Q. Comment on your whole network design and explain each component that you employ.

Data bias padding

The training dataset originally has 1082 normal images and 3110 pneumonia images. This led to the model leaning strongly towards guessing pneumonia. To counter this, we padded the training dataset by manually tripling the normal images. This resulted in a training dataset with 3246 normal images and 3110 pneumonia images.

Data augmentation

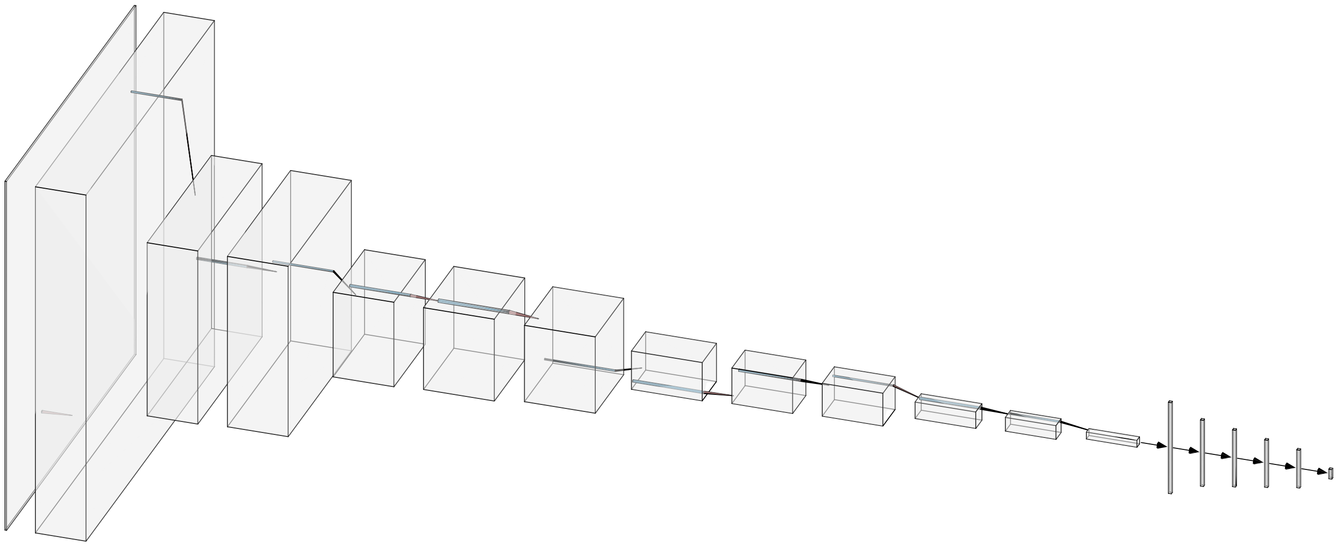
We employed data augmentation to increase the size of the training dataset. This is done by applying random translations, rotations, and scalings to the images.

```
training_data_generator = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range = 40,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    zoom_range = 0.2,
)
```

Model

We employed a deep convolutional neural network (CNN). The overall shape of the network is a funnel shape, with the input being a greyscale `256x256x1` image and the output being `2` nodes. To combat overfitting and encourage generalisation, we added two dropout layers.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      320

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 conv2d_3 (Conv2D)           (None, 56, 56, 128)       409728

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
 2D)

 conv2d_4 (Conv2D)           (None, 26, 26, 64)        73792

 conv2d_5 (Conv2D)           (None, 24, 24, 64)        36928

 dropout (Dropout)           (None, 24, 24, 64)        0

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 10, 10, 32)        18464

 conv2d_7 (Conv2D)           (None, 8, 8, 32)          9248

 max_pooling2d_4 (MaxPooling  (None, 4, 4, 32)         0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 16)                528

