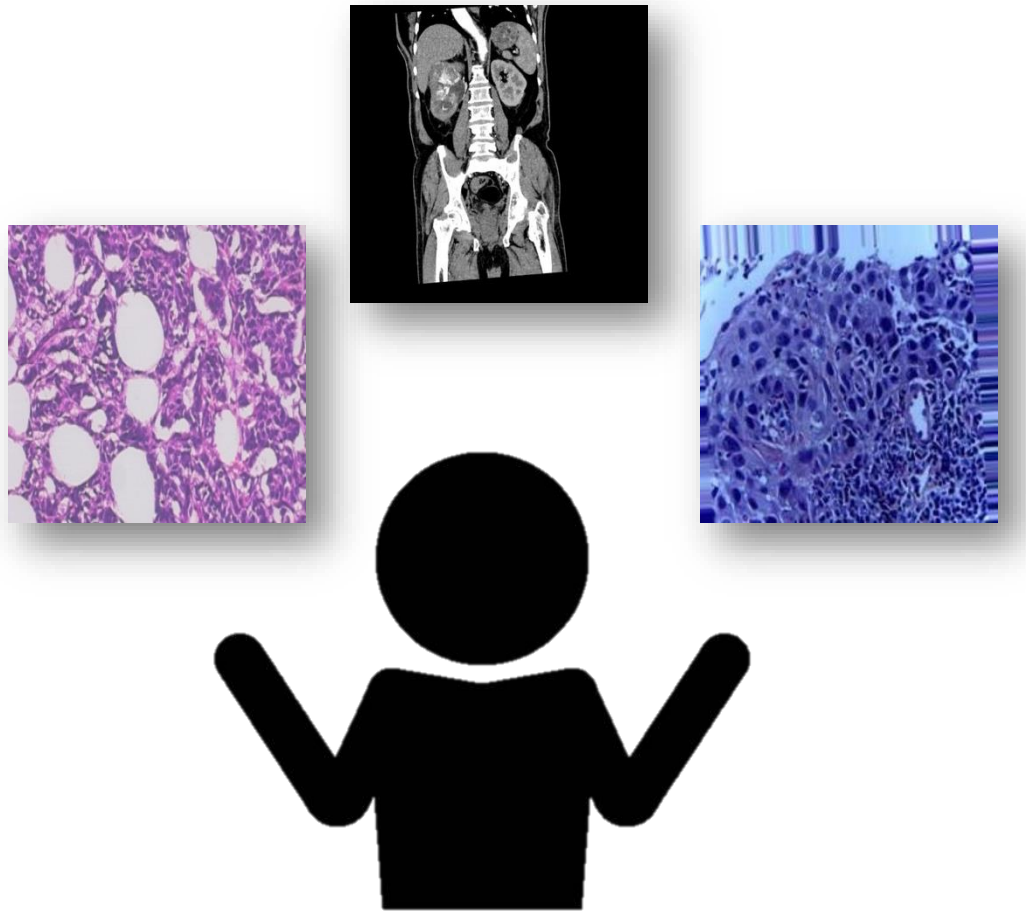
Multi cancer dataset

Valdemar, Sham og Line

11/6 -25

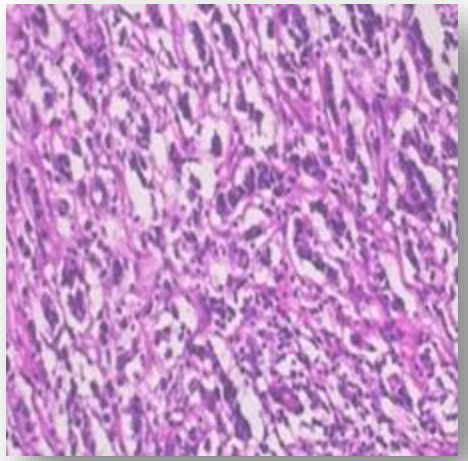
(All members contributed evenly to the project)

UNIVERSITY OF COPENHAGEN

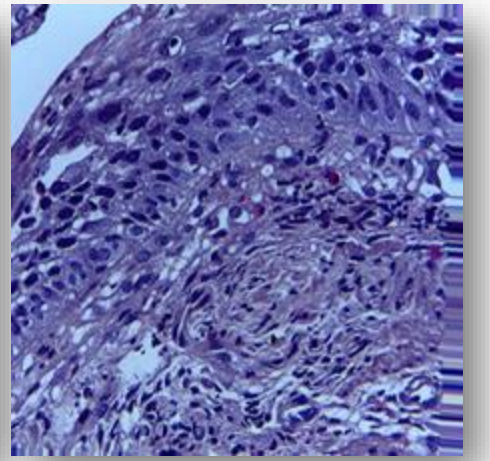


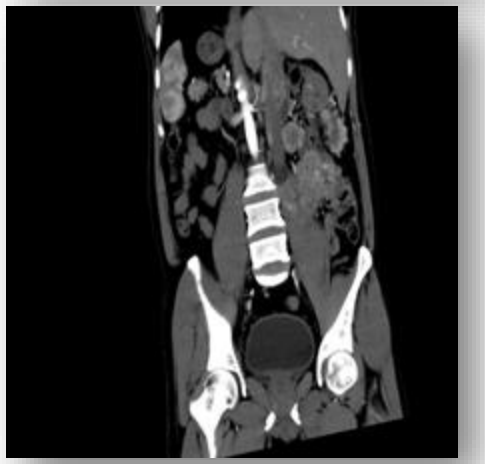
The data

Breast cancer



Oral cancer





Kidney cancer

3 datasets
(Breast, Oral and Kidney)

Binary label

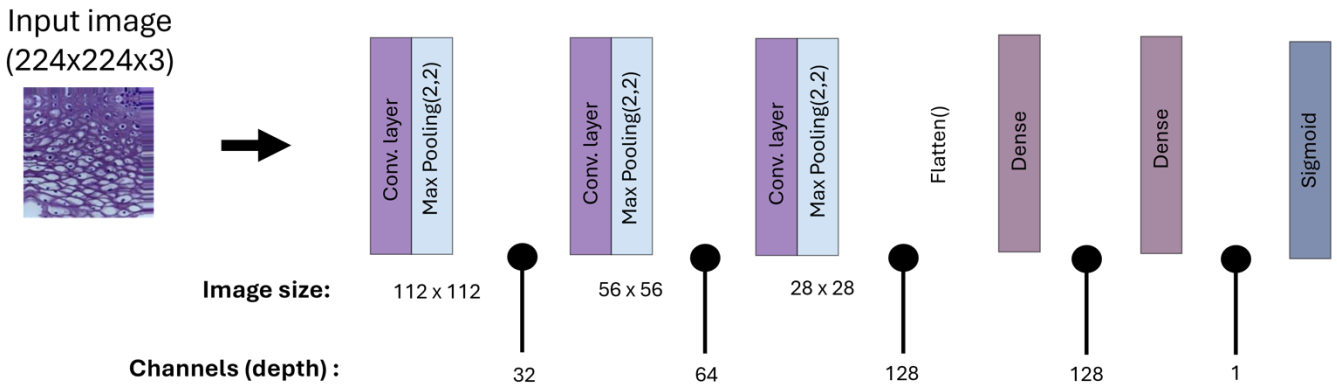
Cancer or healthy cells

30000 data entries (each dataset 10000 entries)

Images (biopsies/scans)

Data pre-processing: 512x512 → 224x224

Initial CNN



Hyper parameters:

- Dropout-rate = 0.5
- Learning rate = 0.001
- Early stopping with patience = 5

	Val accuracy	Val loss	AUC	Epochs	Training time
Oral	0.5	0.693	0.50	6	1823.80 s
Breast	0.95	0.170	0.99	7	1968.60 s
Kidney	1.0	0	1.00	6	1594.03 s

