Group 5: Samuel 1005142 , Matthew 1004263

Topic: Triage Classification

Problem and Objective

In emergency departments (ED) at hospitals, triage nurses facilitate patient flow safely, ensuring timely diagnosis and management. However, lack of triage nurses could cause patient flow to bottleneck. Automating triage classification on incoming patients would help elevate workload on nurses.

Datasets:

For this project, we used 2 datasets from Kaggle that are related to triage.

The first is:
https://www.kaggle.com/datasets/maalona/hospital-triage-and-patient-history-data
The Hospital Triage and Patient History dataset includes patients' symptoms and conditions during triage, as well as their triage class (discharge or admission).

The second is:
https://www.kaggle.com/datasets/erhmrai/ecg-image-data/code
The ECG image dataset contains electrocardiogram (ECG) images of patients and their ECG type (6 classes).

Both are relevant to triage as nurses decide what priority to give patients based on their symptoms/conditions or ECG readings.

Model for first dataset:

For the first dataset (Hospital Triage and Patient History), we first approached by creating 2 models in the Triage folder in Triage.ipynb. The models are SmallModel and PaperModel.

The dataset consists of 972 features and 560,485 samples, with 393,848 discharge labels and 166,638 admission labels.

For both models, we trained taking in 970 features for the initial layer. These features include patients' details such as age, gender, ethnicity, race, religion, medical conditions, symptoms, and vital readings, which covers binary, numeric and categorical features. The department name of the hospital was dropped during the testing, as we felt that this information was not universally applicable to all hospital patients (in other hospitals).

For the architecture of SmallModel, we decided to implement 5 linear layers with ReLU activations, used together with Adam optimizer and cross entropy loss function. With a

learning rate of 0.001 and no weight decay, results of train accuracy = 0.8512 and test accuracy = 0.8518 were obtained.

For PaperModel, we tried to copy a similar version of the DNN used in the paper, using 4 linear layers with ReLU activations and an RMSprop optimizer with learning rate of 1e-3 with cross entropy loss. This model strangely performed much worse than our model, with results of train accuracy = 0.2979 and test accuracy = 0.2950. A possible reason for this was the possibility that the paper used a different set of preprocessing steps, which significantly changed the nature of the data fed into the model.

Even with SmallModel performing its best with test accuracy 0.8518, more complex and pretrained models in Pytorch such as Resnet50, which include more complex features in its architecture such as skip connections, still outperforms our model with accuracy > 0.90.

Next, we filtered the dataset to try and train a model that is able specifically triage for chest pain patients, under the pre-processed.csv excel. We decreased the 970 features to 51, and trained the same models in Triage_proccessed.ipynb.

What we noticed was that the train and test accuracies were slightly lower instead with train accuracy = 0.8108 and test accuracy = 0.8104 for SmallModel. We believe that the decrease in accuracy was due to the removal of patient particulars such as ethnicity, gender, race etc. These features related to demographic information probably contained certain indirect health information, such as the patient's socio-economic status and genetic predisposition to certain types of illness. This indirect demographic information gave a small but notable improvement to patients' triage classification.

Given the relative simplicity of the triage classification task (no state-of-the-art models to compare from) and following the lines of narrowing down on triaging chest pain type patients, we decided to move on to a second dataset of ECG images, as chest pain conditions normally require an ECG scan for more accurate triage.

Model for second dataset:

For the ECG images, the color of the different categories of ECG were different from the dataset, which resulted in the model being 100% accurate. As such, we grayscale the images before turning into black and white, given a certain threshold (RGB value > 128 for white, otherwise black). The model used a train-test split of 80% to 20%. No validation set was used for hyperparameter tuning, due to the immense amount of computation required.

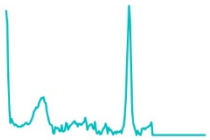

Figure 1: Example of ECG image from dataset (before applying black and white processing)

For the model, we tried to use a relatively recent state-of-the-art YOLOv7 model from https://github.com/WongKinYiu/yolov7 to experiment with the dataset. YOLOv7 is a one-shot detector model which is capable of performing real-time object detection. Running the files train.py and test.py on the image data, our results can be found in /runs/train/exp. With the model and the best epoch and state recorded as .pt files. Running for 20 epochs, the model performs with mean Average Precision (mAP) of 0.88.