 dense_4 (Dense)             (None, 2)                 34

=================================================================
Total params: 717,394
Trainable params: 717,394
Non-trainable params: 0
_________________________________________________________________
```



> We used `relu` as our activation function for all layers except the output layer. For the output layer, we used `sigmoid`.
>
> ```
> model = Sequential(
>     [
>         layers.Conv2D(32, 3, input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, 1), activation="relu"),
>         layers.MaxPooling2D(),
>         layers.Conv2D(64, 3, activation="relu"),
>         layers.MaxPooling2D(),
>         layers.Conv2D(128, 3, activation="relu"),
>         layers.Conv2D(128, 5, activation="relu"),
>         layers.MaxPooling2D(),
>         layers.Conv2D(64, 3, activation="relu"),
>         layers.Conv2D(64, 3, activation="relu"),
>         layers.Dropout(0.2),
>         layers.MaxPooling2D(),
>         layers.Conv2D(32, 3, activation="relu"),
>         layers.Conv2D(32, 3, activation="relu"),
>         layers.MaxPooling2D(),
>         layers.Flatten(),
>         layers.Dense(128, activation="relu"),
>         layers.Dense(64, activation="relu"),
>         layers.Dropout(0.2),
>         layers.Dense(32, activation="relu"),
>         layers.Dense(16, activation="relu"),
>         layers.Dense(NUM_CLASSES, activation="sigmoid"),
>     ]
> )
> ```

For the training process, we used an NVIDIA GeForce RTX 3070 Ti. The training process took 1 hour and 43 minutes to complete.

Hyperparameters

We used the following hyperparameters:

- Input shape: `256x256x1`
- Batch size: `32`
- Dropout rate: `0.2`
- Optimiser: `Adam`
    - Learning rate: `0.001`
    - Loss function: `Categorical cross-entropy`

All unmentioned hyperparameters are left at their default values.

Early stopping

To prevent overfitting, we used early stopping. This stops the training process when the validation loss stops decreasing for 20 epochs.

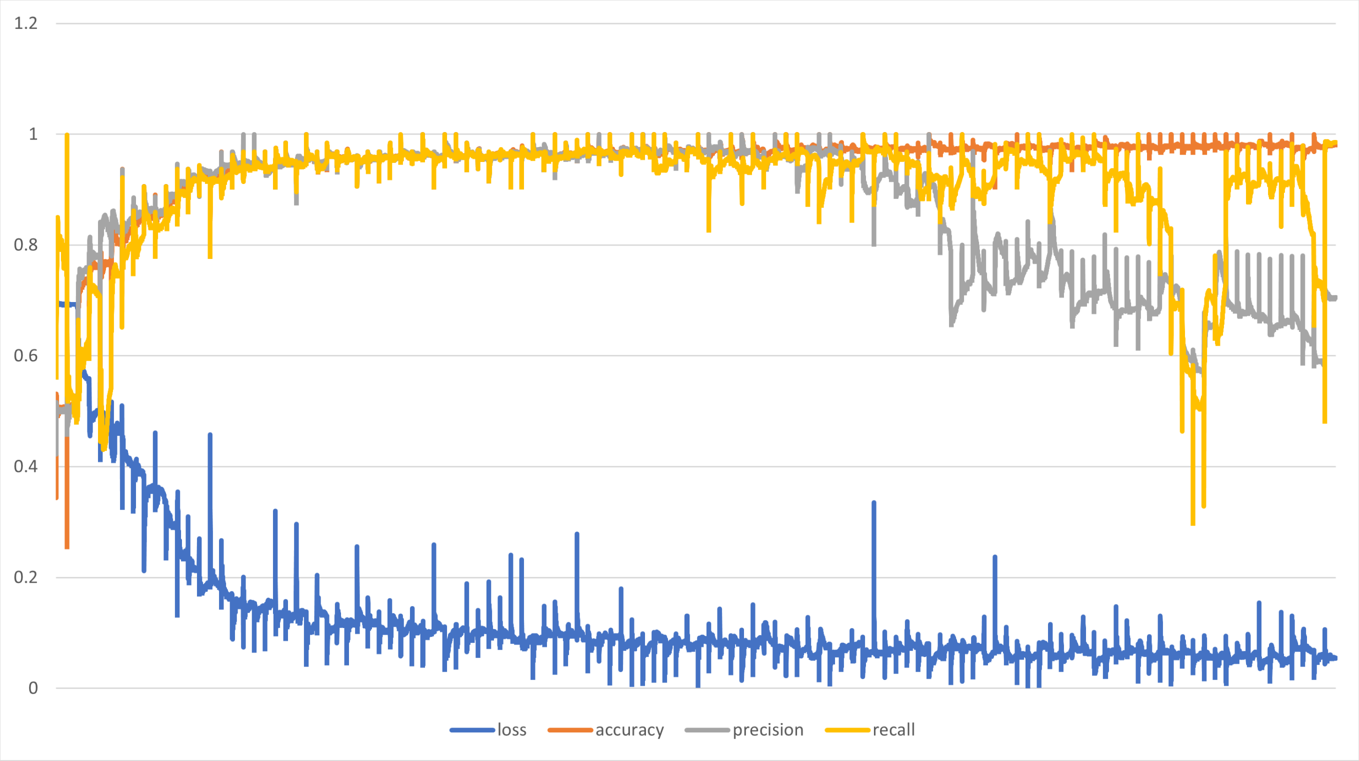
```
early_stopping = EarlyStopping(
    monitor="val_loss",
    patience=20,
    restore_best_weights=True
    # verbose=1,
)
```

Final model selection

During the training process, the performance is evaluated for each epoch. In the end, the weights are reverted back to the epoch with the best performance and it is used as the final model.

Metrics over time

Below shown are the traning metrics over time.



Testing dataset metrics

Below shown are the metrics of final model on the testing dataset.

Property	Value
Accuracy	0.9526542425155640
Precision	0.7135249972343445
Recall	0.9612625241279602
Confusion matrix	[88 177] [156 276]

Terminal output

> Some lines are omitted for brevity, indicated by ellipses.
>
> ```
> 28-11-2023 06:45:27 | prepare => trial hash: 0x1b5c227777c97122
> 28-11-2023 06:45:27 | prepare => output path: ./output/28112023_064527_0x1b5c227777c97122/
> 28-11-2023 06:45:27 | main => starting
> 28-11-2023 06:45:27 | allocateGPUs => allocating GPUs
> 28-11-2023 06:45:27 | allocateGPUs => available GPUs: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
> 28-11-2023 06:45:27 | allocateGPUs => using GPU: PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')
> 28-11-2023 06:45:27 | allocateGPUs => logical GPU: [LogicalDevice(name='/device:GPU:0', device_type='GPU')]
> 28-11-2023 06:45:27 | main => building model
> 28-11-2023 06:45:27 | main => printing model structure
> Model: "sequential"
> _________________________________________________________________
>  Layer (type)                Output Shape              Param #
> =================================================================
>  conv2d (Conv2D)             (None, 254, 254, 32)      320
>
>  max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
>  )
>
>  conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496
>
>  max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
>  2D)
>
>  conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856
>
>  conv2d_3 (Conv2D)           (None, 56, 56, 128)       409728
>
>  max_pooling2d_2 (MaxPooling  (None, 28, 28, 128)      0
>  2D)
>
>  conv2d_4 (Conv2D)           (None, 26, 26, 64)        73792
>
>  conv2d_5 (Conv2D)           (None, 24, 24, 64)        36928
>
>  dropout (Dropout)           (None, 24, 24, 64)        0
>
>  max_pooling2d_3 (MaxPooling  (None, 12, 12, 64)       0
>  2D)
>
>  conv2d_6 (Conv2D)           (None, 10, 10, 32)        18464
>
>  conv2d_7 (Conv2D)           (None, 8, 8, 32)          9248
>
>  max_pooling2d_4 (MaxPooling  (None, 4, 4, 32)         0
>  2D)
>
>  flatten (Flatten)           (None, 512)               0
>
>  dense (Dense)               (None, 128)               65664
>
>  dense_1 (Dense)             (None, 64)                8256
>
>  dropout_1 (Dropout)         (None, 64)                0
>
>  dense_2 (Dense)             (None, 32)                2080
>
>  dense_3 (Dense)             (None, 16)                528
>
>  dense_4 (Dense)             (None, 2)                 34
>
> =================================================================
> Total params: 717,394
> Trainable params: 717,394
> Non-trainable params: 0
> _________________________________________________________________
> 28-11-2023 06:45:27 | main => training model
> Found 6356 images belonging to 2 classes.
> Found 1040 images belonging to 2 classes.
> Epoch 1/1000
> ...
> 199/199 [==============================] - ETA: 0s - loss: 0.0543 - accuracy: 0.9825 - precision: 0.7059 - recall: 0.9858
> 199/199 [==============================] - 37s 184ms/step - loss: 0.0543 - accuracy: 0.9825 - precision: 0.7059 - recall: 0.9858
> Training loss: 0.05431058257818222
> Training accuracy: 0.9825361967086792
> Training precision: 0.7059486508369446
> Training recall: 0.9858401417732239
> 28-11-2023 08:28:31 | main => saving model
> ...
> 28-11-2023 08:28:35 | main => finished
> ```

