Final Report

Jinal Vyas
Arizona State University
Tempe, AZ, USA
jjvyas1@asu.edu

1. List of Major Tasks Done

1. Pre-WarmUp: 90 Experiments for MedMNISTv2
2. WamrUp Exercises: 1-9
3. Snakes Method
4. Otsu's Method
5. Achieved Near Baseline Accuracy for Classification of ChestXray14 using ConvNextV2
6. Achieved Near Baseline Accuracy for Segmentation of ChestXDet using UperNet with Swin Backbone
7. Designed and Implemented a combined model for Performing Classification + Segmentation using a single architecture

2. Task - 1: Pre-Warmup: MedMNSITv2

Here, MedMNIST [8] has been used for obtaining the datasets for 18 diseases, 12 of which are classified under MedMNIST2D and other 6 are classified under MedMNIST3D. The github implementation for ResNet50 [1] can be found here [5].

2.1. Performance Metrics for MedMNIST-2D

Here, for both accuracy and auc, the test accuracy is reported. Validation and Training accuracy can be found in the logs whose link has been given at the end in the report.

	ChestMNIST	BloodMNIST	BreastMNIST	DermaMNIST	OctMNIST	OrganAMNIST
Run-1	0.94771	0.96229	0.45513	0.70075	0.80500	0.92654
Run-2	0.94797	0.95645	0.80769	0.72865	0.76700	0.91681
Run-3	0.94767	0.95265	0.67308	0.71970	0.77500	0.92682
Run-4	0.94794	0.95557	0.67949	0.74963	0.77500	0.93008
Run-5	0.94739	0.95645	0.81410	0.73367	0.76700	0.92418
Mean	0.94774	0.95668	0.68590	0.72648	0.77780	0.92489
Standard Deviation	0.00021	0.00313	0.13081	0.001613	0.01406	0.00445

	OrganCMNIST	OrganSMNIST	PathMNIST	PneumoniaMNIST	RetinaMNIST	TissueMNIST
Run-1	0.90628	0.78940	0.91421	0.84135	0.51000	0.70362
Run-2	0.90287	0.78339	0.84345	0.87019	0.51250	0.69860
Run-3	0.89959	0.77943	0.88217	0.85096	0.52500	0.69966
Run-4	0.90202	0.78283	0.90432	0.85897	0.52000	0.70341
Run-5	0.90567	0.77490	0.86769	0.82212	0.52570	0.70207
Mean	0.90348	0.78181	0.88257	0.84492	0.51484	0.699907
Standard Deviation	0.00276	0.00567	0.02616	0.01472	0.00665	0.00220

Table 1. Accuracy Metrics for MedMNIST-2D

	ChestMNIST	BloodMNIST	BreastMNIST	DermaMNIST	OctMNIST	OrganAMNIST
Run-1	0.77304	0.99788	0.76044	0.88641	0.96913	0.99684
Run-2	0.77255	0.99758	0.83939	0.91156	0.95845	0.99546
Run-3	0.77240	0.99692	0.85589	0.88760	0.95960	0.99503
Run-4	0.77857	0.99703	0.85067	0.91853	0.96215	0.99606
Run-5	0.78028	0.99724	0.85589	0.90858	0.96245	0.99531
Mean	0.77353	0.99733	0.83246	0.902536	0.96235	0.99574
Standard Deviation	0.00336	0.00035	0.03651	0.013091	0.00227	0.00065

	OrganCMNIST	OrganSMNIST	PathMNIST	PneumoniaMNIST	RetinaMNIST	TissueMNIST
Run-1	0.99222	0.97879	0.99158	0.95560	0.71513	0.93873
Run-2	0.99184	0.97504	0.98551	0.96472	0.71334	0.93755
Run-3	0.99201	0.99201	0.98603	0.95628	0.72290	0.93810
Run-4	0.99194	0.97640	0.99007	0.96397	0.72390	0.93838
Run-5	0.99235	0.97520	0.98505	0.95244	0.72472	0.93810
Mean	0.99215	0.98149	0.98745	0.95860	0.71980	0.93757
Standard Deviation	0.00028	0.00729	0.00271	0.00424	0.00439	0.00064

