RadiologyTextGen: A Multi-Modal model for generating text reports for a given chest X-ray image

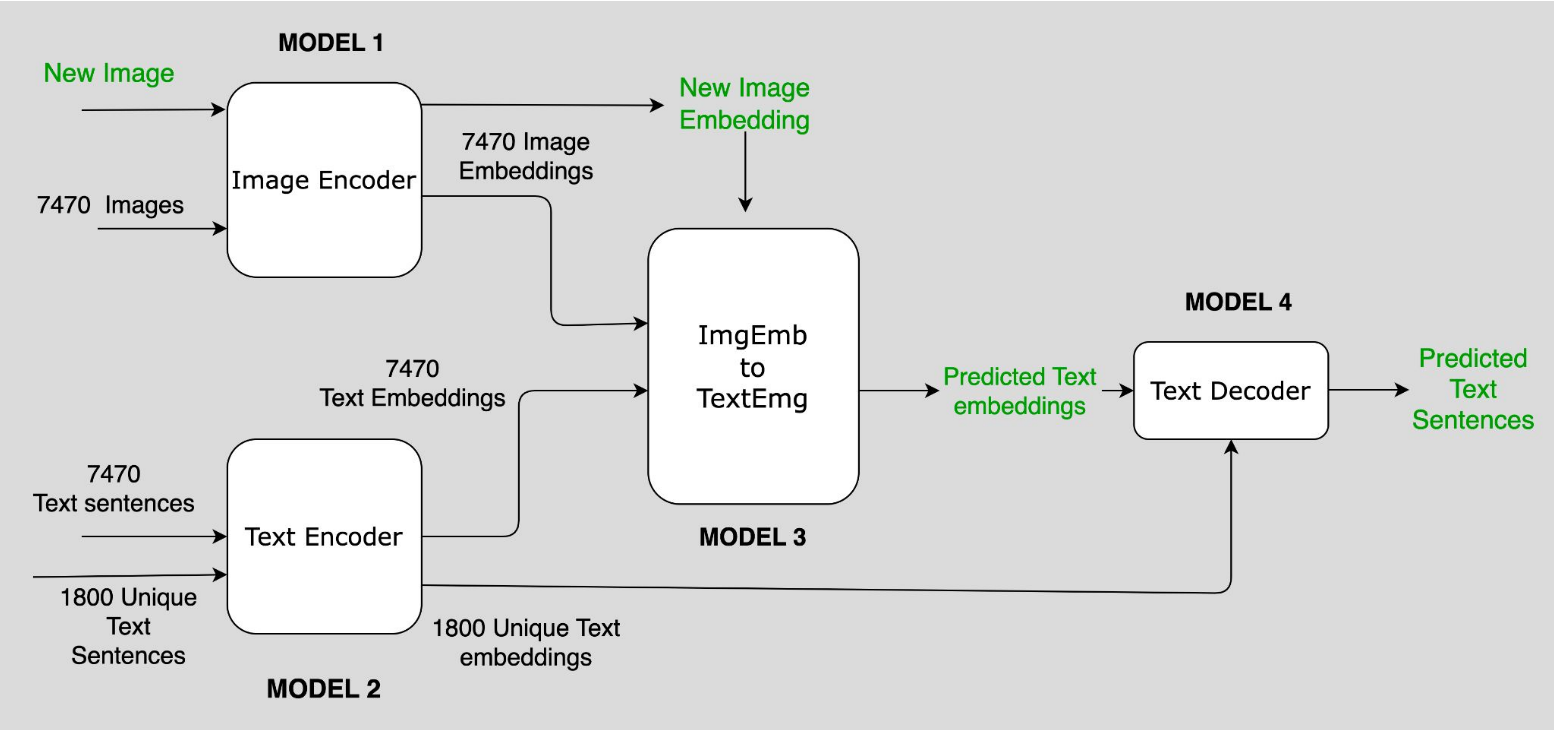
Abdul Moid Mohammed, Ayesha Sajid Ali, Chris Frary, Crystal Guzmán

Objective

Assist healthcare providers in thorax diagnoses.

Utilize computer vision and natural language processing techniques to generate an impression from chest x-rays images.

