DGX Cluster, which is a high performance computing cluster with 8 NVIDIA A100 GPUs. The model is trained for 100 epochs, and is evaluated on a test set to determine its performance. The model is then used to segment images of food into different categories, and the results are analyzed to determine the model's effectiveness.

Data

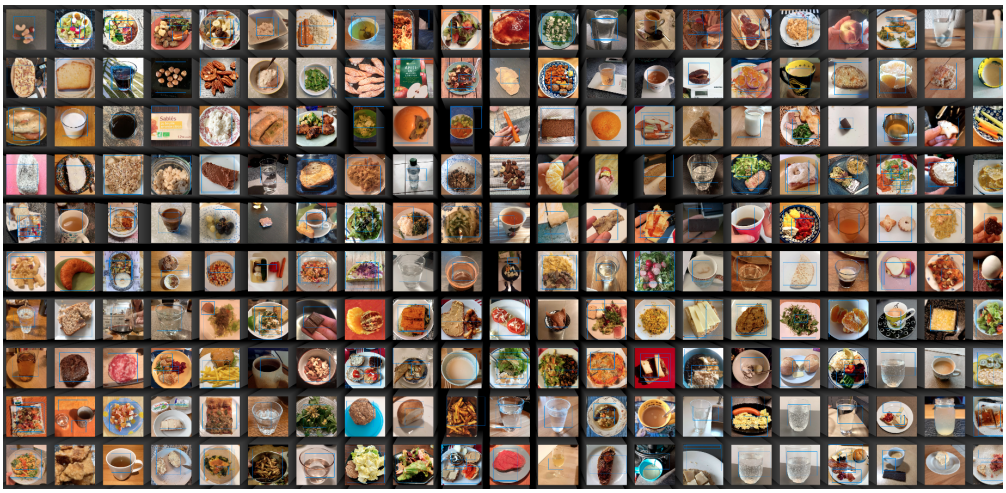


Figure 1: Visualized Data

The data used in this project was obtained from the Food Recognition 2022 Dataset from Kaggle. The dataset contains 39,962 RGB images and 76,491 annotations in teh training class, split into 498 classes. The validation set consists of 1000 RGB images, with 1830 annotations and 498 classes. The test set is the Food Recognition Benchmark. The dataset is formatted in MS-COCO format. Since we are only performing binary segmentation, namely seeing which pixels belong to the object being detected and which are background pixels, we only need the images and the annotations. The images are in the form of .jpg files, and the annotations are in the form of .json files. The images are of varying sizes, and the annotations are in the form of polygons. The dataset is quite large, and contains a wide variety of food images, which will help the model generalize to new images of food. The dataset is also quite diverse, containing images of food from different cultures, with different colors, and different backgrounds, which will help the model learn to classify images of food that may be occluded, be of different colors, have different backgrounds, or lighting conditions that cannot be effectively controlled for via image augmentation. The dataset is also quite balanced, with a large number of images for each class, which will help the model learn to classify images of food that may be occluded, be of different colors, have different backgrounds, or lighting conditions that cannot be effectively controlled for via image augmentation. The dataset is also quite balanced, with a large number of images for each class, which will help the model learn to classify images of food that may be occluded, be of different colors, have different backgrounds, or lighting conditions that cannot be effectively controlled for via image augmentation.