Table 2. AUC Metrics for MedMNIST-2D

2.2. Performance Metrics for MedMNIST-3D

	AdrenalMNIST	FractureMNIST	NoduleMNIST	OrganMNIST	SynapseMNIST	VesselMNIST
Run-1	0.84228	0.57083	0.84839	0.87839	0.73864	0.88743
Run-2	0.82550	0.56250	0.85161	0.88361	0.74716	0.47644
Run-3	0.66779	0.52917	0.85161	0.88033	0.73011	0.51571
Run-4	0.75168	0.63333	0.85161	0.87705	0.73864	0.41623
Run-5	0.67450	0.53333	0.84516	0.90492	0.75000	0.89791
Mean	0.75233	0.56582	0.86980	0.88466	0.74084	0.53874
Standard Deviation	0.0768	0.0373	0.0203	0.0133	0.0071	0.2336

Table 3. Accuracy Metrics for MedMNIST-3D

	AdrenalMNIST	FractureMNIST	NoduleMNIST	OrganMNIST	SynapseMNIST	VesselMNIST
Run-1	0.88577	0.73658	0.88910	0.99102	0.67110	0.76628
Run-2	0.88083	0.76245	0.86287	0.99185	0.70649	0.81622
Run-3	0.86222	0.72348	0.88955	0.99073	0.71624	0.78754
Run-4	0.83286	0.77021	0.86935	0.99291	0.70551	0.81889
Run-5	0.85102	0.72861	0.90898	0.99386	0.71763	0.83467
Mean	0.88254	0.744847	0.88385	0.99207	0.70399	0.78412
Standard Deviation	0.02182	0.02045	0.01725	0.00103	0.02131	0.02531

Table 4. AUC Metrics for MedMNIST-3D

2.3. Comparison with Reported Benchmarks

On looking at these values, almost all the models have scored similar scores except a few, which differ by larger margin. Multiple reasons are possible for that, one of the most prominent being a bad model seed, that's why it is a good practice to use same model seed for all the experiments. Also the stochastic processes induce randomness in the training.

Here is the link to the folder containing the log file for all the 90 experiments: Logs

Methods	PathMNIST		ChestMNIST		DermaMNIST		OCTMNIST		PneumoniaMNIST		RetinaMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)[10]	0.983	0.907	0.768	0.947	0.917	0.735	0.943	0.743	0.944	0.854	0.717	0.524
ResNet-18 (224)[10]	0.989	0.909	0.773	0.947	**0.920**	0.754	0.958	0.763	0.956	0.864	0.710	0.493
ResNet-50 (28)[10]	**0.990**	**0.911**	0.769	0.947	0.913	0.735	0.952	0.762	0.948	0.854	0.726	0.528
ResNet-50 (224)[10]	0.989	0.892	0.773	0.948	0.912	0.731	0.958	**0.776**	0.962	0.884	0.716	0.511
auto-sklearn[11]	0.934	0.716	0.649	0.779	0.902	0.719	0.887	0.601	0.942	0.855	0.690	0.515
AutoKeras[12]	0.959	0.834	0.742	0.937	0.915	0.749	0.955	0.763	0.947	0.878	0.719	0.503
Google AutoML Vision	0.944	0.728	**0.778**	**0.948**	0.914	**0.768**	**0.963**	0.771	**0.991**	**0.946**	**0.750**	**0.531**
Methods	**BreastMNIST**		**BloodMNIST**		**TissueMNIST**		**OrganAMNIST**		**OrganCMNIST**		**OrganSMNIST**	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)[10]	0.901	**0.863**	**0.998**	0.958	0.930	0.676	0.997	0.935	0.992	0.900	0.972	0.782
ResNet-18 (224)[10]	0.891	0.833	**0.998**	0.963	0.933	0.681	**0.998**	**0.951**	**0.994**	**0.920**	0.974	0.778
ResNet-50 (28)[10]	0.857	0.812	0.997	0.956	0.931	0.680	0.997	0.935	0.992	0.905	0.972	0.770
ResNet-50 (224)[10]	0.866	0.842	0.997	0.950	0.932	0.680	**0.998**	0.947	0.993	0.911	**0.975**	0.785
auto-sklearn[11]	0.836	0.803	0.984	0.878	0.828	0.532	0.963	0.762	0.976	0.829	0.945	0.672
AutoKeras[12]	0.871	0.831	0.998	0.961	**0.941**	**0.703**	0.994	0.905	0.990	0.879	0.974	**0.813**
Google AutoML Vision	**0.919**	0.861	**0.998**	**0.966**	0.924	0.673	0.990	0.886	0.988	0.877	0.964	0.749