Project Architecture



Data Description

Data: https://openi.nlm.nih.gov/faq#collection

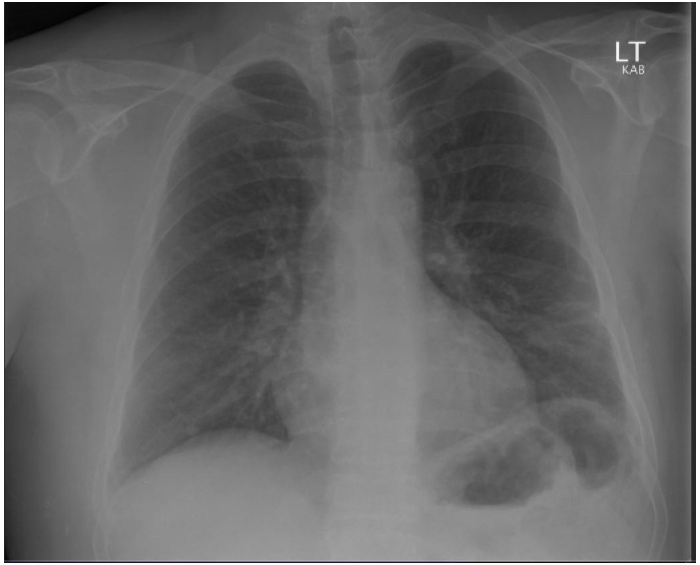
Our dataset contains 3996 radiology reports and 8121 associated X ray images.

The chest x-ray images and narrative radiology reports are sourced from the Indiana University hospital network.

The data are de-identified to maintain patients privacy.

Images are in png and reports are in XML format.

Example Data



<AbstractText Label="COMPARISON"/>

<AbstractText Label="INDICATION">XXXX-year-old male with dyspnea</AbstractText>

<AbstractText Label="FINDINGS">No pneumothorax. No large pleural effusions. Heart size is normal. No acute focal space opacities.</AbstractText>

<AbstractText Label="IMPRESSION">No acute cardiopulmonary abnormalities.</AbstractText>

Data Preprocessing

Extracted the IMPRESSIONS text from its XML Tag.

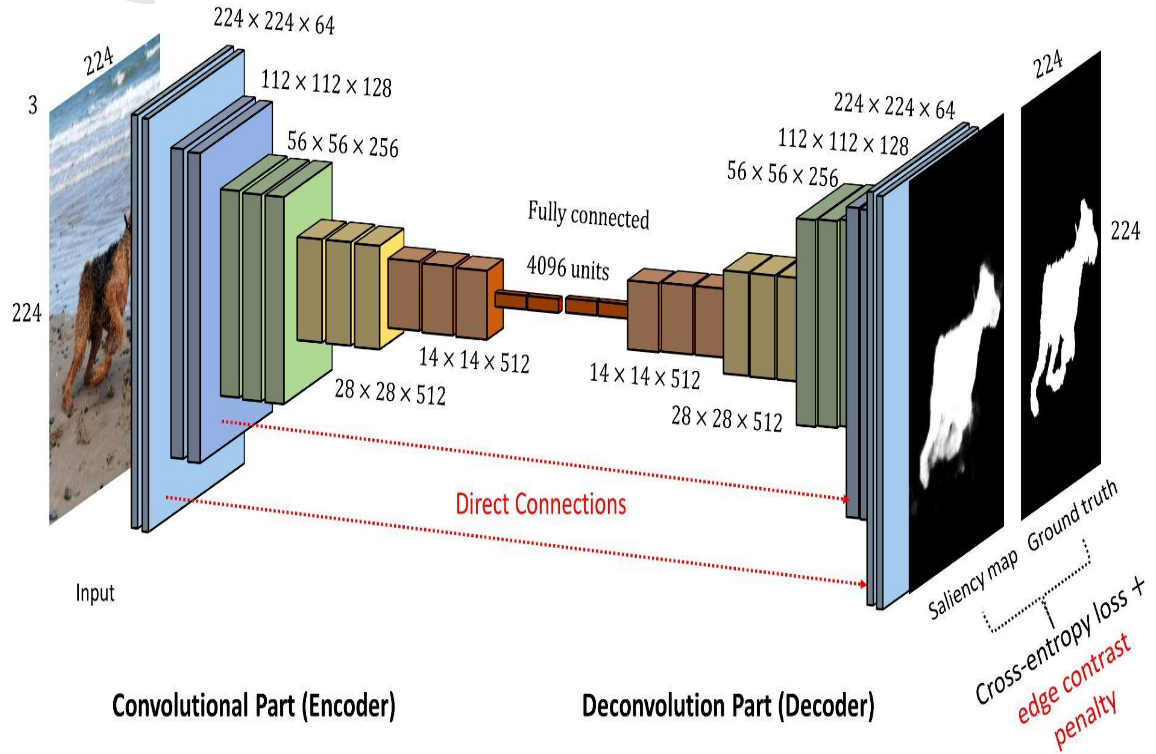
Removed reports not having its corresponding Chest X Ray images.