Methods

Preprocessing

In terms of preprocessing, I used a custom HubSegmentationDataGenerator class which extends the Keras Sequence class in order to load the data, and perform image augmentation. The class is a subclass of the Keras Sequence class, and is used to load the images and annotations, and perform image augmentation. The class takes in the image and annotation datasets, the batch size, image dimensions, and a shuffle flag to determine if the data should be shuffled at the end of each epoch. The **len** method computes the number of batches per epoch, while **getitem** fetches a specific batch, applying necessary preprocessing such as resizing and datatype conversion. Images are resized to the target dimensions and preprocessed for neural network compatibility, and masks are converted from boolean to uint8, resized to match the images, and ensured to be in a single-channel format. If any mask has more than one channel, only the first channel is used, and any extra dimensions are removed to maintain a consistent input shape for the model. This generator thereby streamlines the process of preparing image-mask pairs for efficient training of segmentation models. **on_epoch_end** is automatically called at the end of each epoch during the model's training. Its primary function is to update the indices of the dataset. **data_generation** generates a batch of data by processing the corresponding images and masks for training. For each index in the current batch, it loads the image from the Hub dataset, resizes it to the desired dimensions, and applies any necessary preprocessing steps such as normalizing pixel values. Concurrently, it processes the associated masks by resizing them and converting their data type to ensure consistency, and, if needed, adds a channel dimension to match the model's expected input shape. The processed images and masks are then compiled into arrays which are subsequently returned to the model for training on that particular batch.

Model

In terms of the model, I used a U-Net model style architecture, which is a popular architecture for image segmentation tasks. The U-Net model consists of a encoding path and a decoding path, which are connected by skip connections. The encoding and decoding path are also connected by a convolutional block as a bridge between the two paths. Each encoder block consists of a convolutional block followed by a 2x2 max pooling layer. Each decoder block consists of an transposed convolutional layer with a 3x3 kernel and a stride of 2 for upsampling followed by a concatenation with the corresponding encoder block to facilitate skip connections, and then 2 subsequent convolutional layers using a 3x3 kernel, ReLu activation, and same padding. The convolutional blocks consist of a convolutional layer with a 3x3 kernel, ReLu activation, same padding, and l2 regularization followed by a Dropout layer with a dropout rate of 0.4, and then follwoed by another convolutional layer with a 3x3 kernel, ReLu activation, same padding, and l2 regularization.

The encoder part of the U-Net architecture consists of 4 encoder blocks, with 64, 128, 256, and 512 filters respectively. The bridge consists of a convolutional block with 1024 filters. The decoder part of the U-Net architecture consists of 4 decoder blocks, with 512, 256, 128, and 64 filters respectively. The output layer consists of a convolutional layer with 1 filter and a sigmoid activation function. The model is trained using the binary crossentropy loss function and the Adam optimizer with the default starting learning rate. The model is trained for 100 epochs utilizing early stopping to prevent overfitting wiht a patience of 10 epochs. The model was evauluated using accuracy and the average IoU as the metrics, and the results were analyzed to determine the model's effectiveness.

Results and Discussion

Metrics

Metric	Train	Val
Loss	0.4071	0.4026
Accuracy	0.8007	0.8062
Mean IoU	0.3717	0.3696

Looking at the data and metrics, both the training and validation set performed reasonably well, as the loss for both is relatively low at 0.4071 and 0.4026 for the training and validation sets respectively. In addition, the accuracy metric is also prety good on both the training and validation sets at 0.8007 and 0.8062 respectively. The IoU, or Intersection over Union, is not performing as well at 0.3717 and 0.3696 respectively. An IoU of 0.3717 means that, on average, the overlap area between the predicted masks and the actual masks is 37.17% of their union area for that evaluation set. Similarly, an IoU of 0.3696 indicates a 36.96% overlap in that evaluation set. These scores suggest that the model is performing reasonably well, but could be improved. The model is able to effectively segment images of food into the pixels that are the food in question or the pixels that are just the "background". The model is able to generalize to new images of food as there doesn't seem to be any indication of overfitting.

