Pattern Recognition
Assignment 2

Judy Nabil 7575
Amr Abdelaziz 7447
Ahmed Hany 7387
Mahmoud Haytham 7560
Faculty of Engineering, Alexandria University

March 2025

1 Problem Statement

This project focuses on multiclass classification and segmentation tasks using the HAM10000 dataset. The classes included are:

- MEL: Melanoma
- NV: Melanocytic nevi
- BCC: Basal cell carcinoma
- AKIEC: Actinic keratoses and intraepithelial carcinoma
- BKL: Benign keratosis-like lesions
- DF: Dermatofibroma
- VASC: Vascular lesions

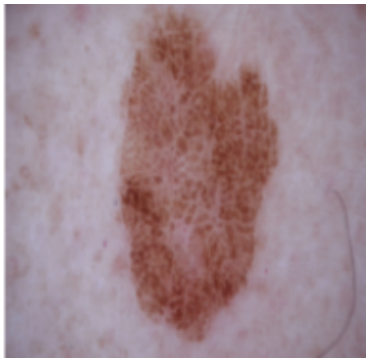
2 Data Loading and Splitting

Utilizing Kaggle's built-in python notebook environment, the dataset could be directly accessed.
After reading the csv file using the *pandas* library, the classes for each image were mapped to a number from 0 to 6 representing the corresponding label. Below are a few samples from the dataset, their string labels, and corresponding numerical label.

	image	MEL	NV	BCC	AKIEC	BKL	DF	VASC	label	label_idx
0	ISIC_0024306	0.0	1.0	0.0	0.0	0.0	0.0	0.0	NV	1
1	ISIC_0024307	0.0	1.0	0.0	0.0	0.0	0.0	0.0	NV	1
2	ISIC_0024308	0.0	1.0	0.0	0.0	0.0	0.0	0.0	NV	1
3	ISIC_0024309	0.0	1.0	0.0	0.0	0.0	0.0	0.0	NV	1
4	ISIC_0024310	1.0	0.0	0.0	0.0	0.0	0.0	0.0	MEL	0

Figure 1: Dataset samples

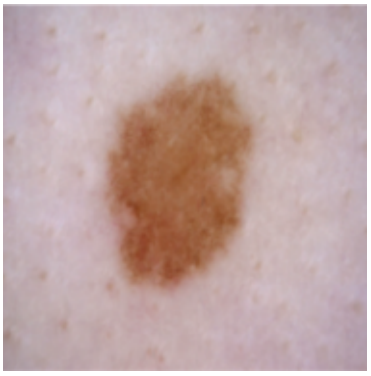
After loading the dataset and mapping the labels, we proceeded to split it into a 70:15:15 split corresponding to training:validation:test. The decision for the split ratio was based on the size of the dataset it self (10,000 images) which was relatively small.
The split was stratified according to dataset's labels to account for the serious imbalance in the number of samples for each class.
Below are a few samples from the test set and their segmentation masks:



(a) Image

(b) Segmentation Mask

Figure 2: Sample 1



(a) Image

(b) Segmentation Mask

Figure 3: Sample 2

3 Data Processing

Utilizing the *Dataset* and *DataLoader* classes from the *torch* library, the *transforms* class from the *torchvision* library, and the *os* library, we processed the data to a form suitable for our tasks.

- Six instances of the Dataset class were created using the train, validation, and test dataframes obtained after splitting the previous steps were performed. The first three instances were created for the classification task and the latter three were created for the segmentation task.
- Lazy loading was employed to prevent memory issues during model training and a 256x256 resize transform was added to the getitem method to increase model training speed and reduce memory usage.
- Six instances of the DataLoader class were created using the previously created instances of the Dataset classes with a batch size value of 32 for the classification task and 16 for the segmentation task. Shuffle was enabled only for the training DataLoaders.

4 Classification Model

4.1 Model Details

Our model of choice for the classification task was the ResNet18 model. This model has a total of 11180103 (11 million) parameters and is great for classification task. It can also be used as an encoder for segmentation tasks.

We modified the output layer of the model to account for the difference between the pre-training dataset and our dataset. Our output layer is a regular FCN layer with 7 neurons, one for each unique label in the dataset.

We implemented an early stopping function that monitors the validation loss for each epoch of fine-tuning and stops the model if its performance has not improved after a set number of epochs.