Table 3. Benchmark on each dataset of MedMNIST2D in metrics of AUC and ACC.

Methods	OrganMNIST3D		NoduleMNIST3D		FractureMNIST3D		AdrenalMNIST3D		VesselMNIST3D		SynapseMNIST3D	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18[10] +2.5D	0.977	0.788	0.838	0.835	0.587	0.451	0.718	0.772	0.748	0.846	0.634	0.696
ResNet-18[10] +3D	**0.996**	**0.907**	0.863	0.844	0.712	0.508	0.827	0.721	0.874	0.877	0.820	0.745
ResNet-18[10] +ACS[30]	0.994	0.900	0.873	0.847	0.714	0.497	**0.839**	0.754	**0.930**	**0.928**	0.705	0.722
ResNet-50[10] +2.5D	0.974	0.769	0.835	0.848	0.552	0.397	0.732	0.763	0.751	0.877	0.669	0.735
ResNet-50[10] +3D	0.994	0.883	0.875	0.847	0.725	0.494	0.828	0.745	0.907	0.918	**0.851**	**0.795**
ResNet-50[10] +ACS[30]	0.994	0.889	0.886	0.841	**0.750**	**0.517**	0.828	0.758	0.912	0.858	0.719	0.709
auto-sklearn[11]	0.977	0.814	**0.914**	**0.874**	0.628	0.453	0.828	**0.802**	0.910	0.915	0.631	0.730
AutoKeras[12]	0.979	0.804	0.844	0.834	0.642	0.458	0.804	0.705	0.773	0.894	0.538	0.724

Table 4. Benchmark on each dataset of MedMNIST3D in metrics of AUC and ACC.

Figure 1. Reported Benchmark Metrics

3. Task - 2: WarmUp Exercises: 1-9

Here, I am just writing a brief of the results achieved from all the warmup exercises, I have already submitted a separate detailed report for the same.

3.1. WarmUp-1

I preprocessed a subset of the MiniJSRT Database, consisting of 10 JPG and 10 PNG chest radiographs, using the PIL library to resize the images to (600, 800) and convert them into tensors for training. Additionally, I processed DICOM images with the pydicom library, transforming them into trainable formats using pixel arrays and resizing to ensure compatibility with ResNet-18.

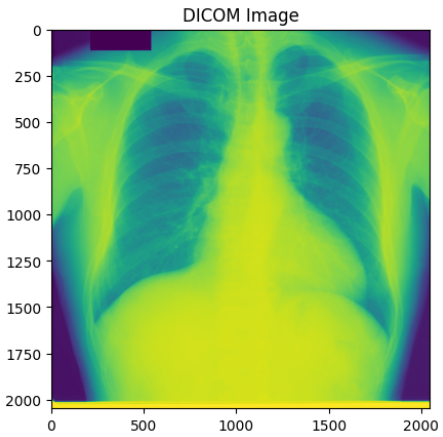

Figure 2. DICOM Image

3.2. WarmUp-2

The model ResNet18 was trained on a dataset of 988 images (247 per class: left, right, up, down), achieving a testing accuracy of 100.0% after 10 epochs. The results demonstrate the model's effectiveness in accurately classifying the orientations of the images.