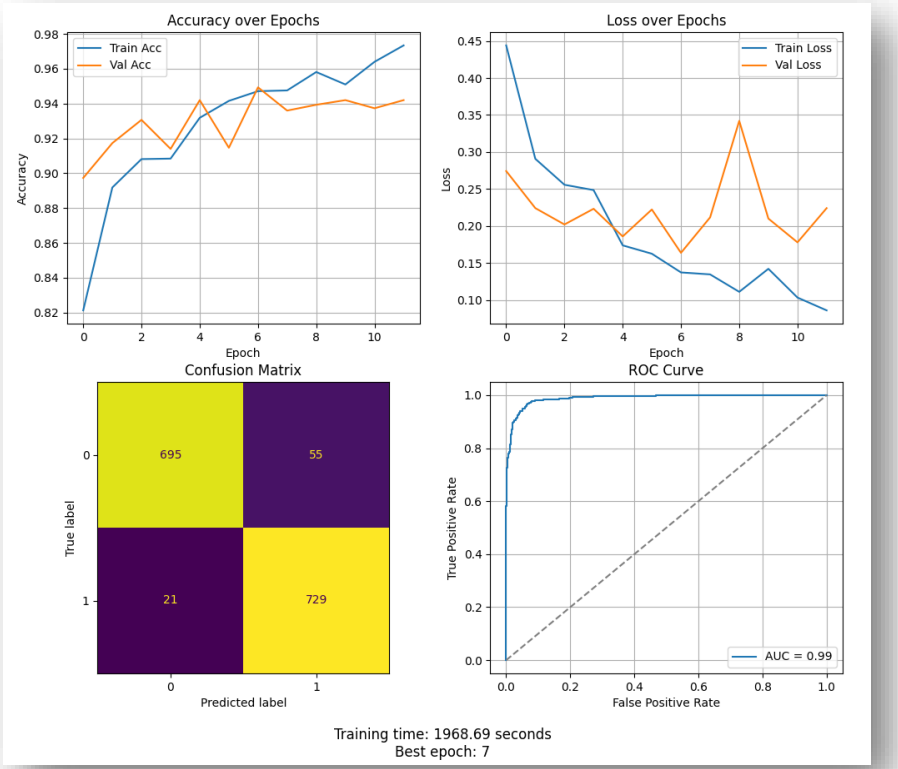
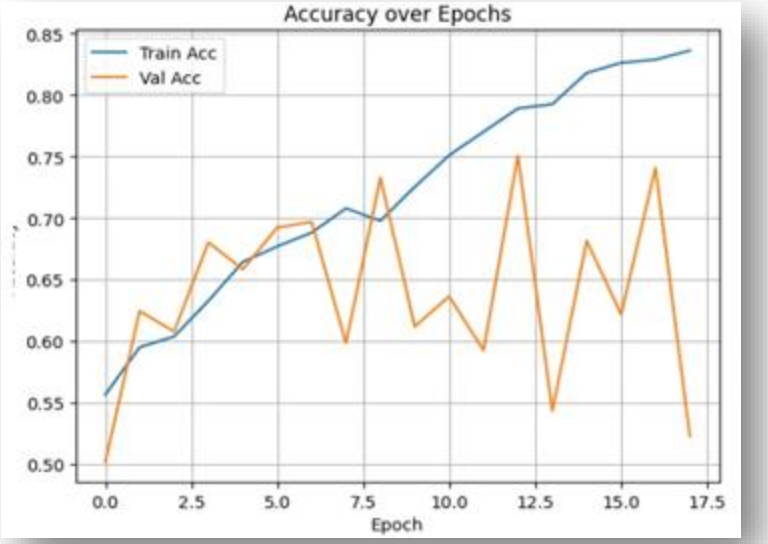
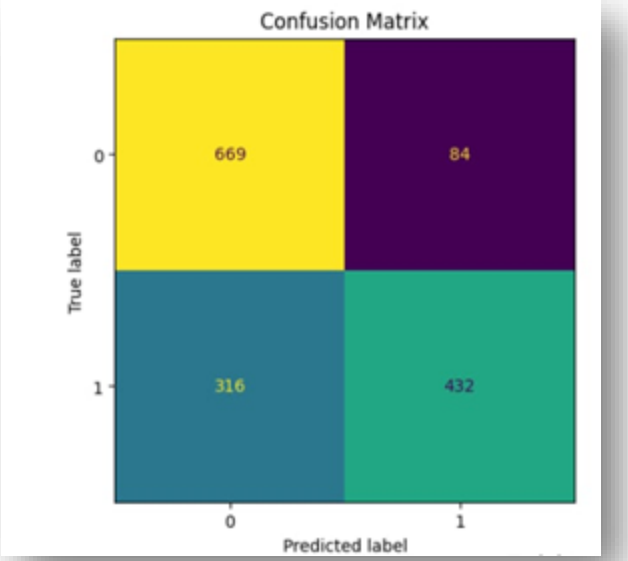


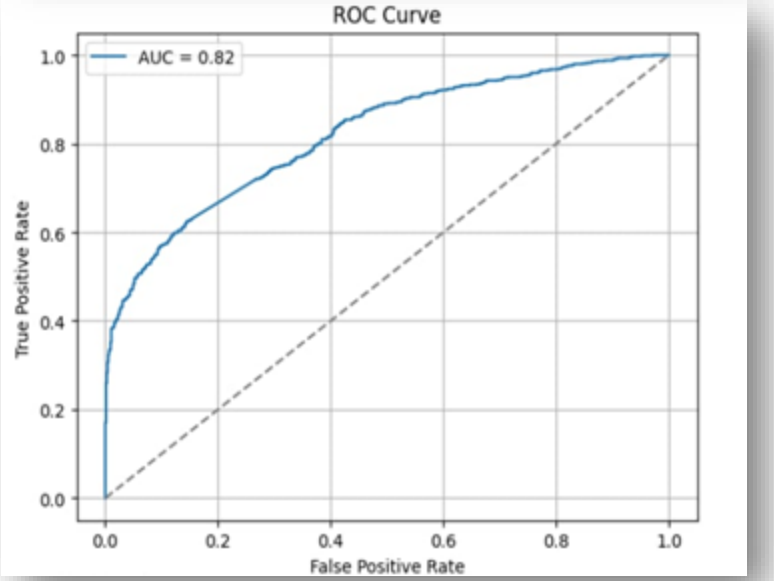
Figure: Results for the *breast cancer* data set

Group shuffle split the data

- Realistic results on *oral cancer* dataset with AUC = 0.82
- Perfect (not realistic) results on *kidney cancer* dataset with AUC = 1



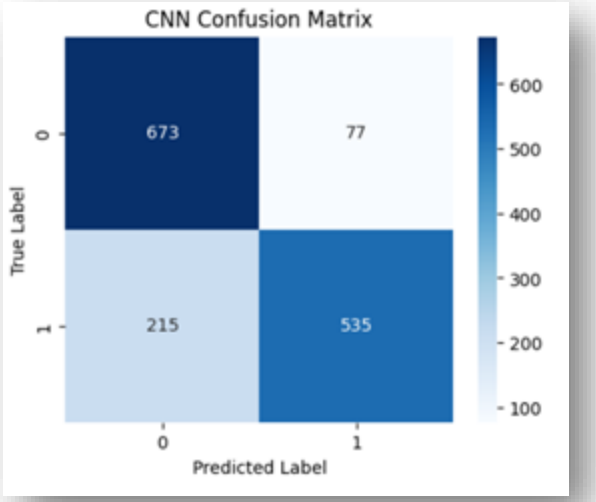


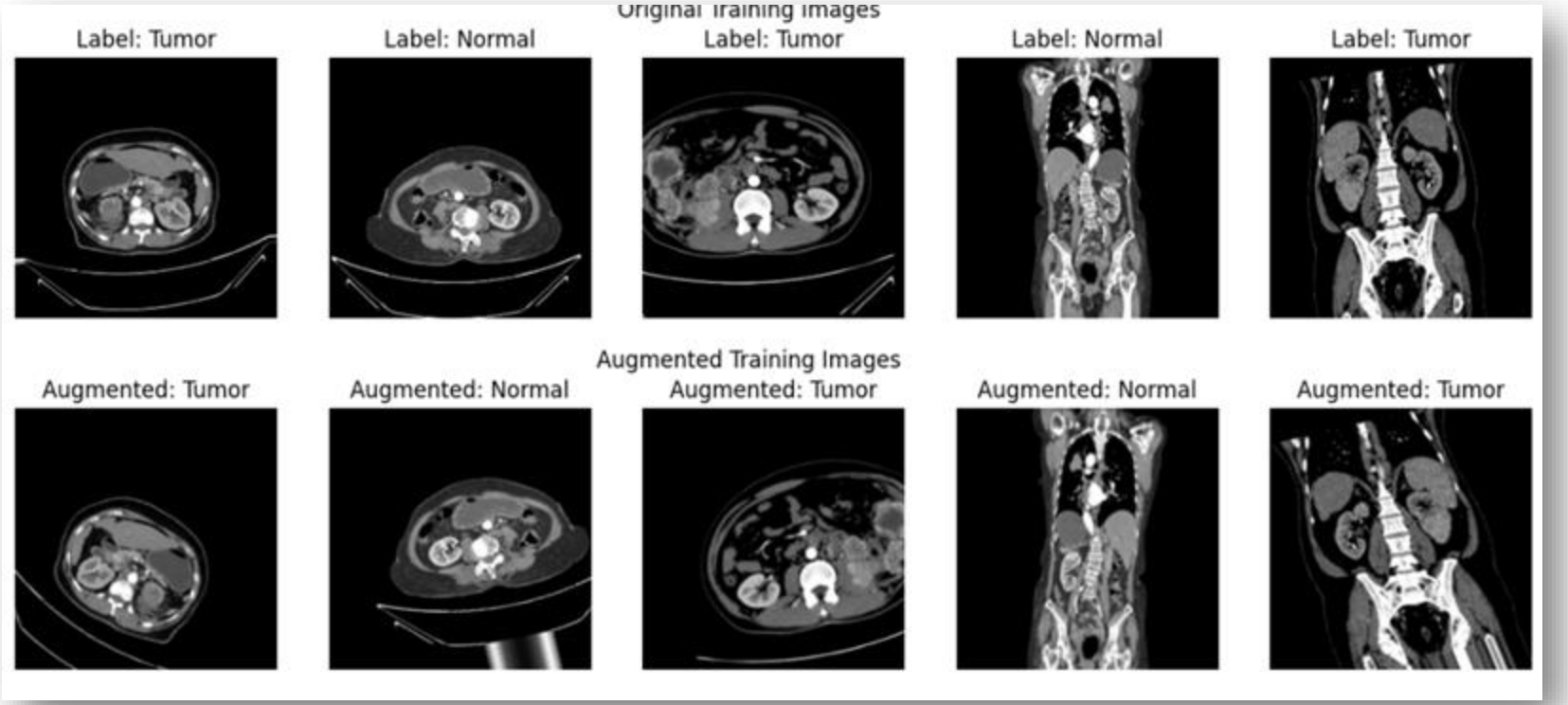


Figures: Results for the *oral cancer* data set

Data augmentation on kidney cancer data

- Lower performance, but more realistic model behaviour
- AUC = 0.92 & accuracy = 0.7