In our dataset, there are multiple Chest X ray images associated with one report. So, we duplicated reports to have consistent image and text pairing.

7470 reports <-> 7470 images.

Extracted unique sentences from the IMPRESSIONS Tag. (1800)

MODEL 1: Image Encoder: VGG-based Autoencoder



Goal: Generate Image Embeddings

1. Train Image **VGG 19 Encoder** by extracting feature vectors from unlabeled x-ray images.
2. Image **VGG 19 Decoder** then generated images using VGG 19 Encoder feature vectors.
3. Original images vs. generated images produced a **LOSS function value** which was used in backpropagation (adjusting weights) and allowed us to further train the VGG 19 Encoder and produce adequate Image Embeddings.

MODEL 1: 3 Variations

Goal: attain model variation with lowest loss value. 3 variations:

1. Complete autoencoder architecture (utilized 5 blocks or convolution layers that applied filters to extract features)

2. Stopped encoder at block 4 **allowed layer pooling** to reduce dimension to 14x14x512 for decoder use

3. Stopped encoder at block 4 & **halted before pooling** compression resulting in 28x28x512 size for decoder use

```
Layer (type)                Output Shape            Param #
=================================================================
input_3 (InputLayer)        [(None, 224, 224, 3)]   0

model (Functional)          (None, 4096)            139570240

model_1 (Functional)        (None, 224, 224, 3)     104357795

=================================================================
Total params: 243928035 (930.51 MB)
Trainable params: 243928035 (930.51 MB)
Non-trainable params: 0 (0.00 Byte)
```

Autoencoder with encoder output at fc2

training loss: 0.0027

validation loss: 0.0043

test loss : 0.0042

```
Layer (type)                Output Shape            Param #
=================================================================
input_3 (InputLayer)        [(None, 224, 224, 3)]   0

model (Functional)          (None, 14, 14, 512)     10585152

model_1 (Functional)        (None, 224, 224, 3)     1568067

=================================================================
Total params: 12153219 (46.36 MB)
Trainable params: 12153219 (46.36 MB)
Non-trainable params: 0 (0.00 Byte)
```

Autoencoder with encoder output at block4_pool

training loss: 4.6728e-04

validation loss: 7.5245e-04

test loss : 7.4689e-04

```
Layer (type)                Output Shape            Param #
=================================================================
input_3 (InputLayer)        [(None, 224, 224, 3)]   0

model (Functional)          (None, 28, 28, 512)     10585152

model_1 (Functional)        (None, 224, 224, 3)     1550467

=================================================================
Total params: 12135619 (46.29 MB)
Trainable params: 12135619 (46.29 MB)
Non-trainable params: 0 (0.00 Byte)
```

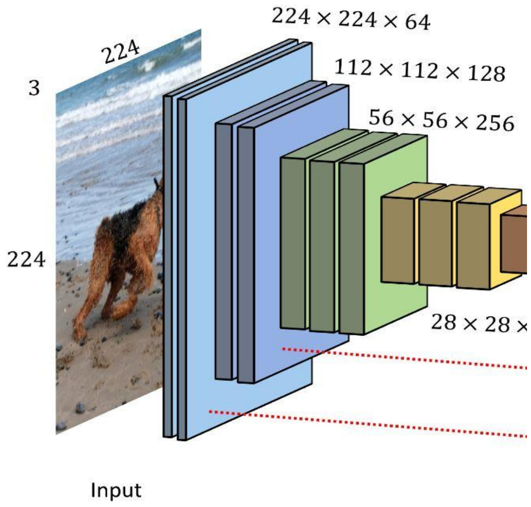
Autoencoder with encoder output at block4_conv4

training loss: 2.9424e-04

validation loss: 4.8640e-04

test loss: 4.7700e-04

MODEL 1: Image Encoder



Convolutional Part (Encoder)

Image encoder trained, we then passed all 7470 x-ray images thru the encoder to get an image embedding (feature vector) for each image to be used in the fusion model (model 3).