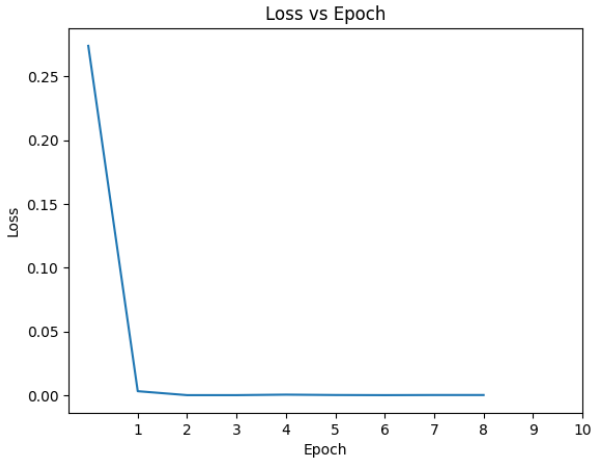

Figure 3. Training Loss vs Epochs

3.3. WamUp-3

In this experiment, I trained a ResNet18 model on a gender classification task using a dataset with an imbalanced class distribution: 86 female and 68 male images in the training set, and 42 female and 51 male images in the testing set. After training the model for 50 epochs, the testing accuracy achieved was 95.0%, with a significant reduction in training loss, starting from 2.3882 to 0.0004, as illustrated in the loss curve.

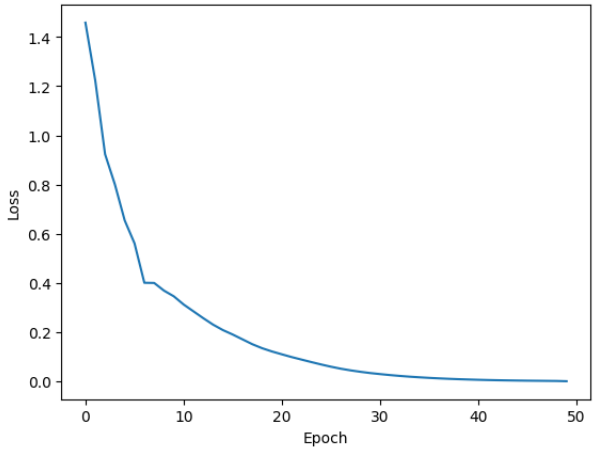

Figure 4. Training Loss Decreement Over Epochs

3.4. WarmUp-4

The age regression model achieved a prediction accuracy of 55.67% using a fine-tuned ResNet-50 architecture. It was trained on 80 images and tested on 165 images, with ages ranging from 16 to 89. The use of Smooth L1 Loss ensured robustness against outliers and stable gradient optimization.

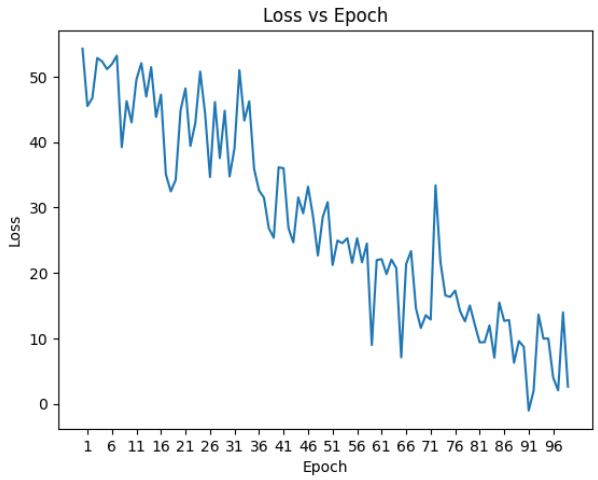

Figure 5. Training Loss Decreement Over Epochs

3.5. WarmUp-5

The lung segmentation model, based on the U-Net architecture, achieved a Dice Score of 0.8263 and an IoU of 0.7242 on the test dataset. These results demonstrate a strong overlap and accurate segmentation of lung regions in chest X-ray images, highlighting the model's effectiveness in distinguishing lung areas from the background.