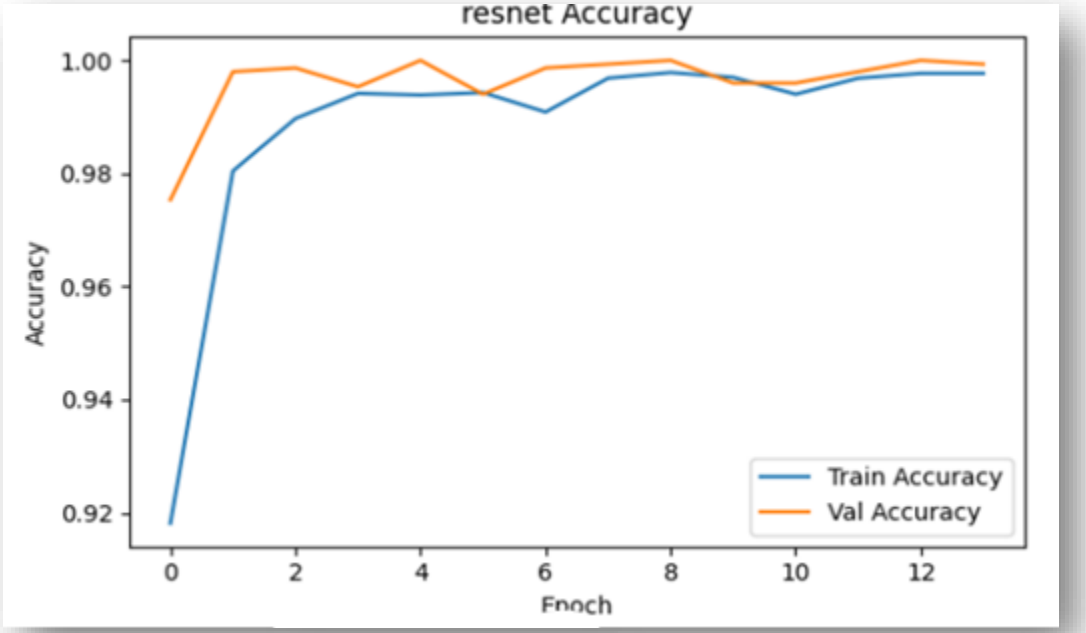


Figures: Results for the *kidney cancer* data set

Foundation model ”ResNet50”

Results for *kidney cancer*:

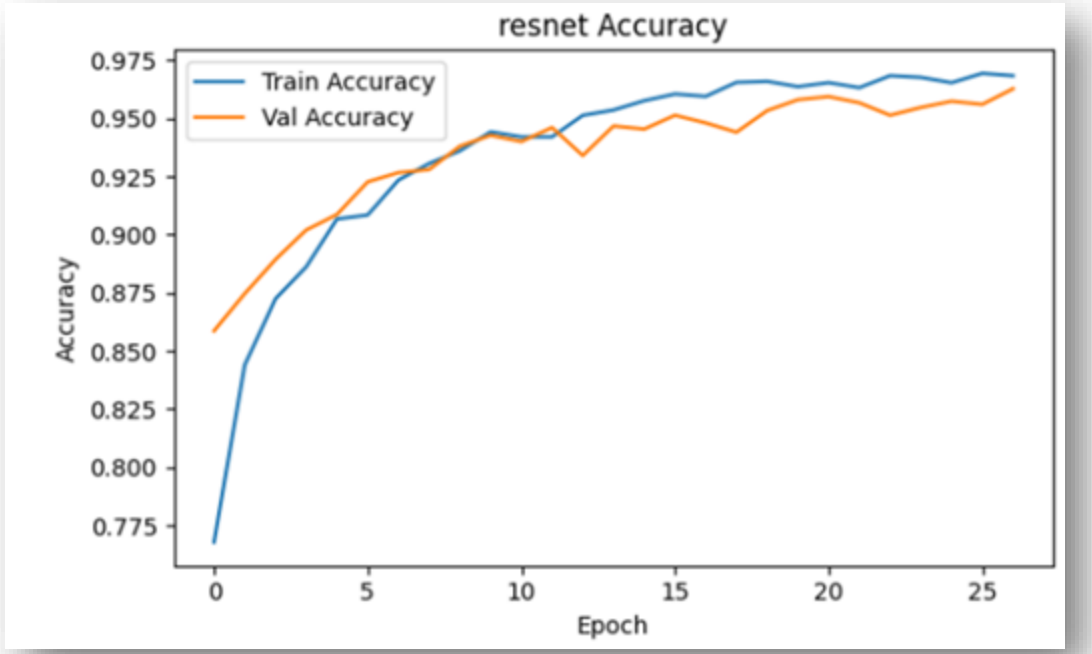
- AUC= 1 & accuracy 1



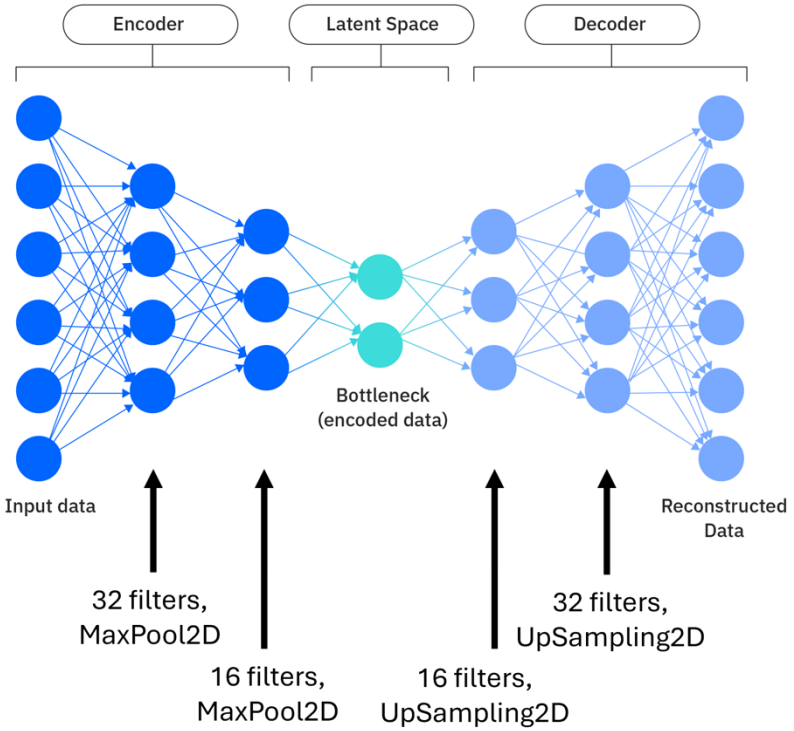
Longer training time ~79 min

Results for *oral cancer*:

- AUC 0.99 & accuracy 0.95



Auto-encoding the images



- Trained on 2000 randomly chosen images out of 10000
- Original image: **(224, 224, 3)**
  → latent space: **(56, 56, 16)**

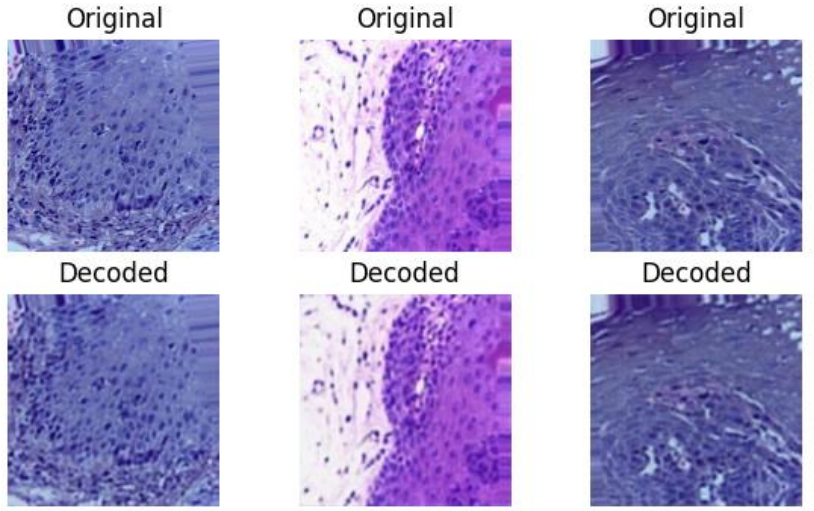
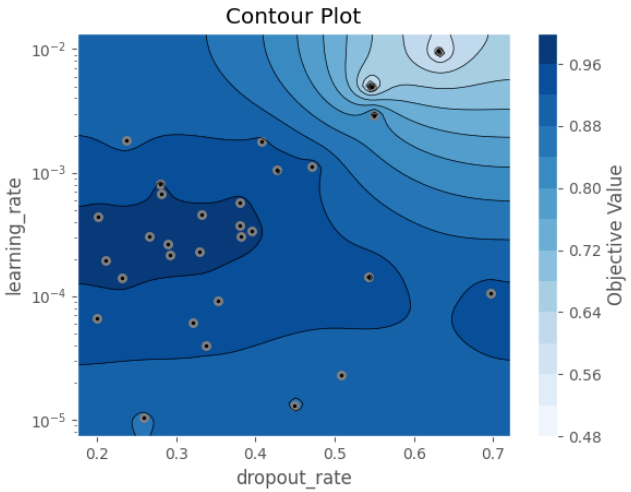