Below are figures of the training results (figure 2):

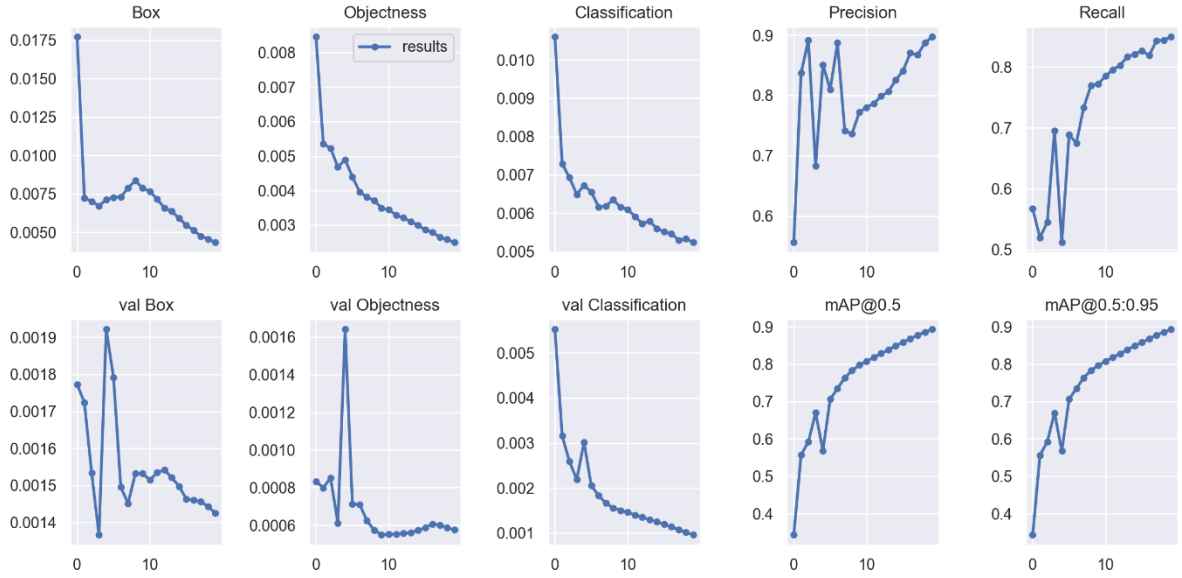

Figure 2: Results of model, noting Precision and mAP scores

The Confidence vs Precision score of model for all 6 classes (figure 3):

