

Performance Analysis for Projection-Correction Methods in Motion Deblurring Problems

Sara Casadio, Enrico Ferraiolo, Giovanni Savoca

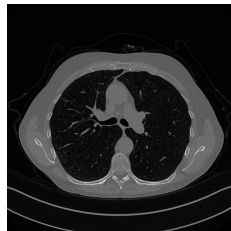
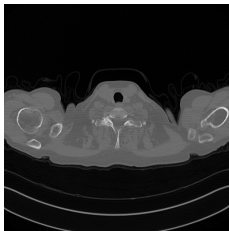
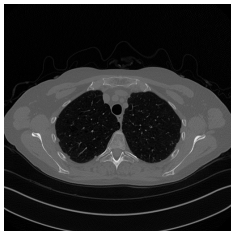
Alma Mater Studiorum - Università di Bologna
Corso di Laurea in Informatica

20 maggio 2025

- The project analyzes the performance of two **Projection-Correction** algorithms for reconstructing medical images affected by **motion blur**.
- The studied algorithms are:
 - **Diffusion Posterior Sampling (DPS)**
 - **Regularization by Denoising with Diffusion (RED-Diff)**
- Both methods are based on **pre-trained diffusion models**.
- Objective: evaluate the effectiveness of these methods in recovering degraded images.

- **Objective:** Analyze the performance of *Projection-Correction* methods **DPS** and **RED-Diff** for motion blur removal on medical images
- **Phase 1:** Dataset preprocessing (128x128)
- **Phase 2:** Data augmentation to increase dataset diversity
- **Phase 3:** Training a DDIM diffusion model on medical data
- **Phase 4:** Simulation of motion blur and its removal
- **Phase 5:** Implementation and comparison of *Projection-Correction* methods: **DPS** and **RED-Diff**
- **Phase 6:** Quantitative evaluation of performance using metrics such as **PSNR** and **SSIM**

- Released by the **Mayo Clinic** for research purposes.
- Contains chest **Computed Tomography (CT)** scans.
- Images are in **grayscale (black and white)** format.
- Focus on **lung nodules** and early detection of **lung cancer**.
- Widely used in machine learning and AI medical imaging research.



- All CT images were preprocessed and resized to a fixed dimension of **128 × 128 pixels**.
- This standardization ensures consistency for input into deep learning models.
- The dataset was **augmented** to avoid overfitting and improve model generalization (detailed later).

- **Obiettivo:** Train a denoising diffusion model (DDIM U-Net) su immagini in scala di grigi
- **Componenti principali:**
 - 1 Data Augmentation
 - 2 DataLoader
 - 3 Compilazione del modello
 - 4 Loop di training con mixed-precision

- **Base Dataset:** Dataset Mayo
 - Grayscale \rightarrow 1 channel
 - Resize images to 128×128
- **Augmentations** (8 types):
 - *None*: no transformation
 - Rotation $\pm 5^\circ$ (rotation + centering)
 - Flip horizontal
 - Gaussian noise (mean=0, std=10)
 - Salt and pepper noise (prob=2%)
 - Brightness (factor=1.2)
 - Contrast (factor=1.3)
- **Implementazione essenziale:**

- **DDPMScheduler** for training diffusion process
 - Timesteps 1000
- **DDIMScheduler** for sampling
 - Timesteps 1000

- **Why:** optimize the model for better performance
- **Usage:**

```
model = torch.compile(model)
```
- **Benefits:** improved batch throughput

- **GradScaler and autocast:**
 - GradScaler for scaling gradients
 - autocast for automatic mixed precision
- Reduce memory usage and speed up training

- ❶ Loss function: MSE
- ❷ Start the training `model.train()`
- ❸ For each epoch:
 - Move images to GPU (if available)
 - Generate noise and timesteps
 - Compute noise prediction on the input data
 - Prediction + MSE loss
 - Optimization + `scheduler.step()`
- ❹ Save validation samples to visualize the model performance during training
- ❺ Compute and log average losses
- ❻ Save model weights each epoch

- **Validation:**

- `model.eval()` to set the model to evaluation mode
- MSE loss on validation set

- **Checkpoint:**

- Save the model weights to a `.pth` file
- Update loss, PSNR and SSIM history in `history.txt`
- Monitor train vs validation loss over epochs aswell as PSNR and SSIM between the generated and original images
 - For each epoch sample 10 images from the validation set and compute the metrics

- **Loss Plot:** visualizes the training and validation loss over epochs
- **Purpose:**
 - Monitor the model's performance

Grazie per l'attenzione