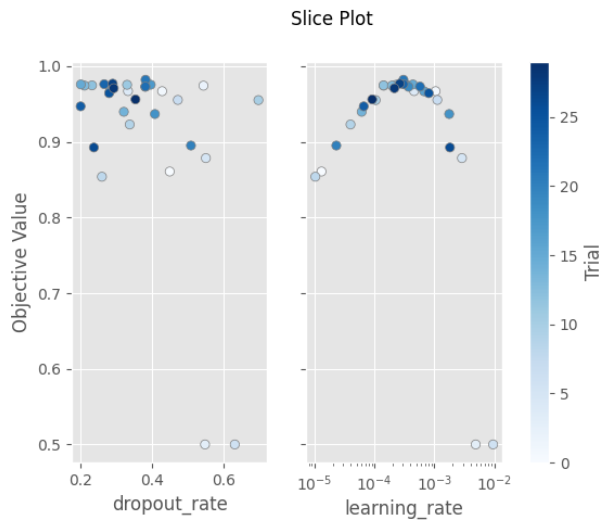
Figure: Original and decoded images for the *oral cancer* data set

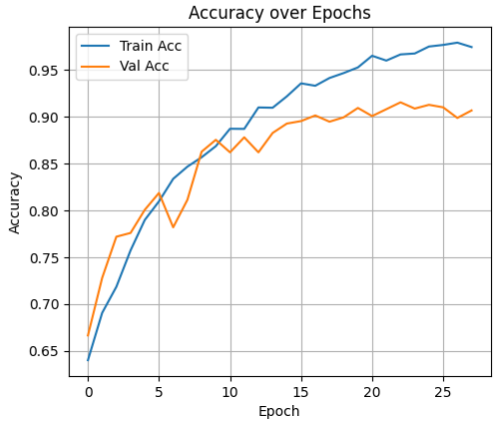
	Val accuracy	Val loss	AUC	Epochs	Training time
Oral	0.86	0.30	0.95	11	102.2 s
Breast	0.96	0.11	0.99	16	141.1 s
Kidney	0.98	0.20	0.99	13	170.4 s

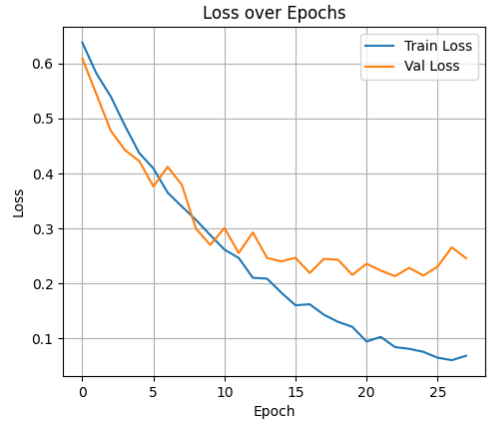
Hyperparameter optimization on encoded oral cancer data

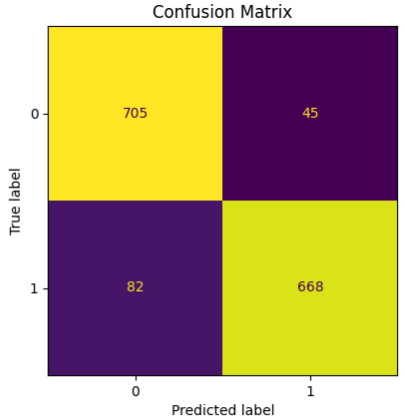


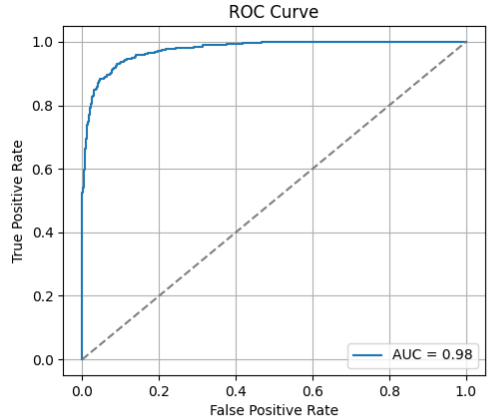
Learning rate = 0.0003
Dropout rate = 0.38











Training time: 180.96 seconds
Best epoch: 23

Feature importance on breast cancer data

Grad-Cam

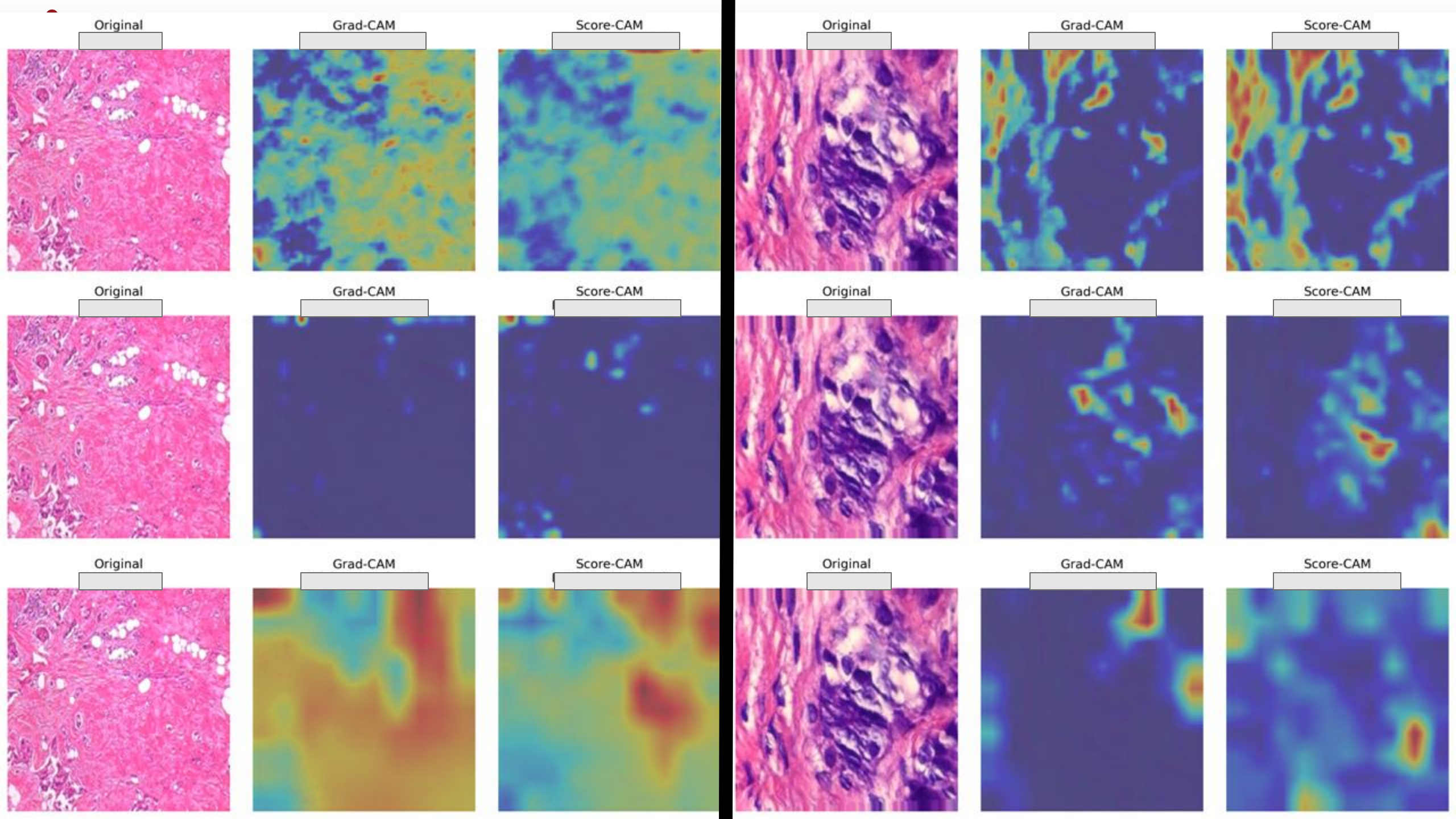
- Calculates gradient of each activation map and compares to the output score. The gradient for each activation map is the weight for a final activation map.