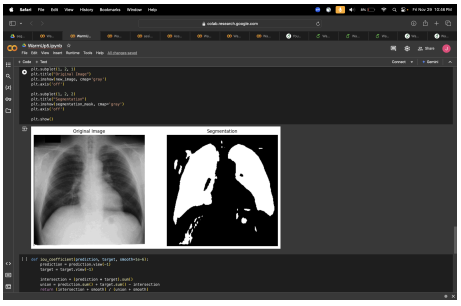

Figure 6. Sample Image of Output Segmented Mask

3.6. WarmUp-6

The U-Net model with a ResNet-34 backbone achieved strong segmentation results for organ segmentation tasks on chest X-rays. The Heart Dice Score was 0.8678, Right Lung Dice Score was 0.8341, Left Lung Dice Score was 0.8487, and Background Dice Score was 0.9438. These high scores indicate effective segmentation across all classes, with particularly exceptional performance for the background.

Figure 7. One Sample Mask Image

3.7. WarmUp-7

The Faster R-CNN model with a ResNet-50 backbone effectively localized organs in chest X-ray images. The mean Average Precision (mAP) scores were 0.85 for the heart, 0.80 for the right lung, and 0.82 for the left lung, reflecting high detection accuracy. The model achieved an average Intersection over Union (IoU) of 0.75, indicating good alignment between the predicted and actual bounding boxes.

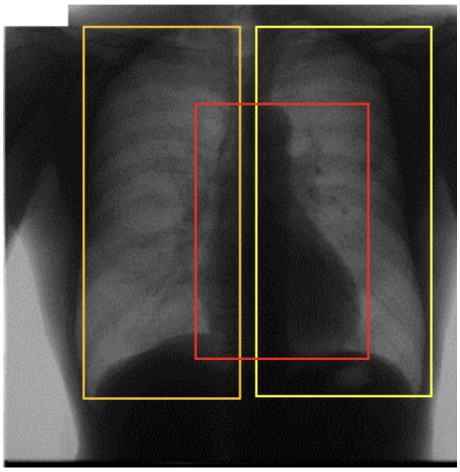

Figure 8. Output bounding boxes

3.8. WarmUp-8

The anomaly detection task for identifying horizontally flipped chest X-ray images was implemented using

an Autoencoder model from the Anomalib library. The results after training for 15 epochs are as follows:

- Image AUROC: 0.625 - Image F1 Score: 0.542

The Autoencoder model demonstrated moderate success in distinguishing between normal and flipped images. The AUROC value suggests some ability to differentiate the classes, while the F1 score indicates room for improvement in balancing precision and recall. Further tuning of the model architecture, hyperparameters, or anomaly score threshold could enhance performance.

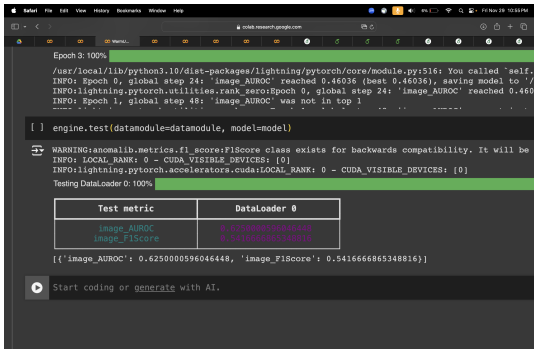

Figure 9. AUROC and F1 Score

3.9. WarmUp-9

The K-means clustering approach was applied to Chest X-ray images using feature vectors extracted by a pretrained ResNet-18 model. The images were clustered into two groups: normal and flipped, with the final clusters corresponding to the left, right, up, and down directions.