4.2 Model Performance

First we have the Loss-Accuracy curves for the train and validation sets:

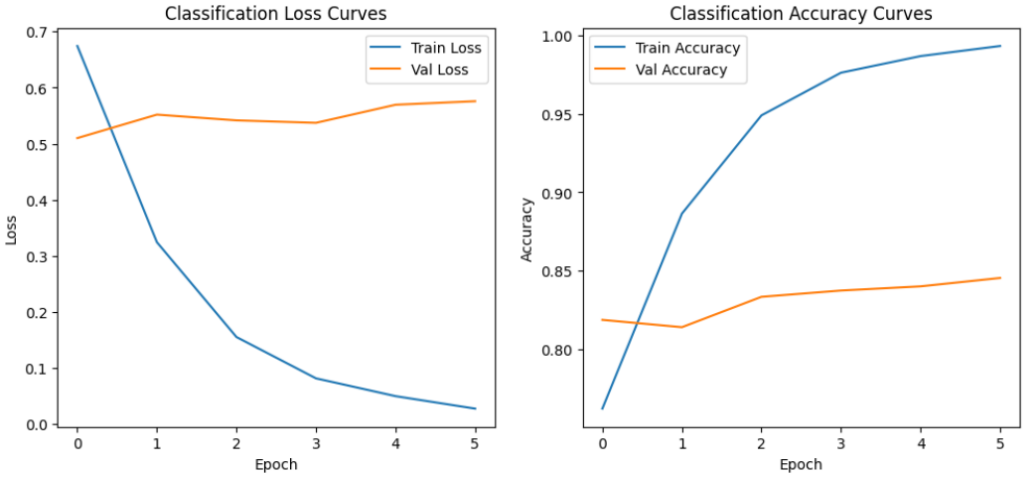


Figure 4: Classification Curved

The model achieved an overall accuracy of 0.8429807052561543 on the test set. Below are the tabulated per-class metrics for the test set:

Class	Precision	Recall	F1 Score	Support
MEL	0.66	0.57	0.61	167
NV	0.88	0.96	0.92	1006
BCC	0.78	0.83	0.81	77
AKIEC	0.76	0.39	0.51	49
BKL	0.76	0.56	0.64	165
DF	0.72	0.76	0.74	17
VASC	0.94	0.68	0.79	22

Average	Precision	Recall	F1 Score	Support
Macro	0.79	0.68	0.72	1503
Weighted Macro	0.83	0.84	0.83	1503

Finally, we have the confusion matrix for the test set:

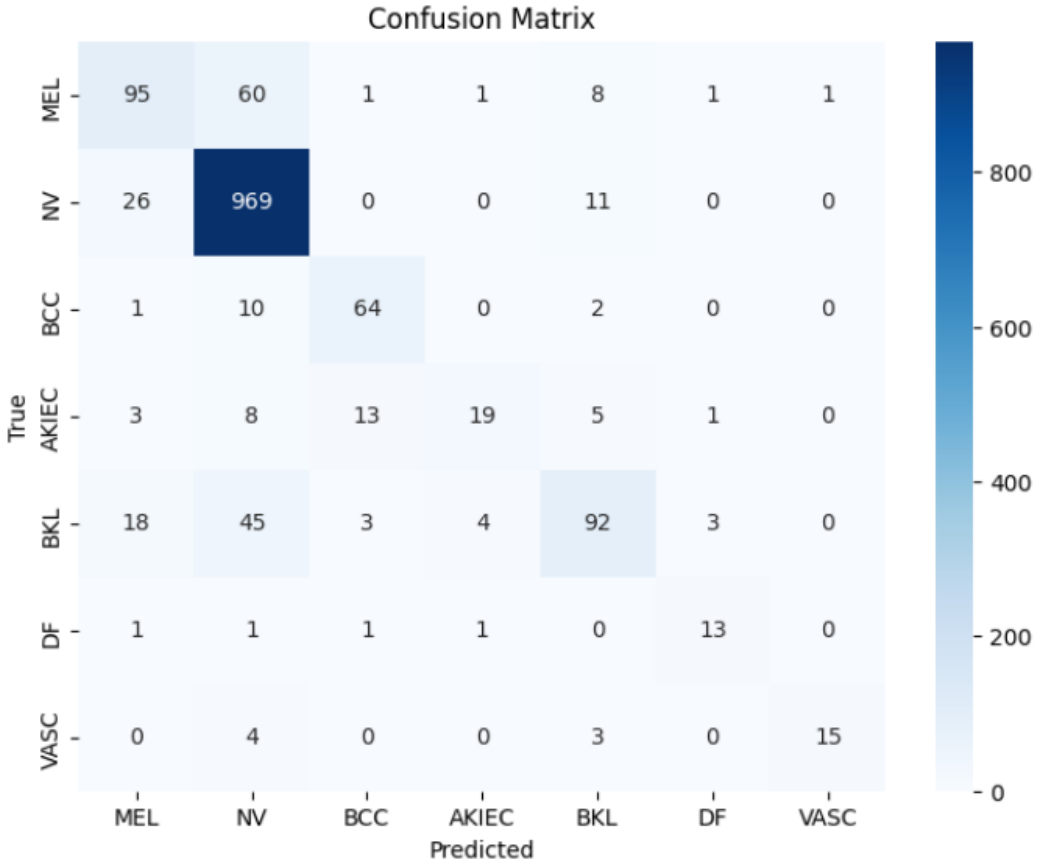


Figure 5: Classification Confusion Matrix

5 Segmentation Model

5.1 Model Details

For the segmentation task, we built the UNET segmentation model using *torch*. The model takes an RGB image input and outputs a single mask highlighting the detected cancer patch.
Note that this model output is only a mask without an associated class.