Score-Cam

- Use each activation map and mask on the image to feed to the model for a predicted score. Use scores as weights for each activation map to compute on final activation map.

- Three convolutional layers
  - First layer edges and texture, whereas final layer depicts information with impact.

No idea what the cancer looks like, therefore no clue if cancer is marked.
Need to do it in collaboration with pathologist.


Original
Grad-CAM
Score-CAM
Original
Grad-CAM
Score-CAM
Original
Grad-CAM
Score-CAM
Original
Grad-CAM
Score-CAM
Original
Grad-CAM
Score-CAM
Original
Grad-CAM
Score-CAM

Three best models – for us and our computers

Kidney data	Breast data	Oral data
Data augmentation	Auto Encoding	Auto Encoding
Basic CNN	Basic CNN	Bayesian opt.
Evaluation	Evaluation	Basic CNN
		Evaluation
AUC: 0.92 Accuracy: 0.7 Runtime: 507 s	AUC: 0.99 Accuracy: 0.96 Runtime: 141.06 s	AUC: 0.98 Accuracy: 0.92 Runtime: 180.96 s

Conclusions

- We made three successful CNN models for classification
- Visualised important features with Grad-cam and Score-cam
- The optimal model depends on the dataset (Greyscale vs. colored images)
- “Fast computers breed lazy programmers” - in this case our computers were not fast enough therefore we were very aware of runtime and optimization

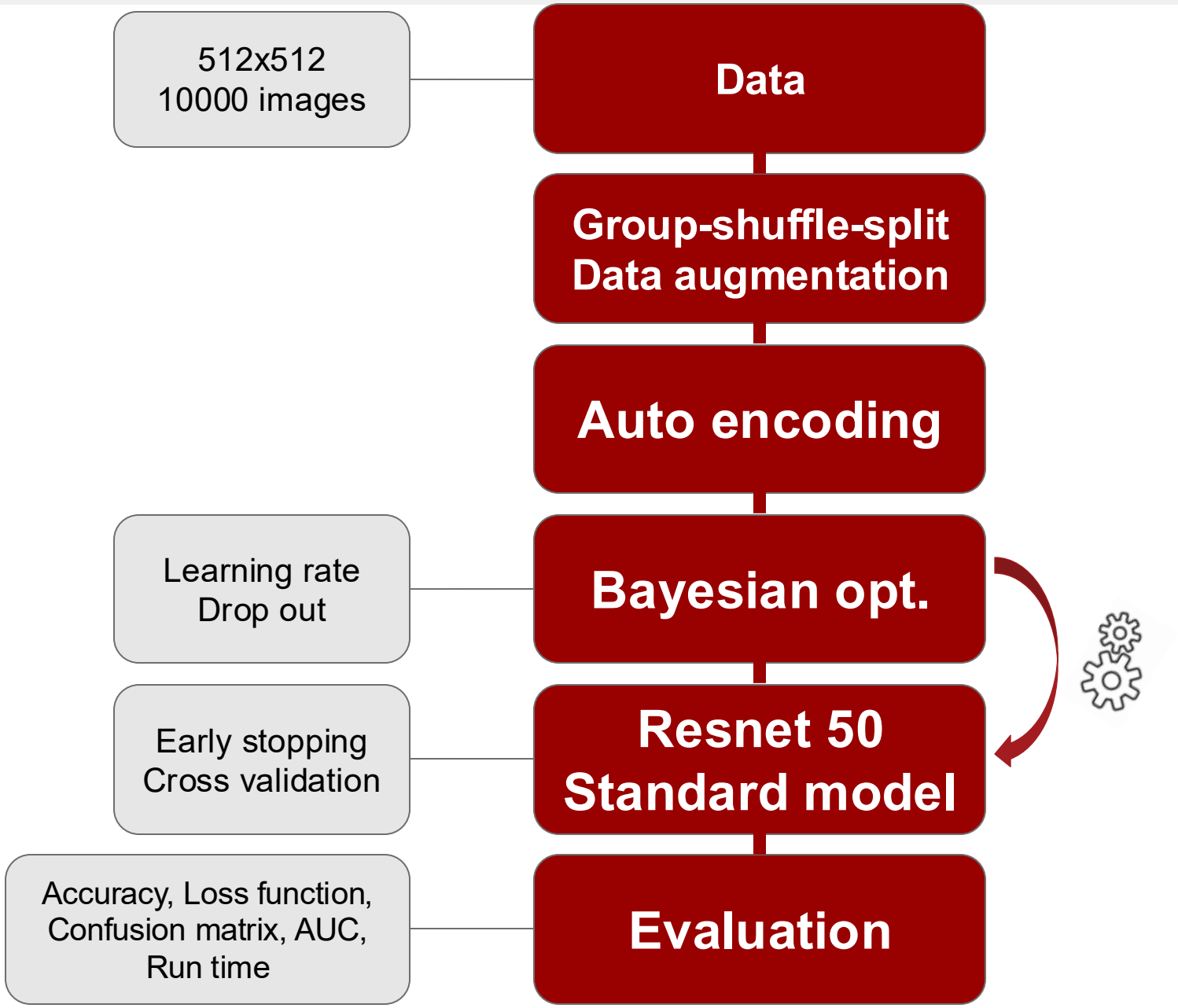
THANK YOU FOR LISTENING!

Scan for access to github with code and a few results

Appendix

Final model

Optimal model in context with infinite time and computer power



Initial CNN

The first thing we did was to try and run a basic CNN model on each of our three datasets. The model was:

Input: Raw images of shape *(224, 224, 3)* (color) or *(224, 224, 1)* (greyscale)

Architecture:

- 3 × [Conv2D + ReLU + MaxPooling2D] blocks with filters (32, 64 and 128)
- Flatten → Dense (128, ReLU) → Dropout
- Output layer: Dense(1, Sigmoid) for *binary classification*

Training:

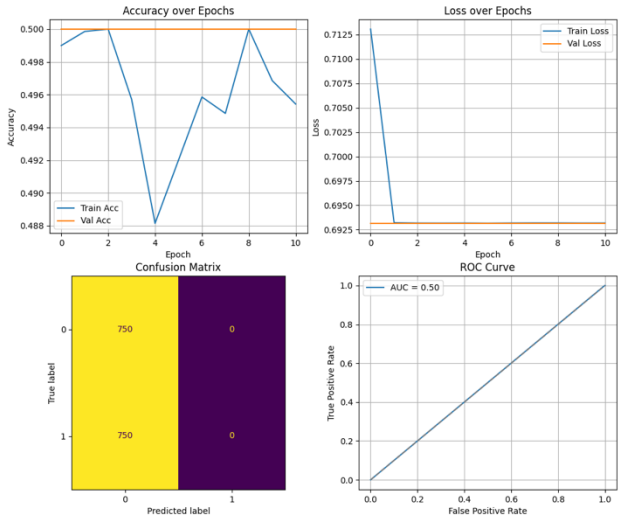
- Optimizer: Adam (learning rate = 0.001)
- Loss: Binary cross-entropy
- Metrics: Accuracy & AUC
- Early stopping on validation loss (patience = 5)
- Batch size: 32
- Max epochs: 30

Performance tracking:

- Validation loss and accuracy curves
- ROC-curve and AUC
- Confusion matrix
- Best epoch selected by lowest val_loss
- Training time recorded

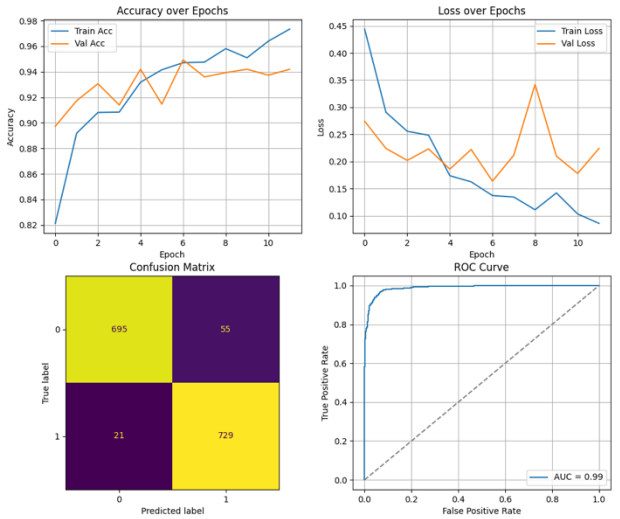
Initial CNN results

Oral



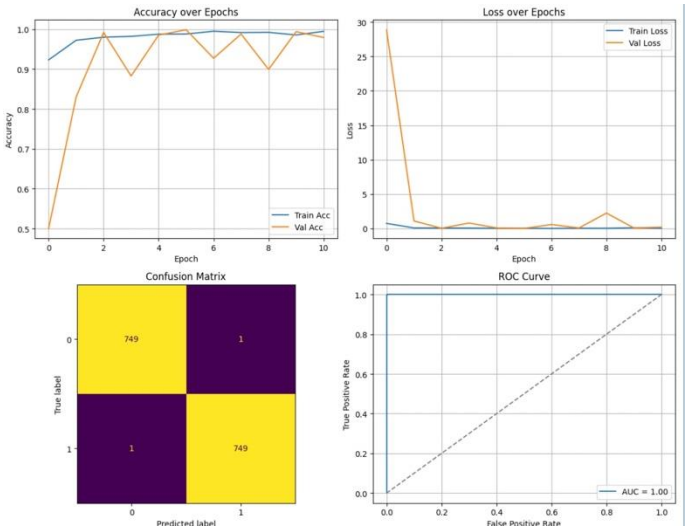
AUC: 0.50
Training time: 1823.80 sec
Best epoch: 6