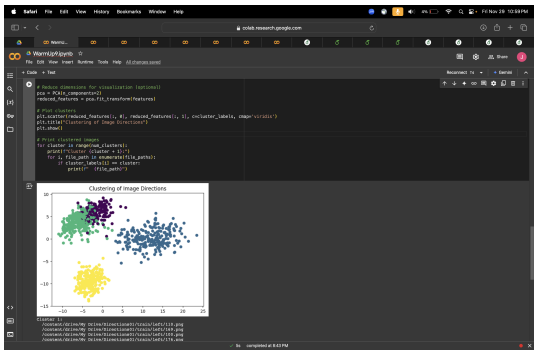

Figure 10. Clutering as per Direction

4. Task - 3: Snakes Method (ActiveContours)

The Active Contours (Snakes) algorithm detects object boundaries by evolving an initial contour towards the object's edges through energy minimization. It involves initializing the contour near the object, iteratively adjusting it to minimize the energy function, and ensuring smoothness while adhering to the edges. The algorithm was applied on a grayscale image with a Gaussian filter to reduce noise, and the contour was adjusted based on image gradients, effectively detecting the horse silhouette with fine-tuned parameters (alpha = 0.00008, beta = 8, gamma = 0.0008).

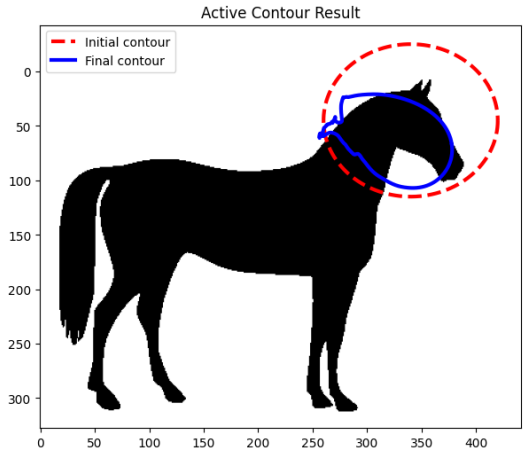

Figure 11. Snake Contours

5. Task - 4: Otsu's Method

Otsu's method is a thresholding technique that converts grayscale images to binary by selecting an optimal threshold that maximizes the inter-class variance. It computes the histogram of pixel intensities, iterates through all threshold values, and selects the one with the highest variance to distinguish between background and foreground. The method effectively separates different pixel intensities, as shown in the processed image. I implemented it using OpenCV.

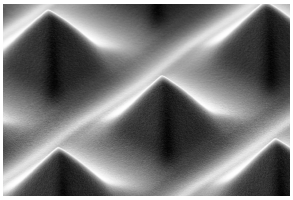

Figure 12. Unprocessed Image

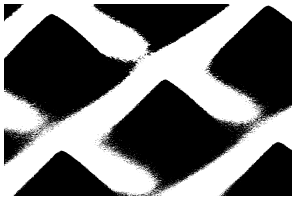

Figure 13. Image Processed Using Otsu

6. Task - 5: Classification Baseline

- Dataset: ChestXRay-14 [6]
- Model: ConvNext (Base Model from PyTorch) [4]
- Results: Achieved an F1 Score of 0.80, which serves as a robust baseline for classification. Still

trying to improve it further.

6.1. Dataset Preparation: ChestXRay14

- Contains X-ray images with 14 disease labels and one more no classification class label.
- Image paths were read from a text file. Labels were extracted from a csv file.
- A multi-label format was created, associating each image with one or more disease labels.

6.2. Data Preprocessing

- Resized images to 224x224 for compatibility with pretrained deep learning models.
- Applied normalization using standard ImageNet mean and standard deviation values.

