Table of Contents

1. Introduction 2

2. Environment Setup 2
- Requirements 2
- Directory Structure 2

3. Dataset Preparation 2
- Pretraining Datasets 2
- Fine-tuning Datasets 3

4. Model Pretraining (Two-Phase) 3
- Command 3
- Key Parameters 3
- Output 3
- Verifying Pretraining Results 3

5. Classifier Training 4
- Command 4
- Key Parameters 4
- Output 4

6. Training with Masks 4
- Command 4
- Key Parameters 5
- Mask Training Process 5

7. Evaluating the Model 5
- Standard Evaluation 5
- Masked Evaluation 5
- Evaluation Outputs 5

8. Visualizing Results 5
- Loss Plots 5
- Understanding Visualizations 5

9. Tips for Optimal Performance 6
- Pretraining Strategy 6
- Fine-tuning Strategy 6
- Hyperparameters 6
- Hardware 6

10. Troubleshooting 6
- "CUDA out of memory" 6
- "Model checkpoint not found" 6
- "Poor reconstruction quality" 6
- "Low classification accuracy" 7
- Verify Model Parameters 7

1. Introduction

Sequential Dual SVQVAE is an advanced hierarchical vector quantized autoencoder that:

- Uses a sequential training approach with two stacked SVQVAEs
- First trains SVQVAE1 with 3 levels of encoding
- Then trains SVQVAE2 using concatenated encodings from SVQVAE1
- Finally trains a classifier that leverages both stacks for image classification

The model is particularly effective for medical images where certain features may be more important than others for classification.

---

2. Environment Setup

Requirements

- Python 3.8+
- PyTorch 2.0+
- torchvision
- matplotlib
- scikit-learn
- pandas
- seaborn

Directory Structure

Ensure your project has the following structure:

```
hsvqvae/
├── models/
│   ├── seq_dual_svqvae.py
│   ├── svqvae.py
│   └── vqvae.py
├── data/
│   ├── stats.py
│   └── constants.py
├── step1_pretrain_seqdual.py
├── step2_train_wbc_seqdual.py
├── step3_train_wbc_seqdual_with_mask.py
├── step1-2_check_training_seqdual.py
├── step4-1_check_acc_SequentialDualSVQVAE.py
└── step4-2_check_mask_acc_SequentialDualSVQVAE.py
```

---

3. Dataset Preparation

Pretraining Datasets

- CAM16 dataset for histopathology images
- PRCC dataset for additional general features

Fine-tuning Datasets

White blood cell datasets with different sizes:

- wbc_1: 1% training data
- wbc_10: 10% training data
- wbc_50: 50% training data
- wbc_100: 100% training data

For training with masks, ensure you have corresponding mask images in a "mask" directory parallel to your data directory.

---

4. Model Pretraining (Two-Phase)

Pretraining is done in two sequential phases: first SVQVAE1, then SVQVAE2.

Command

```
python step1_pretrain_seqdual.py \
  --batch_size 24 \
  --svqvae1_epochs 100 \
  --svqvae2_epochs 100 \
  --save_every 10 \
  --level_stagger_epochs 10 \
  --method "recon_all" \
  --description "Sequential pretraining of dual SVQVAE model"
```

Key Parameters

- --svqvae1_epochs: Number of epochs to train the first stack
- --svqvae2_epochs: Number of epochs to train the second stack
- --level_stagger_epochs: Number of epochs before training deeper levels
- --method: Reconstruction method ("recon_all" or "recon_level")
  - recon_all: Reconstruct input from each level encoding
  - recon_level: Reconstruct only previous level from each encoding

Output

Pretraining creates a directory in `runs/pretrain-seqdual-{method}-b{batch_size}-e{svqvae1_epochs}-{svqvae2_epochs}-s{level_stagger_epochs}-{timestamp}/` containing:

- checkpoints/model_phase1_{epoch}.pt
- checkpoints/model_phase2_{epoch}.pt
- checkpoints/model_svqvae1_final.pt
- checkpoints/model_final.pt
- losses.json
- model_config.py

Verifying Pretraining Results

```
python step1-2_check_training_seqdual.py
```

Update the `model_checkpoint` path to your model:

```
model_checkpoint = 'runs/pretrain-seqdual-recon_all-b24-e100-100-s10-
0422_145623/checkpoints/model_final.pt'
```

This generates visualizations in the `output/` directory showing reconstructions from both SVQVAE stacks.