```
28-11-2023 08:30:25 | prepare => output path: ./output/28112023_064527_0x1b5c227777c97122/
28-11-2023 08:30:25 | prepare => input model path: ./output/28112023_064527_0x1b5c227777c97122/model_epoch109_28112023_082227.keras
28-11-2023 08:30:25 | main => starting
28-11-2023 08:30:25 | allocateGPUs => allocating GPUs
28-11-2023 08:30:25 | allocateGPUs => available GPUs: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
28-11-2023 08:30:25 | allocateGPUs => using GPU: PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')
28-11-2023 08:30:25 | allocateGPUs => logical GPU: [LogicalDevice(name='/device:GPU:0', device_type='GPU')]
28-11-2023 08:30:25 | main => loading model
28-11-2023 08:30:26 | main => testing model
Found 697 images belonging to 2 classes.

 1/22 [>.............................] - ETA: 55s
 3/22 [===>..........................] - ETA: 0s
 ...
22/22 [==============================] - ETA: 0s
22/22 [==============================] - 5s 107ms/step
28-11-2023 08:30:31 | main => printing testing metrics

 1/22 [>.............................] - ETA: 6s - loss: 0.1358 - accuracy: 0.9375 - precision: 0.7500 - recall: 0.9375
 2/22 [=>............................] - ETA: 2s - loss: 0.0937 - accuracy: 0.9688 - precision: 0.7209 - recall: 0.9688
 ...
22/22 [==============================] - ETA: 0s - loss: 0.1558 - accuracy: 0.9527 - precision: 0.7135 - recall: 0.9613
22/22 [==============================] - 2s 90ms/step - loss: 0.1558 - accuracy: 0.9527 - precision: 0.7135 - recall: 0.9613
Testing loss: 0.15576231479644775
Testing accuracy: 0.952654242515564
Testing precision: 0.7135249972343445
Testing recall: 0.9612625241279602
Testing confusion matrix:
[[ 88 177]
 [156 276]]
28-11-2023 08:30:33 | main => finished
```