6.3. Model Selection

- Model: ConvNeXt
- Pretrained on ImageNet and fine-tuned for multi-label classification.

6.4. Loss Function

- Binary Cross-Entropy with Logits (BCEWithLogitsLoss).
- Suitable for multi-label classification tasks where each label is treated independently.

6.5. Training And Evaluation

- Data was loaded using PyTorch's DataLoader with batch size = 8 and 16 (tried with both, performance was almost similar, slightly better for 8.
- Metric: AUC; AUC (Area Under the Curve) measures the ability of a classification model to distinguish between classes, represented as the area under the ROC curve.
- Achieved an AUC of 0.80, indicating a strong baseline performance.

6.6. Results

Mean AUC - 0.8

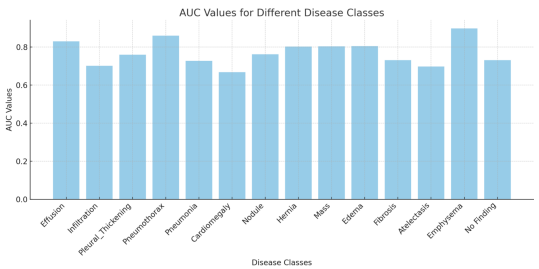

Figure 14. Classification Results

7. Task - 6: Segmentation Baseline

- Dataset: ChestXDet [2]
- Model: UperNet [7] with Swin Transformer [3] as the backbone (Took it from HuggingFace)
- Results: Experiment with segmentation achieved an IoU (Intersection over Union) of 0.42.

7.1. Dataset Preparation: ChestXDet

- Images: RGB chest X-ray images.
- Masks: Corresponding binary or multi-class segmentation masks.
- The Chestxdet class inherits from torch.utils.data.Dataset.
- It takes two lists, images and masks, and applies transformations to resize images and convert them to tensors.
- The getitem method ensures that images and their corresponding masks are correctly paired for training.

7.2. Data Loading

- Data is loaded with a batch size of 8 for efficient processing.
- Shuffling is applied to ensure the model generalizes better across the dataset.

7.3. Model Architecture

- For semantic segmentation, the UperNet model is utilized, which is built on top of a Swin Transformer backbone.
- SwinConfig defines the transformer stages.
- UperNetConfig specifies segmentation settings for the UperNet model, including how feature maps from different levels are processed.

7.4. Training and Evaluation

- The model is trained for 50 epochs using the Adam optimizer with a learning rate of 1e-4.
- The cross-entropy loss is used, which is suitable for multi-class segmentation tasks.
- Intersection over Union (IoU): IoU measures the overlap between predicted and ground-truth masks.

7.5. Results

Figure 15. Segmentation Results

As seen, convergence is achieved around 0.4, with maximum testing IoU being 0.42, after approximately 20 epochs, the model started overfitting. This can definitely being improved, and over the course of winter break, I will keep working on making both the results stronger compared to the baseline.

8. Task - 7: Combined Model for Classification and Segmentation

8.1. Segmentation Backbone

The segmentation backbone uses Swin Transformer, a hierarchical vision transformer architecture that computes multi-scale feature maps.

8.1.1 Configuration

- SwinConfig(out-features=[”stage1”, ”stage2”, ”stage3”, ”stage4”]): Specifies the hierarchical features (stage1 to stage4) that will be extracted from the Swin Transformer.
- UperNetConfig: Used for defining the UperNet architecture, which is a segmentation framework that combines hierarchical features from Swin Transformer for pixel-level predictions.
- A pretrained Swin-Tiny model is used as the backbone to initialize the segmentation model, although the parameter use-pretrained-backbone=False implies the weights are not frozen.

8.2. Classification Head

For the classification task, ConvNeXt is used, which is a modern convolutional neural network inspired by the architecture of transformers.