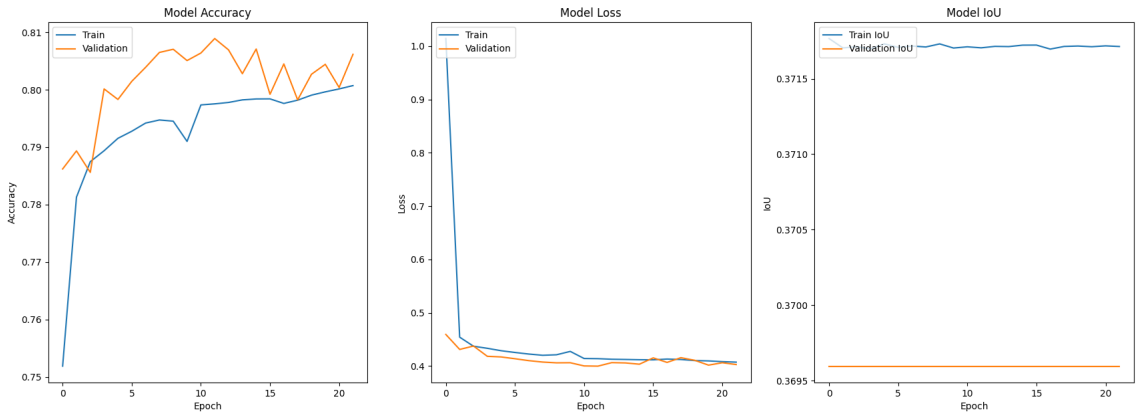


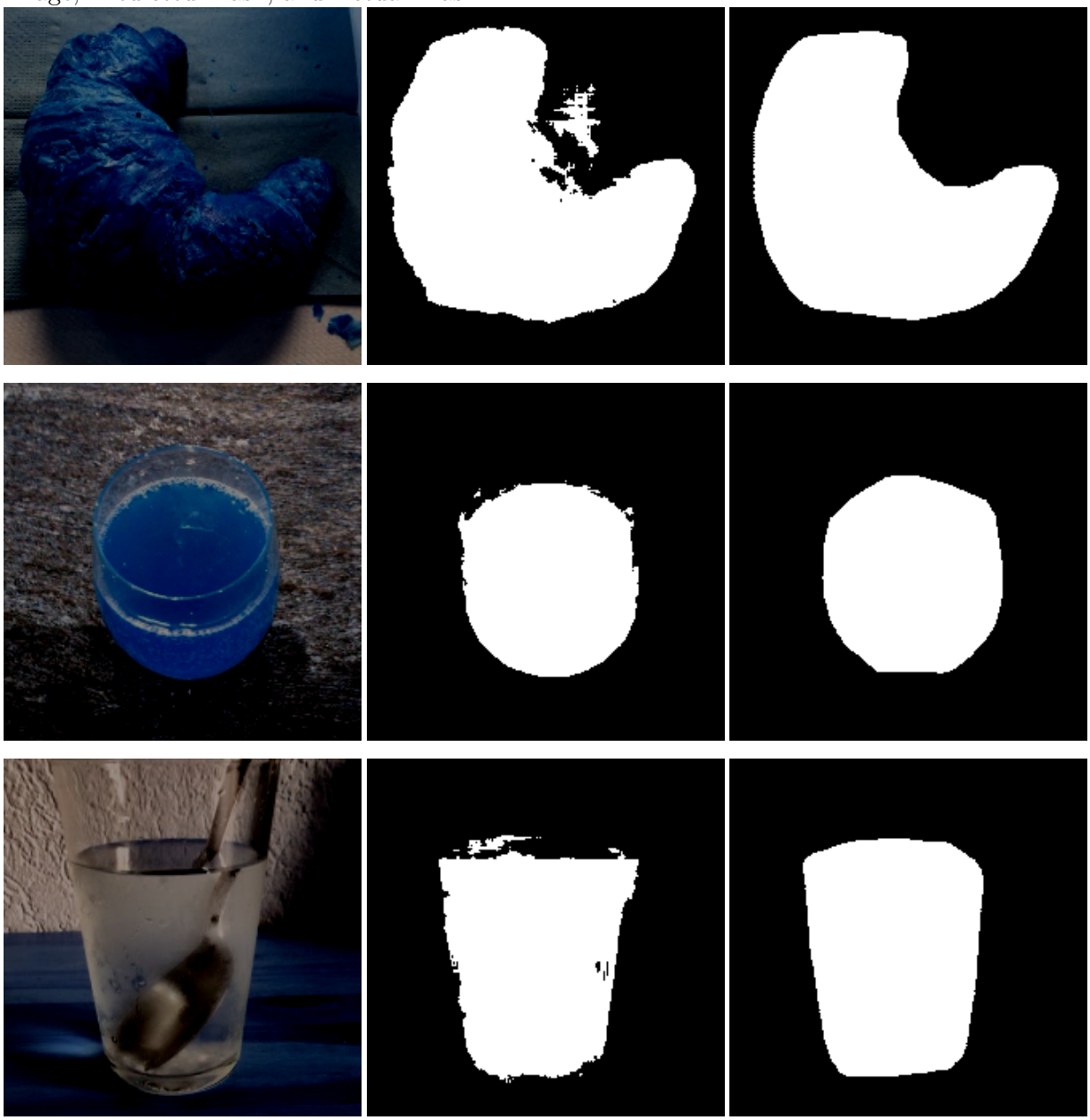
Figure 2: Metrics Over Time

The first graph indicates that the accuracy of the model on both the training and validation datasets improve with each epoch, but the validation accuracy does seem to fluctuate but generally increases, indicating learning. The second graph represents Model Loss, showing a downward trend in both training and validation loss, which is a good sign that the model is improving and learning to minimize the error over time. Finally, the third graph shows the IoU (Intersection over Union), which is a measure of the accuracy of object detection. In this graph, the IoU score remains relatively flat, suggesting that there might not be significant changes in how well the predicted masks overlap with the actual masks throughout training or is being influenced by an aspect of the data or model that is not changing across epochs.