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 224, 224, 3)]	0
model (Functional)	(None, 28, 28, 512)	10585152
model_1 (Functional)	(None, 224, 224, 3)	1550467

Total params: 12135619 (46.29 MB)
Trainable params: 12135619 (46.29 MB)
Non-trainable params: 0 (0.00 Byte)

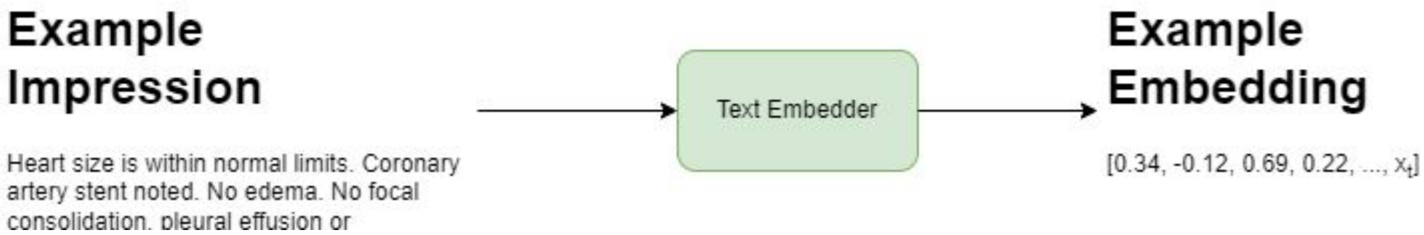
MODEL 2: Text Embedding

Transform the sentences in IMPRESSIONS to the embedding space

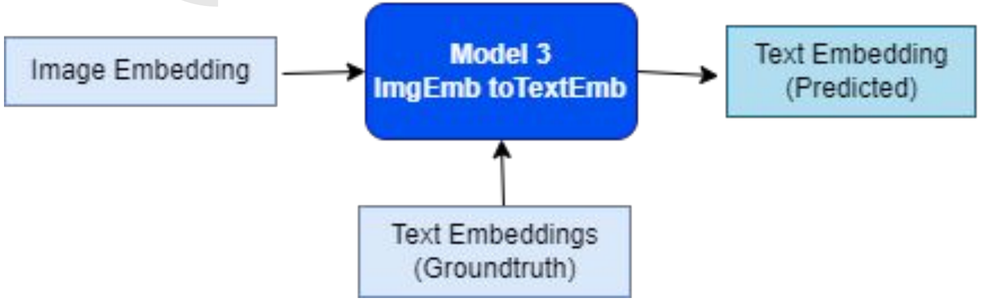
ClinicalBERT model trained on 2M notes from MIMIC medical database

ClinicalBERT tested 80.8% accuracy on MedNLI natural language inference task (Romanov and Shivade, 2018), 3% improvement over general BERT

768 dimension embedded sentence vector



MODEL 3: Image_Embedding to Text Embedding



Hyperparameters:

Batch_Size = 16

Epochs = 600

Validation_split = 0.2

Loss: Mean_Absolute_Error

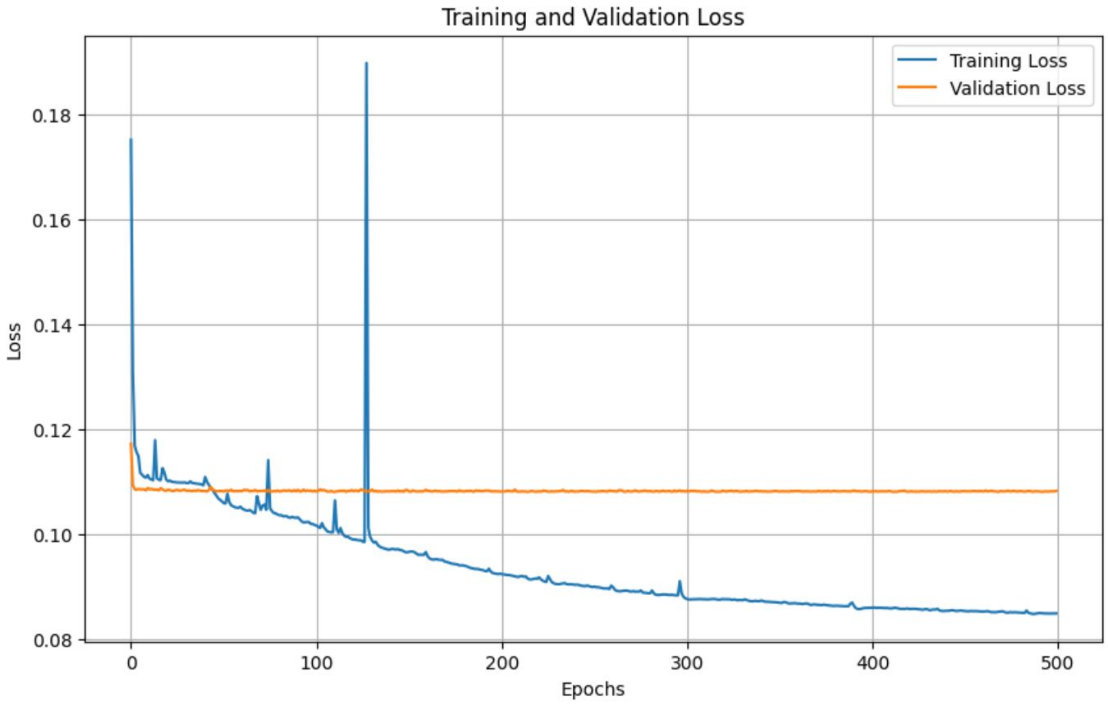
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 512)]     0