Breast



AUC: 0.99
Training time: 1968.69 sec
Best epoch: 7

Kidney

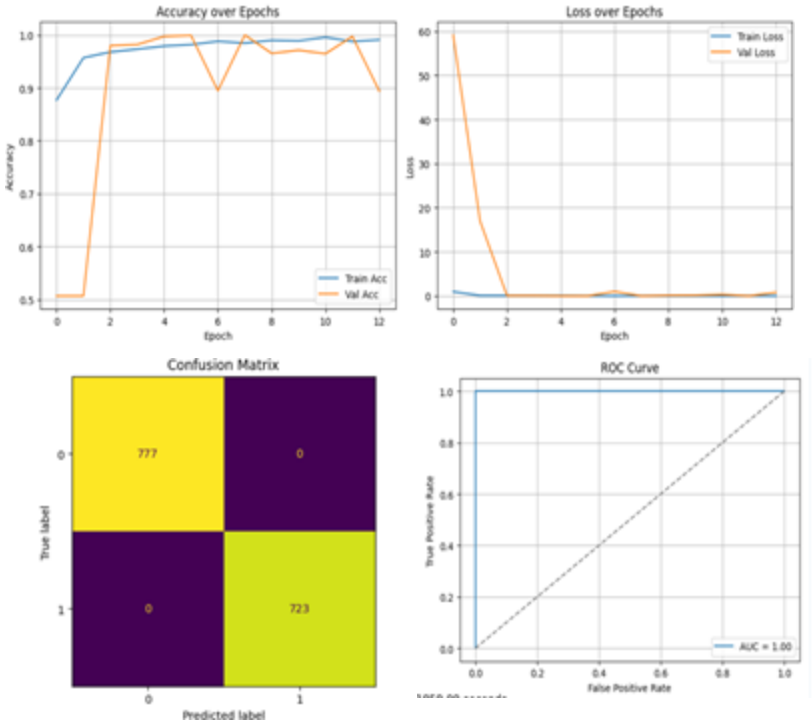


AUC: 1.00
Training time: 1594.03 sec
Best epoch: 6

Group shuffle split

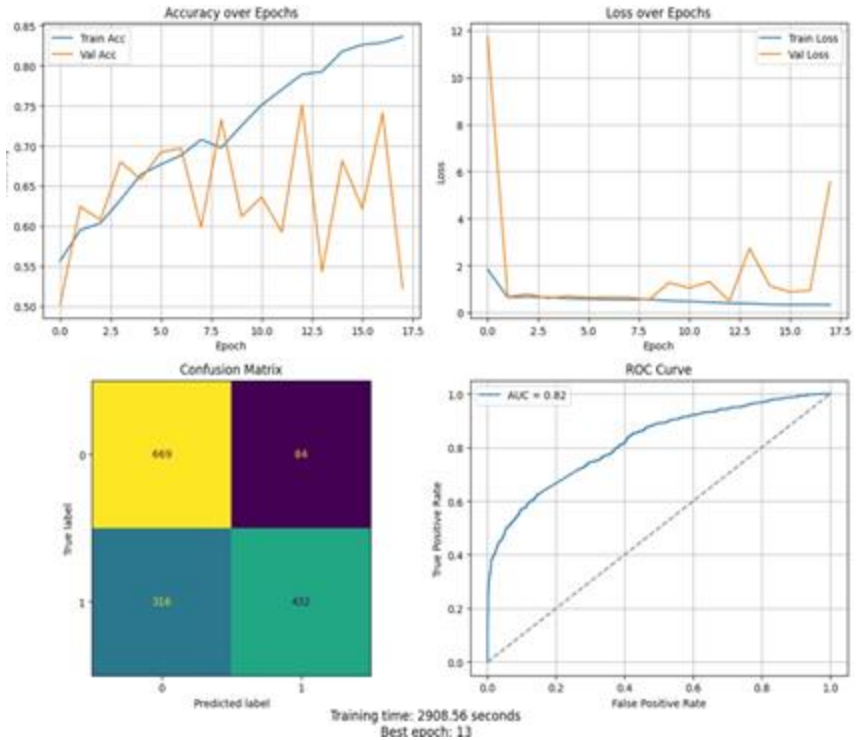
Due to poor initial results on both the *kidney and oral datasets*, we tried implementing **group shuffle split** of our data instead of using the classic test_train_split to avoid possible data leakage. The same CNN model was trained on the newly split data:

Kidney



AUC: 1.00

Oral



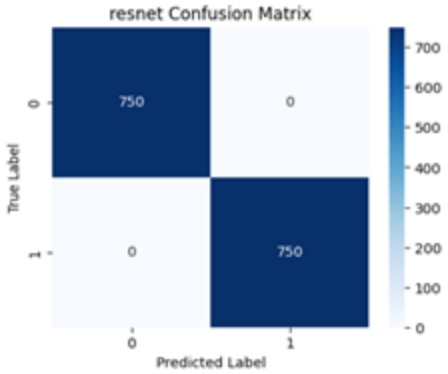
AUC: 0.82

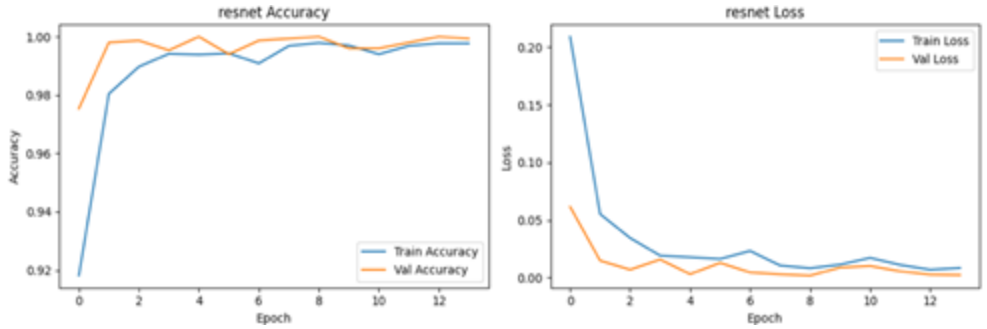
Foundation model - ResNet50

Still, no improvement was observed during training of the CNN model on the *kidney dataset.* Using the original train-test split, we implemented a new model based on the ResNet50 architecture, using pre-trained weights to facilitate learning. The new model was also tried on the *oral dataset.*

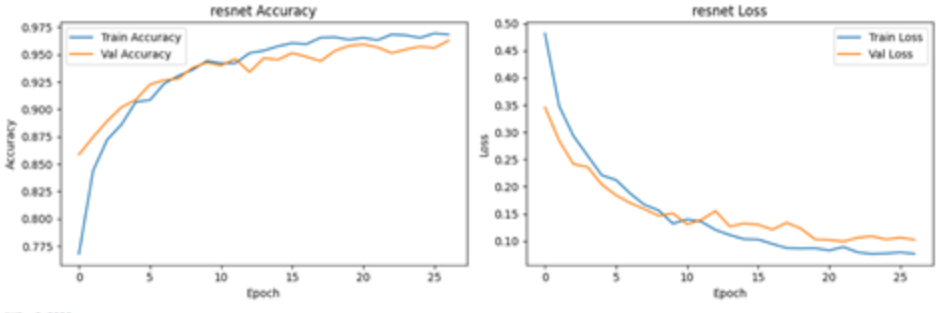
Kidney

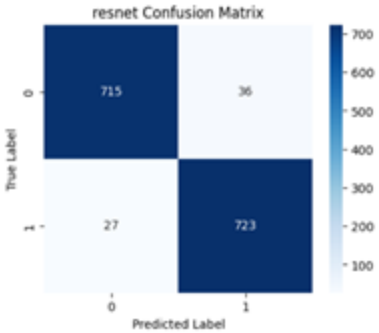
AUC: 1.00





Oral





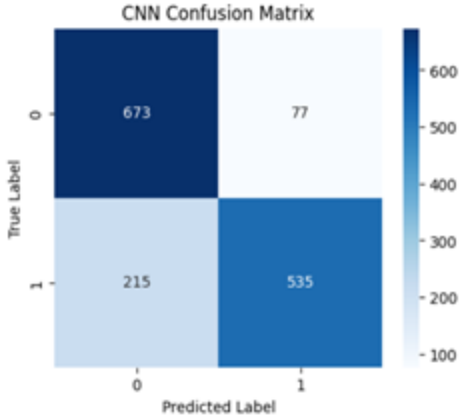
AUC: 0.99
Training time: 5258 s

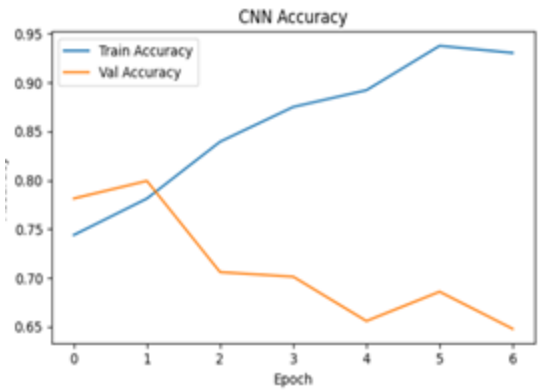
Data augmentation on kidney data

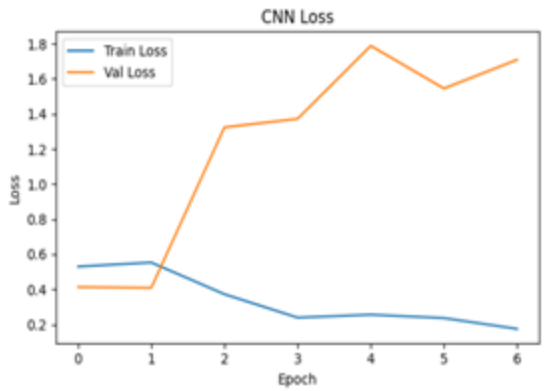
A last thing that was tried to successfully train a model on the kidney data was the implementation of data augmentation.

