**Final Report**

1) Model design and justification

Following is the model design

Input Image

(512x512x3)

ConvNEXT Encoder

(shared backbone)

Gated MLP

(dynamically routes

Tasks to expert heads)

Segmentation head

Classification head

For this task I have chosen ConvNEXT encoder to extract visual features from retinal images.

ConvNEXT offers strong performance on high-resolution data and has pretrained weights on Imagenet, which can enable better feature reuse.

A gating network (router) is introduced, this tries to predict the task (segmentation / classification ) for each image. It consists of a MLP , trained using cross-entropy loss against the ground truth task vector, encouraging model to learn routing behavior rather than relying on predefined rules.

Two task specific heads are used one for lesion segmentation and another for disease grading (retinopathy and edema severity).

**Segmentation head :**

The segmentation head is designed to predict multi-label lesion maps for five retinal findings: MA, HE, EX, SE, and OD. The segmentation head follows a lightweight U-Net-style decoder built to upsample high-level features from the ConvNeXt encoder back to full-resolution lesion maps

Here the decoder consists of 4 sequential Upblocks each performing:

ConvTranspose2d (ker=2,st=2), this upsamples resoultion, ReLu, Conv2d and then 1x1 conv2d that projects 32 channel output to 5 (classes). Interpolate resizes the output to native resolution (512x512).

Flow:

Input: Encoder output (B, 768, 13, 13) ->UpBlock1 (768 → 256, 26×26) ->UpBlock2 (256 → 128, 52×52) ->UpBlock3 (128 → 64, 104×104) ->UpBlock4 (64 → 32, 208×208) - >Final Conv2d (32 → 5) + Interpolation

Output: Segmentation maps (B, 5, 512, 512)

Head is optimized using binary cross entropy + dice loss.

**Classification head :**

This head is responsible for predicting disease grading, head contains two independent branches—one for retinopathy level (5 classes) and another for edema severity (3 classes). Each branch consists of a global average pooling layer, followed by an MLP (Linear → ReLU → Dropout → Linear → output : { fc\_retino(x) → shape (B, 5) — logits for 5 DR classes

fc\_edema(x) → shape (B, 3) — logits for 3 edema classes}

Both heads share the same encoder output but are functionally isolated. This separation prevents gradients from one task interfering with the optimization of the other, which is especially important when task labels are mutually exclusive in a multitask setup.

2) Dataset handling and preprocessing steps

The model is trained on the IDRiD dataset, which contains two distinct subsets:

1) Segmentation samples: with pixel-level annotations for five lesion types

2) Classification samples: with labels for diabetic retinopathy (DR) and diabetic macular edema (DME) severity.

These two tasks are mutually exclusive per image in the original dataset — segmentation images do not have DR labels, and vice versa. Therefore, training a dynamic multitask model required a unified and controlled way to sample both types.

To manage this, I constructed a custom metadata JSON file that consolidates:

1) The image path

2)Task vector: [1, 0] for segmentation, [0, 1] for classification

3)For segmentation: file paths to all 5 lesion masks

4)For classification: integer labels for retinopathy (0–4) and edema (0–2)

Each entry represents a single training sample with exactly one task type.

Sample : [

{

"image\_path": "/content/drive/MyDrive/idrid/1. Original Images/a. Training Set/IDRiD\_378.jpg",

"task": "cls",

"task\_vector": [

0,

1

],

"retino\_label": 0,

"edema\_label": 0

},

{

"image\_path": "/content/drive/MyDrive/idrid/A. Segmentation/1. Original Images/a. Training Set/IDRiD\_46.jpg",

"task": "seg",

"task\_vector": [

1,

0

],

"mask\_paths": {

"MA": "/content/drive/MyDrive/idrid/A. Segmentation/2. All Segmentation Groundtruths/a. Training Set/1. Microaneurysms/IDRiD\_46\_MA.tif",

"HE": "/content/drive/MyDrive/idrid/A. Segmentation/2. All Segmentation Groundtruths/a. Training Set/2. Haemorrhages/IDRiD\_46\_HE.tif",

"EX": "/content/drive/MyDrive/idrid/A. Segmentation/2. All Segmentation Groundtruths/a. Training Set/3. Hard Exudates/IDRiD\_46\_EX.tif",

"SE": "/content/drive/MyDrive/idrid/A. Segmentation/2. All Segmentation Groundtruths/a. Training Set/4. Soft Exudates/IDRiD\_46\_SE.tif",

"OD": "/content/drive/MyDrive/idrid/A. Segmentation/2. All Segmentation Groundtruths/a. Training Set/5. Optic Disc/IDRiD\_46\_OD.tif"

}

Since dataset is highly imbalanced, (few seg) this metadata allowed me to balance task type across the dataset. Having this single metadata helped me build MultiTaskDataset that dynamically loads either segmentation or classification data in the same branch.

Here task vector servers as ground truth for the gating network which learns to route images to the correct expert head. By adding label and paths helped in visualization,evaluation and prediction.

3) Training and evaluation strategy

The model is trained using a joint multitask setup where each input image is assigned to either a segmentation or classification task. A single model performs both tasks, and the appropriate expert head is supervised based on the image's ground truth task type.

During each training step, the image is first passed through a shared ConvNeXt encoder, and the resulting features are sent to:

1) A gating network (MLP) to predict which task the image belongs to.

2) Both task-specific heads (segmentation and classification), but only the correct head is updated based on the true task.

If the ground truth task is classification, only the classification loss (retinopathy + edema) is computed and backpropagated. If it is segmentation, only the binary cross-entropy + Dice loss for lesion masks is applied. The losses are weighted by the task's proportion in the current batch to keep training balanced.

The gating network is supervised with a cross-entropy loss to predict the correct task (seg or cls) using the ground truth task vector. An additional entropy penalty is added to encourage confident (low-entropy) gating decisions.

The optimizer used is AdamW with a learning rate of 1e-4, and the model is trained for 15 epochs. After training, the final model weights are saved for evaluation and visualization.

How gating network should behave :

The gating network learns to predict whether an input image belongs to the segmentation or classification task based on visual patterns in the encoder's features. For example, images with DR grade labels but no visible lesions should be routed to the classification head, while those with lesion mask annotations should go to the segmentation head.

If the gating network routes an image to the wrong expert (e.g., a segmentation image is routed to classification), no task loss is computed — the model receives no supervision for that image, but the gating loss is still computed, which teaches the router to correct its mistake over time. In simple terms: when the gate chooses wrong, the task head does nothing, but the gate gets penalized to learn better routing.

4) Performance metrics and analysis

*- Posted in final jupyter notebook*

5)Visualizations

*- Posted in final jupyter notebook*