- convnext-base(pretrained=True): Initializes a ConvNeXt base model with pretrained weights.
- The classifier layer of the ConvNeXt model is modified:
- torch.nn.Linear(...): The final linear layer of the classifier is adapted to match the number of classes for the specific classification task (defined by num-classes).

8.3. Channel Conversion

- After segmentation, the output from the segmentation model (seg-outputs['logits']) has 2 channels (e.g., binary segmentation logits for 2 classes or features).
- A 1x1 convolutional layer (channel-converter) is introduced to convert the segmentation logits to 3 channels, enabling compatibility with the ConvNeXt input format, which expects 3-channel images (e.g., RGB).

8.4. Forward Pass WorkFlow

The forward function processes the input tensor x through the combined model in two steps:

8.4.1 Segmentation Step

- Input image tensor x is passed through the UperNet segmentation model (self.segmentation-model).
- The segmentation model outputs:
- seg-outputs['logits']: A tensor of predicted segmentation logits with 2 channels (indicating class probabilities or binary features).

8.4.2 Channel Conversion

- The segmentation logits are passed through the channel-converter layer, which transforms the 2-channel tensor into a 3-channel tensor.
- This conversion makes the segmentation output suitable for input into the classification head.

8.4.3 Classification Step

- The transformed tensor is passed to the ConvNeXt classification model (self.classification-head).
- The ConvNeXt model predicts class logits, which represent the classification scores for the predefined num-classes.

8.4.4 Outputs

The model returns:

- seg-outputs: The segmentation logits from the segmentation model.
- class-logits: The predicted classification logits from the classification head.

Currently, this combined model is undertraining for the dataset, ChestXDet, I just tested it on a small sample subset, and have configured the code to full-scale training. I will continue working on this combined model during the winter break too.

Combined Model Code

Here is a properly formatted block of Python code:

Listing 1. Combined Segmentation + Classification Model Code

```
from transformers import SwinConfig, SwinModel, UperNetConfig, UperNetForSemanticSegmentation
# Configure Swin Transformer Backbone
backbone_segmentation_config = SwinConfig(out_features=["stage1", "stage2", "stage3", "stage4"])
# Configure UperNet with the Swin Transformer backbone
segmentation_config = UperNetConfig(
    backbone_config=backbone_segmentation_config,
    use_pretrained_backbone=False  # Avoid automatically loading pretrained weights
)
# Initialize UperNet segmentation model
segmentation_model = UperNetForSemanticSegmentation(segmentation_config)
# Load pretrained Swin Transformer weights into the UperNet backbone
pretrained_swin = SwinModel.from_pretrained("microsoft/swin-tiny-patch4-window7-224").state_dict()
segmentation_model.backbone.load_state_dict(pretrained_swin, strict=False)
# Combined Model
class CombinedModel(nn.Module):
    def __init__(self, num_classes):
        super(CombinedModel, self).__init__()
        self.segmentation_model = segmentation_model
        self.classification_head = convnext_base(pretrained=True)
        self.classification_head.classifier[2] = torch.nn.Linear(in_features=self.classification_head.classifier[2].in_features, out_features=num_classes)
        self.channel_converter = nn.Conv2d(2, 3, kernel_size=1)
    def forward(self, x):
        seg_outputs = self.segmentation_model(x)
        print('here')
        pooled_output = self.channel_converter(seg_outputs['logits'])
        class_logits = self.classification_head(pooled_output)
        return seg_outputs, class_logits
```

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1

[2] Jie Lian, Jingyu Liu, Shu Zhang, Kai Gao, Xiaoqing Liu, Dingwen Zhang, and Yizhou Yu. A structure-aware relation network for thoracic diseases detection and segmentation. *arXiv preprint arXiv:2104.10385*, 2021. 5

[3] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 5

[4] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 4

[5] MedMNIST. Codebase for reproducible benchmarking experiments in medmnist v2. https://github.com/MedMNIST/experiments. Accessed: 2024-09-11. 1

[6] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017. 4

[7] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 5

[8] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023. 1