3 Best 3 Worst Predictions

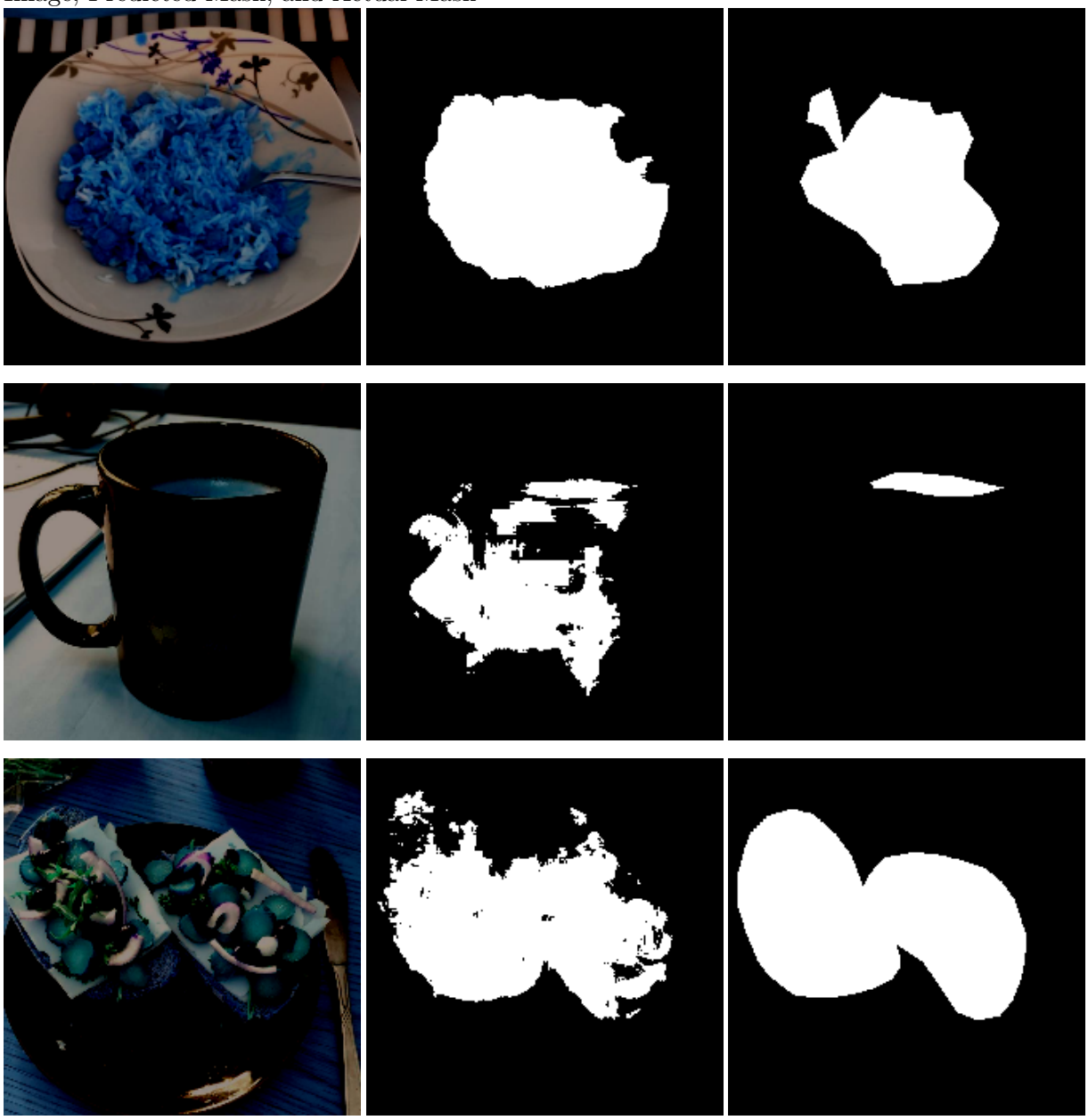
3 Best Predictions

Image, Predicted Mask, and Actual Mask



3 Worst Predictions

Image, Predicted Mask, and Actual Mask



Grad-CAM Heatmaps

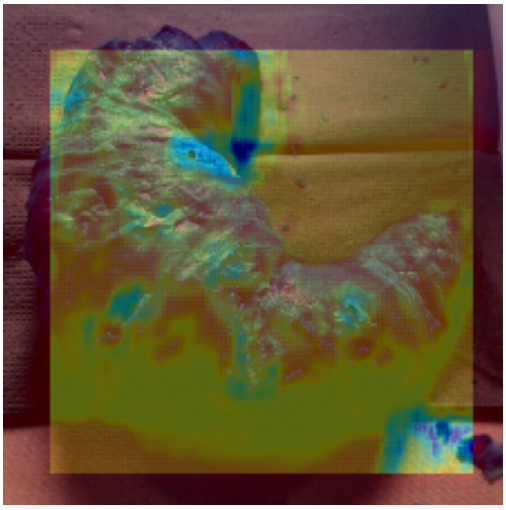


Figure 3: Heatmap of a Croissant

The model seems to put high importance on the "ridges" of the coirssant, which is a good indication that the model is learning to identify the food in the image.

Figure 4: Heatmap of a Cup of Coffee

The model seems to put high importance on the majority of the coffee cup, and some importance (yellow) on the sides of the cup. The actual coffee however is not as important to the model.

Discussion and Conclusion

This U-Net model was trained for the task of binary image segmentation, aiming to identify food items within images. Throughout the training process, various parameters were adjusted, such as the learning rate and utilizing different layers, to optimize performance. Metrics such as loss, accuracy, and Intersection over Union (IoU) were utilized to gauge the model's efficacy.The model demonstrated decent performance on the validation set, but the IoU metric indicated room for improvement in accurately capturing all relevant pixels belonging to food items. Given the complexity of the images and variability of the subjects (different food items), further tuning, potentially with deeper architectures or more accurate masks, might yield better results. In terms of deploying this model to production, careful consideration is necessary. While the model shows promise, the current level of performance is not sufficient for critical applications where high precision is mandatory. For non-critical applications that can tolerate some margin of error, such as initial filtering in a food social media app such as Yelp, this model might be adequate.In conclusion, this assignment was a good exercise in understand how to approach and implement image segmentation.