Kidney – with data augmentation

AUC: 0.92

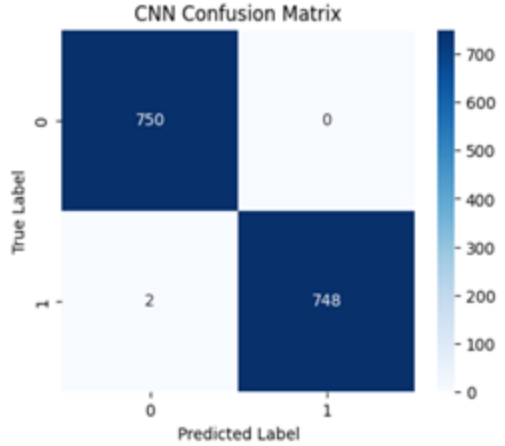


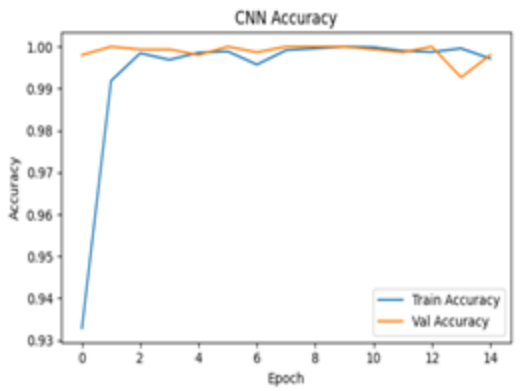


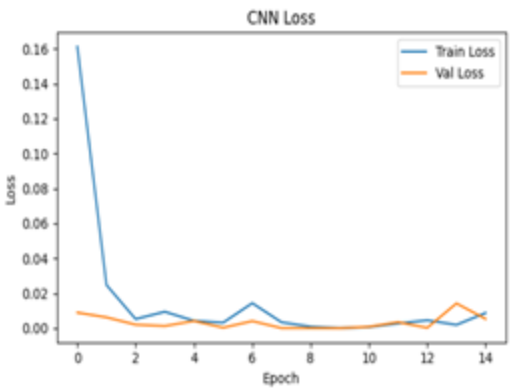


Kidney – without data augmentation

AUC: 1.00







Auto-encoding

Another approach to improve both the performance and training time of the initial CNN models on the three different datasets was to implement an autoencoder that compresses the raw images into a latent space representation. The set-up of the auto-encoder:

Input: Raw images of shape *(224, 224, 3)* (color) or *(224, 224, 1)* (greyscale)

Architecture:

- Encoder: 2 × [Conv2D + ReLU + MaxPooling2D] blocks with 32 and 16 filters
- Decoder: 2 × [Conv2D + ReLU + UpSampling2D] blocks with 16 and 32 filters, followed by 1 × [Conv2D + Sigmoid] block with 3/1 filters for reconstruction

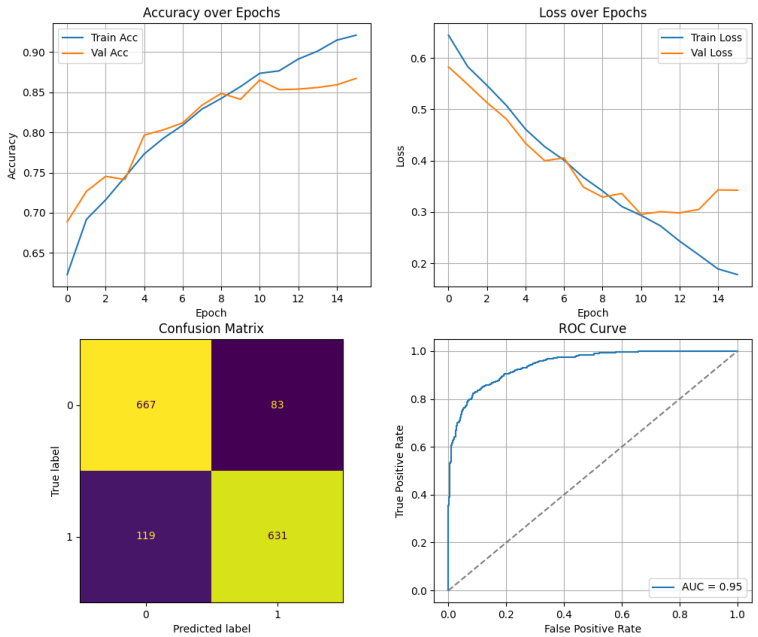
Training:

- Trained on a subset (2000 images) of the dataset
- Optimizer: Adam (learning rate = 0.001)
- Loss: Mean squared error
- Batch size: 16
- Epochs: 15
- Validation split: 0.1

Initial CNN model on the encoded images

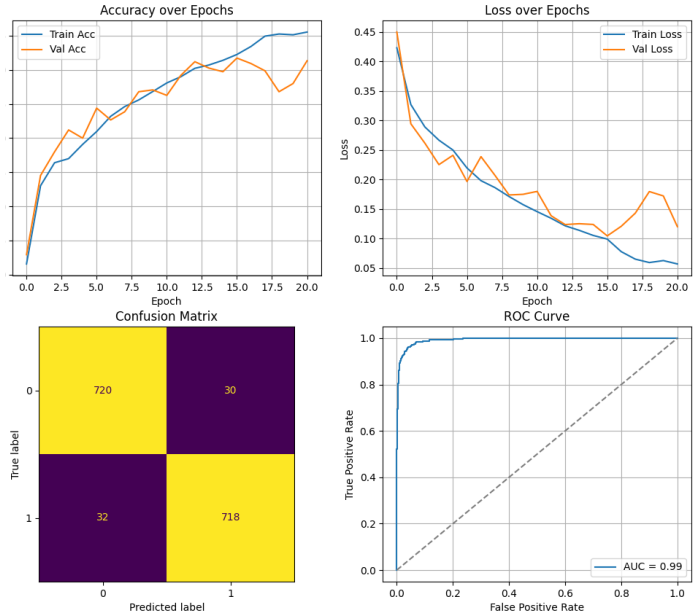
The first basic CNN model was trained again on *all datasets*, but with the encoded images as input instead of the raw. The results were:

Oral



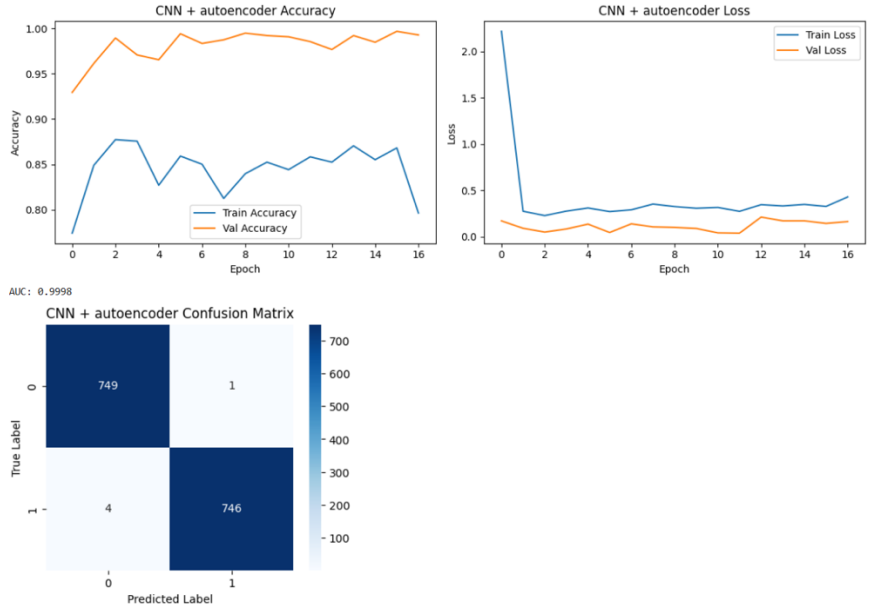
AUC: 0.95
Training time: 102.21 sec
Best epoch: 11