---

5. Classifier Training

Command

```
python step2_train_wbc_seqdual.py \
  --batch_size 24 \
  --epochs 50 \
  --save_every 5 \
  --checkpoint "runs/pretrain-seqdual-recon_all-b24-e100-100-s10-
0422_145623/checkpoints/model_final.pt" \
  --dataset "wbc_100" \
  --alternating_epochs 2 \
  --training_phase "classifier"
```

Key Parameters

- --checkpoint: Path to the pretrained model
- --dataset: Dataset size to use (wbc_1, wbc_10, wbc_50, wbc_100)
- --training_phase: Which components to train
  - classifier: Train only the classifier
  - svqvae1: Train only SVQVAE1
  - svqvae2: Train only SVQVAE2
  - full: Train all components
- --alternating_epochs: How often to switch between classification and reconstruction training

Output

Training creates a directory in `runs/train-seqdual-{dataset}-{training_phase}-{timestamp}/` containing:

- checkpoints/seqdual_model_{epoch}.pt
- checkpoints/seqdual_best_{epoch}.pt
- checkpoints/seqdual_model_final.pt
- losses.json

---

6. Training with Masks

Command

```
python step3_train_wbc_seqdual_with_mask.py \
  --batch_size 24 \
  --epochs 50 \
  --save_every 5 \
  --checkpoint "runs/train-seqdual-wbc_100-classifier-
0422_152045/checkpoints/seqdual_best_25.pt" \
  --dataset "wbc_50" \
  --alternating_epochs 2 \
  --training_phase "full"
```

Key Parameters

- Similar to standard training, but uses masks during training
- Use `--training_phase "full"` to fine-tune the entire model

Mask Training Process

1. Uses masks from the mask directory
2. Applies random noise to masked regions
3. Trains model to be consistent between masked and unmasked versions
4. Encourages the model to focus on the unmasked areas (typically the cell of interest)

---

7. Evaluating the Model

Standard Evaluation

```
python step4-1_check_acc_SequentialDualSVQVAE.py
```

Update `model_checkpoint`:

```
model_checkpoint = 'runs/train-seqdual-wbc_100-classifier-
0422_152045/checkpoints/seqdual_best_25.pt'
```

Masked Evaluation

```
python step4-2_check_mask_acc_SequentialDualSVQVAE.py
```

Evaluation Outputs

- Confusion matrices in `output/`
- Accuracy statistics in terminal
- Training/validation loss plots
- For masked evaluation: comparison between masked and clean performance

---

8. Visualizing Results

Loss Plots

```
python step4-3_plot_loss.py
```

Update `checkpoint_file` path:

```
checkpoint_file = 'runs/train-seqdual-wbc_100-classifier-
0422_152045/checkpoints/losses.json'
```

Understanding Visualizations

- Reconstruction visualizations: image reconstruction quality
- Feature maps: insights from latent space

- Confusion matrices: classification errors
- Loss curves: training progress

---

9. Tips for Optimal Performance

Pretraining Strategy

- Use `recon_all` for better reconstructions
- Batch size: 24-32
- 100+ epochs for both SVQVAEs

Fine-tuning Strategy

- Start with classifier only
- Fine-tune full model with masks
- Use `--alternating_epochs 2`

Hyperparameters

- Image size: 512x512
- Embedding dims: [3,3,3,3,3]
- VAE channels: [128,128,128,128,128]
- Codebook size: [512,512,512,512,512]

Hardware

- GPU: 8GB+
- Pretraining time: ~8-12 hrs
- Fine-tuning: ~2-4 hrs

---

10. Troubleshooting

"CUDA out of memory"

- Reduce batch size or image size
- Use fewer embedding dims

"Model checkpoint not found"

- Check path and timestamps
- Match file structure

"Poor reconstruction quality"

- Increase embedding dims

- Train for more epochs
- Try both `recon_all` and `recon_level`

"Low classification accuracy"

- Train classifier first
- Experiment with dataset size
- Use mask training

Verify Model Parameters

```
trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Trainable parameters: {trainable_params}")
```

Expected values:

- Classifier only: ~50K–500K
- Full model: ~10M–15M