 conv2d (Conv2D)             (None, 28, 28, 512)       2359808

 max_pooling2d (MaxPooling2  (None, 14, 14, 512)       0
 D)

 conv2d_1 (Conv2D)           (None, 14, 14, 256)       1179904

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 256)         0
 g2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 128)         295040

 max_pooling2d_2 (MaxPoolin  (None, 3, 3, 128)         0
 g2D)

 conv2d_3 (Conv2D)           (None, 3, 3, 64)          73792

 max_pooling2d_3 (MaxPoolin  (None, 1, 1, 64)          0
 g2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 256)               16640

 dense_1 (Dense)             (None, 512)               131584

 dense_2 (Dense)             (None, 768)               393984

=================================================================
Total params: 4450752 (16.98 MB)
Trainable params: 4450752 (16.98 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Model 3 Results

Results:

Training Loss: 0.0835

Validation Loss: 0.1081



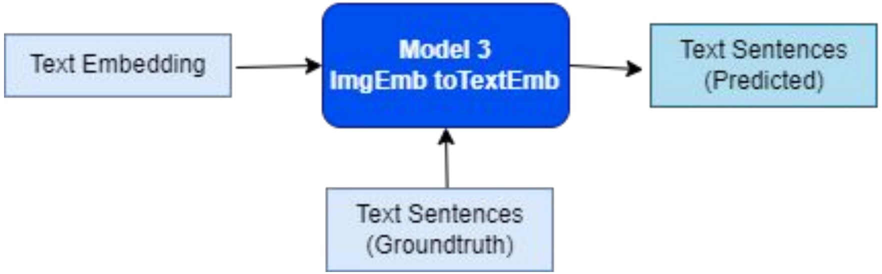
MODEL 4: Text_Embedding to Text Sentence

Our text data had 7470 sentences (including duplicates), but we chose to train on unique sentences (1771 sentences)

Why unique sentences?

Duplicates in data can have several implications

- Overfitting
- Inefficient training
- Impact on model's training

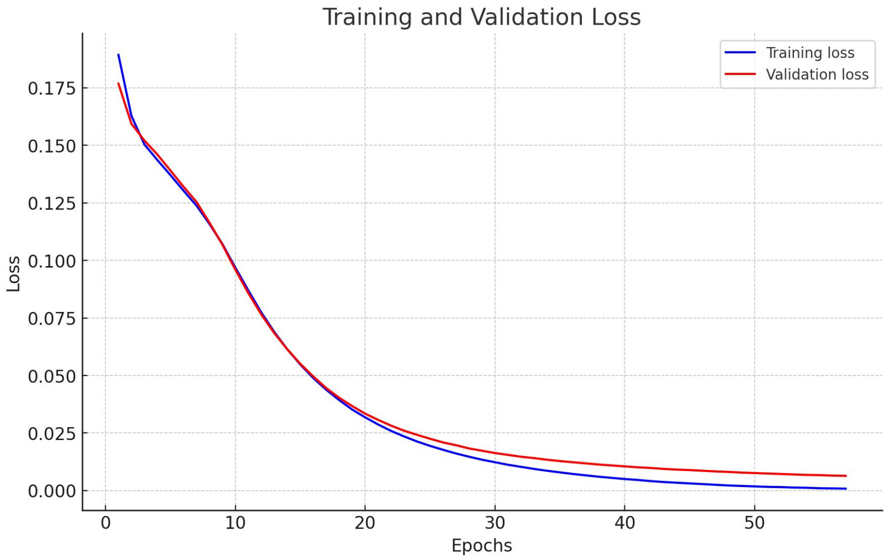


Hyper-parameters:

- Batch Size: 16
- Epochs: 500
- Validation Split: 0.2
- Loss: sparse_categorical_crossentropy

MODEL 4: Text_Embedding to Text Sentence

```
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 768)]                0         []

 input_2 (InputLayer)        [(None, None)]               0         []

 reshape (Reshape)           (None, 768, 1)               0         ['input_1[0][0]']

 embedding (Embedding)       (None, None, 256)            1024256   ['input_2[0][0]']

 lstm (LSTM)                 [(None, 256),                264192    ['reshape[0][0]']
                              (None, 256),
                              (None, 256)]

 lstm_1 (LSTM)               [(None, None, 256),          525312    ['embedding[0][0]',
                              (None, 256),                           'lstm[0][1]',
                              (None, 256)]                           'lstm[0][2]']

 dropout (Dropout)           (None, None, 256)            0         ['lstm_1[0][0]']

 dense (Dense)               (None, None, 4001)           1028257   ['dropout[0][0]']

==================================================================================================
Total params: 2842017 (10.84 MB)
Trainable params: 2842017 (10.84 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

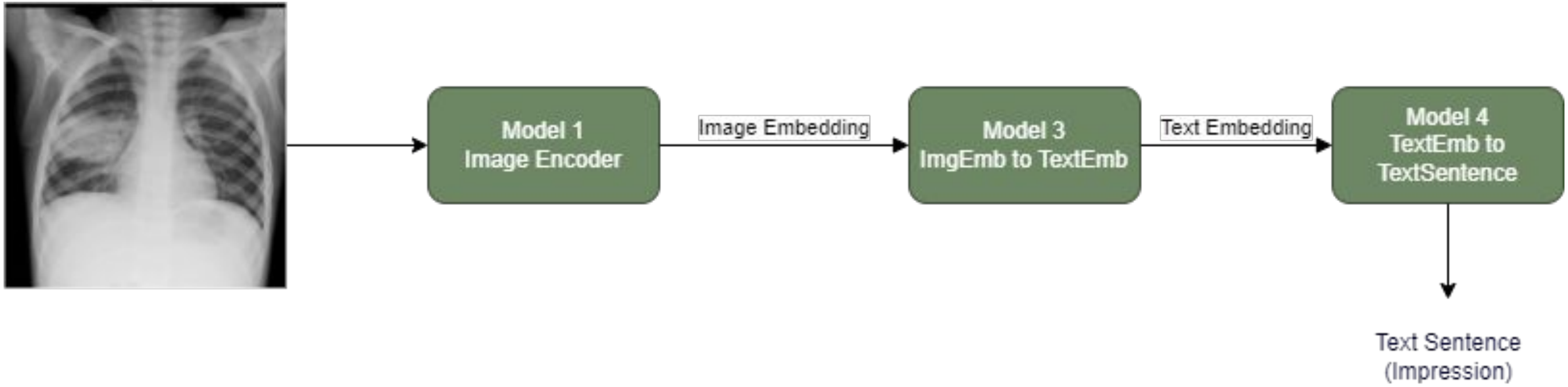


Results:

Training Loss: 0.0009

Validation Loss: 0.0064

Image Inference




```
# Generate text using the model
generated_text = generate_text(model4, new_text_embedding, tokenizer)
print(generated_text)
```

```
nonenlarged total quality abdominal extensive thin suggestive probable a abdomen identified given would indeterminate this left lung impression arteriosum radiography hydropn
eumothorax admission different hypoinflated airways lungs
```

Conclusion & Limitations

Although we were able generate sentence from Chest Xray image, it’s not accurate.

The biggest limitation is lack of varied data

Model 3 (Image to Text Embedding) could be improved by further modifying the model architecture

Future Work

Model 3 (Image Embedding to Text Embedding) Improvement:

- Use augmented images
- Augment text sentences

Collect more training data (images and reports), possibly source from hospital.

Enhance the model performances by refining their architectures.

Include other tags like INDICATION, FINDINGS

References

Data source: https://openi.nlm.nih.gov/faq#collection

Web Article: https://vaclavkosar.com/ml/Multimodal-Image-Text-Classification

Other References:

ClinicalBERT:

https://medium.com/nwamaka-imasogie/clinicalbert-using-deep-learning-transformer-model-to-predict-hospital-readmission-c82ff0e4bb03

QUESTIONS?