Breast



AUC: 0.99
Training time: 141.06 sec
Best epoch: 16

Kidney

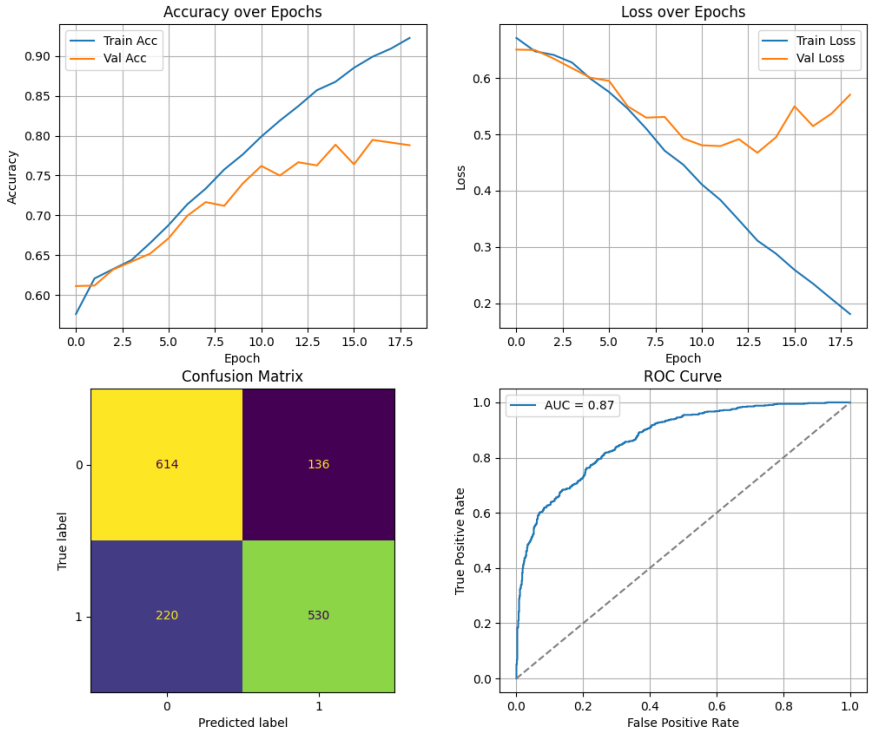


AUC: 0.99
Training time: 170.4 s
Best epoch: 13

Auto-encoder with a dense layer

We then tried to add a dense layer to the auto-encoder to compress the latent space from (56, 56, 16) to (56, 56, 1) in the hope that compressing the features even further might improve the training process and time even more. The new autoencoder was implemented on *the oral dataset* and the basic CNN model was run on the encoded images:

Oral

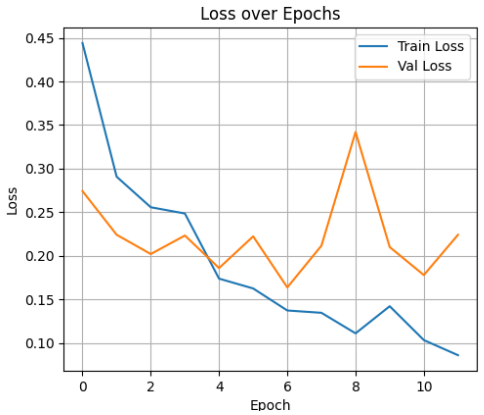


AUC: 0.87
Training time: 173.20 sec
Best epoch: 14

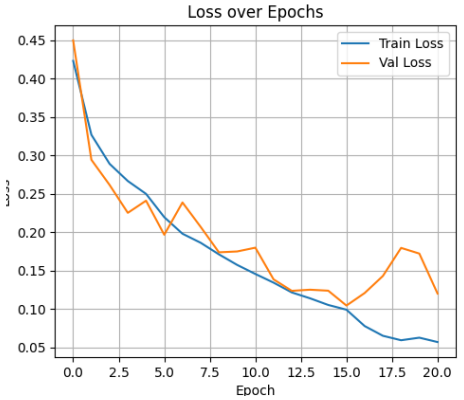
Additional slide that didn't make it into the presentation

- Reduced training time significantly!
- Captured important features to make model on oral cancer data learn
- Healthier training process for breast cancer data.

Raw images

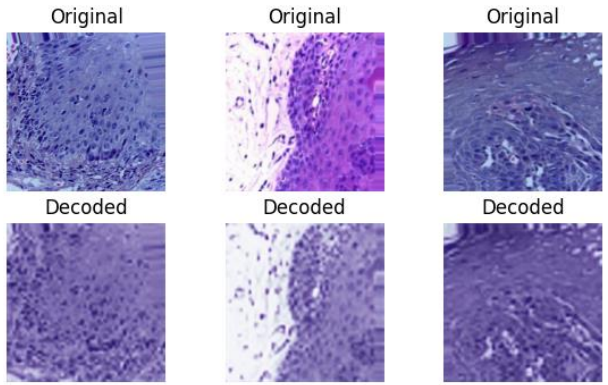


Latent space

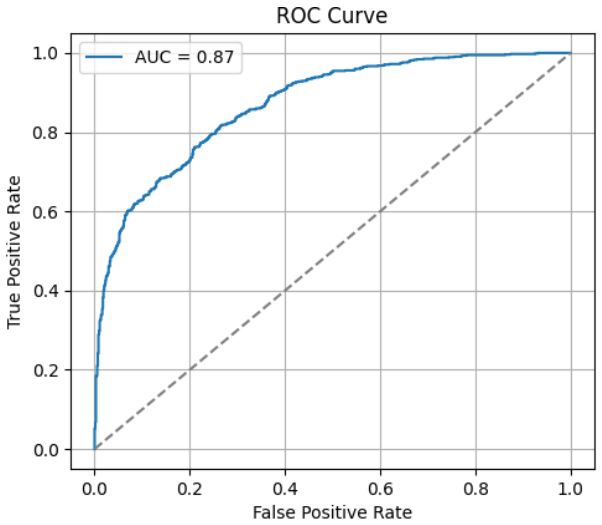


Figures: Loss curves for the *breast cancer* dataset

Introducing a **dense layer** to the auto encoder on *oral cancer* data



Latent space: **(56, 56, 1)**



Hyper parameter optimization

Since a great improvement in the training process and time of the basic CNN model on the *oral dataset* was seen when auto-encoding the images into latent space of (56, 56, 16), we wanted to try and improve the model even further. Therefore, Bayesian optimization was tried for this model:

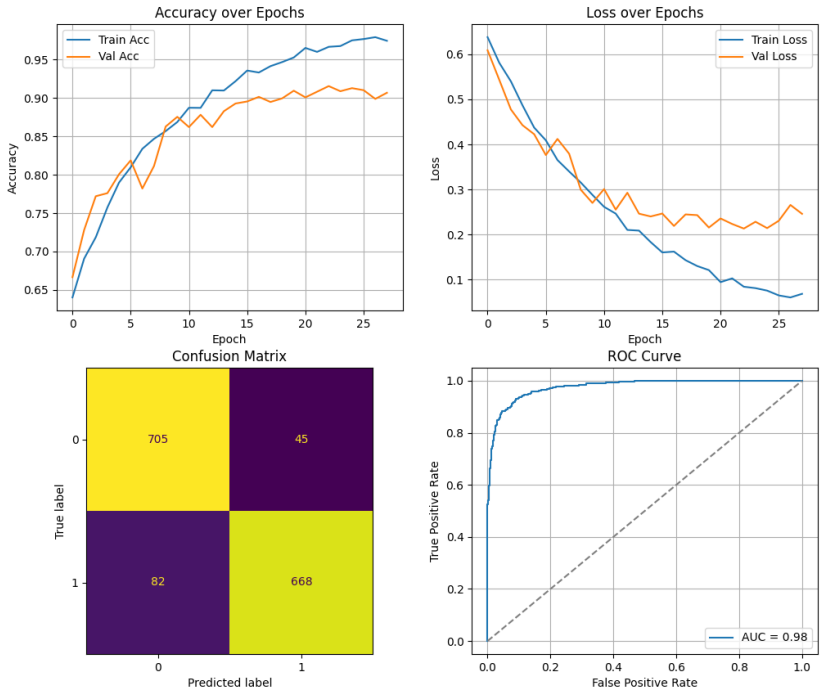
Optuna Hyperparameter Optimization Setup:

- Objective: Maximize validation AUC
- Number of trials: 30
- Hyperparameters tuned:
  - Learning rate in range [1e-5 : 1e-2]
  - Dropout rate in range [0.2 : 0.7]

Optimal HPs:

- Learning rate = 0.0003
- Dropout rate = 0.38

Oral



AUC: 0.98
Training time: 180.96 sec
Best epoch: 23

Grad-Cam and Score-Cam

Grad-CAM and Score-CAM were applied to a CNN model trained on 10,000 autoencoded breast biopsy images. To ensure efficient runtime, the CAMs were computed on the encoded images. These representations were then decoded to approximate the original images, allowing the visual explanations to be overlaid the original image space.

Article for Score-Cam and Grad-Cam with comparison between different packages:

https://arxiv.org/pdf/1910.01279