Early stopping was also utilized for this model's training.

5.2 Model Performance

First we have the Loss-Accuracy, IoU, and Dice Coefficient curves for the train and validation sets:

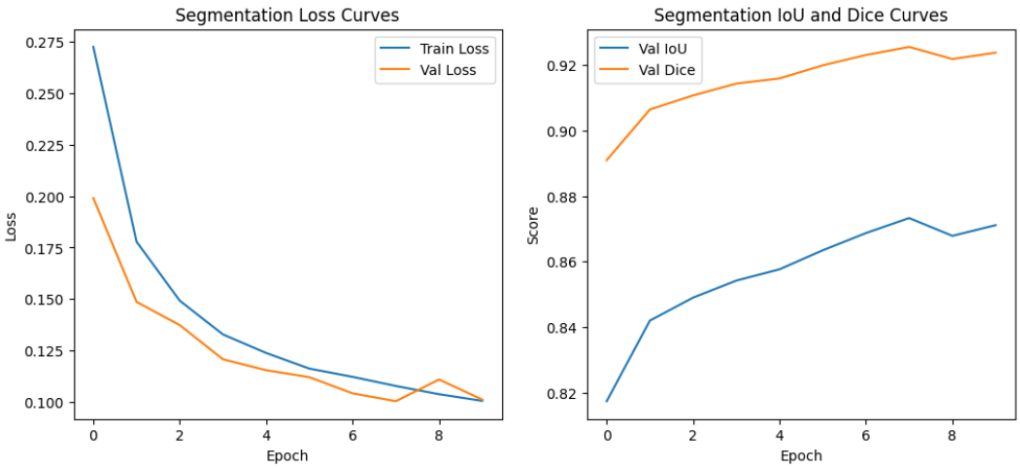


Figure 6: Segmentation Curves

The model achieved the following on the test set:

- Loss: 0.1101
- IoU: 0.8621
- Dice Coefficient: 0.9165

Below are a few samples from the test set and their predicted masks:

(a) Image (b) True Mask (c) Predicted Mask

Figure 7: Segmentation Result 1

(a) Image (b) True Mask (c) Predicted Mask

Figure 8: Segmentation Result 2

6 Bonus: Utilize a Single Model for Both Scenarios

6.1 Model Details

To fulfill the task requirements, we opted for an encoder-decoder model archetype with a branched decoder.

Our encoder of choice is the ResNet18 network. It is responsible for extracting the features from the input images to be used for the classification and segmentation tasks later on.

We have two branches in the decoder, one for each task:

- Classification branch: This branch takes the encoder output and passes it through an average pooling layer before passing it through a fully connected layer that outputs the probabilities for each class.
- Segmentation branch: Similar to the UNET decoder, but without the encoder-decoder residuals. It takes the encoder output and passes it through a series of transpose convolutions and convolution layers before finally outputting the predicted mask for the input image.

6.2 Model Performance

6.2.1 Classification Task

The model achieved an accuracy of 0.8449767132401863 on the test set.

Below are the tabulated metrics for the test set:

Class	Precision	Recall	F1-Score	Support
MEL	0.77	0.36	0.49	167
NV	0.88	0.97	0.93	1006
BCC	0.84	0.68	0.75	77
AKIEC	0.52	0.78	0.62	49
BKL	0.79	0.68	0.73	165
DF	0.67	0.71	0.69	17
VASC	0.84	0.73	0.78	22

Average	Precision	Recall	F1-Score	Support
Macro Avg	0.76	0.70	0.71	1503
Weighted Avg	0.84	0.84	0.83	1503

Below is the confusion matrix for the test set:

Table 1: Confusion Matrix

Actual \Predicted	**Predicted Class**						
	MEL	NV	BCC	AKIEC	BKL	DF	VASC
MEL	60	82	1	8	12	2	2
NV	7	980	2	5	12	0	0
BCC	0	10	52	12	2	1	0
AKIEC	2	0	5	38	2	1	1
BKL	8	34	2	8	112	1	0
DF	1	1	0	2	1	12	0
VASC	0	5	0	0	0	1	16

6.2.2 Segmentation Task

The model achieved the following metrics on the test set:

- Loss: 0.3912
- IoU: 0.4730
- Dice Coefficient: 0.6216

Below are a few samples from the test set and their predicted masks:

(a) Image (b) True Mask (c) Predicted Mask

Figure 9: Segmentation Result 1

(a) Image (b) True Mask (c) Predicted Mask

Figure 10: Segmentation Result 2