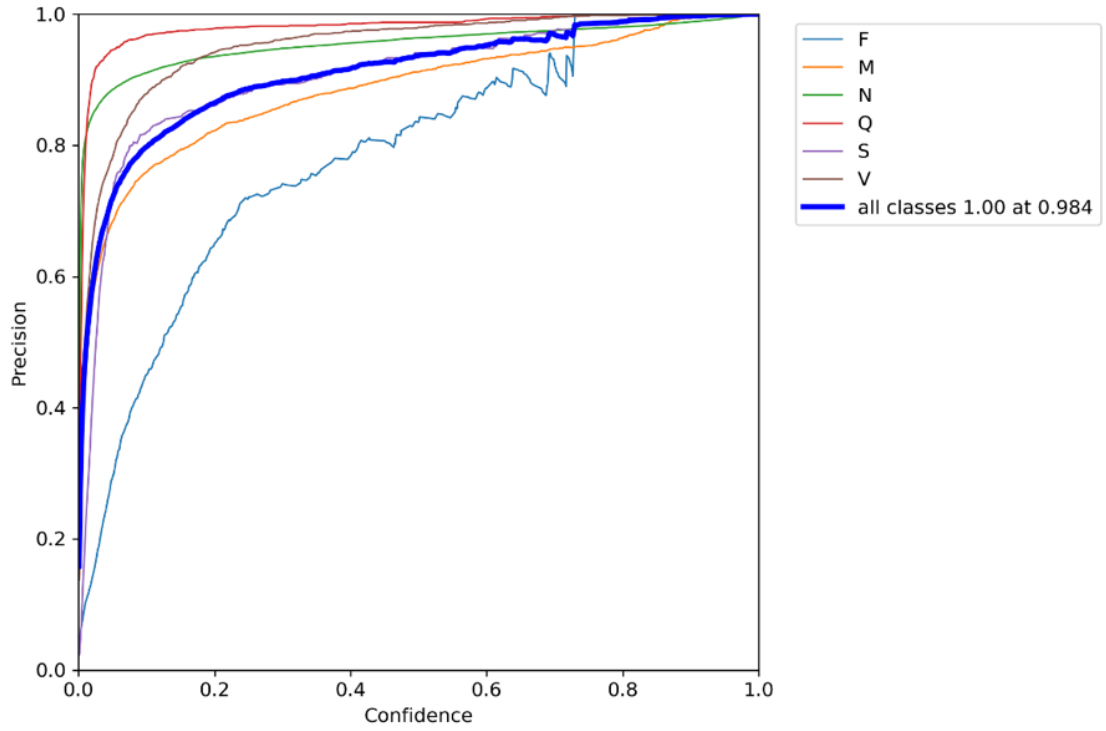

Figure 3: Precision Score diagram

Figure 4 and 5 compares test sample and test output of the ECG images:

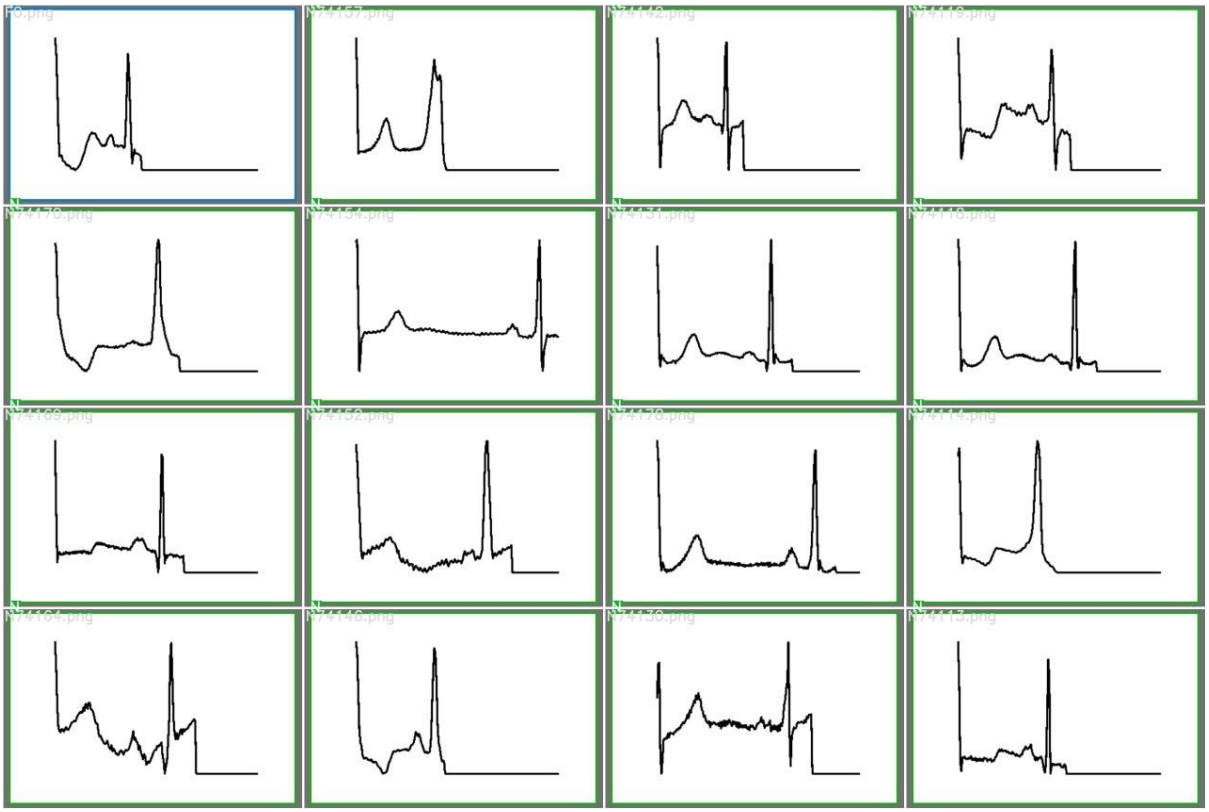

Figure 4: example of test sample

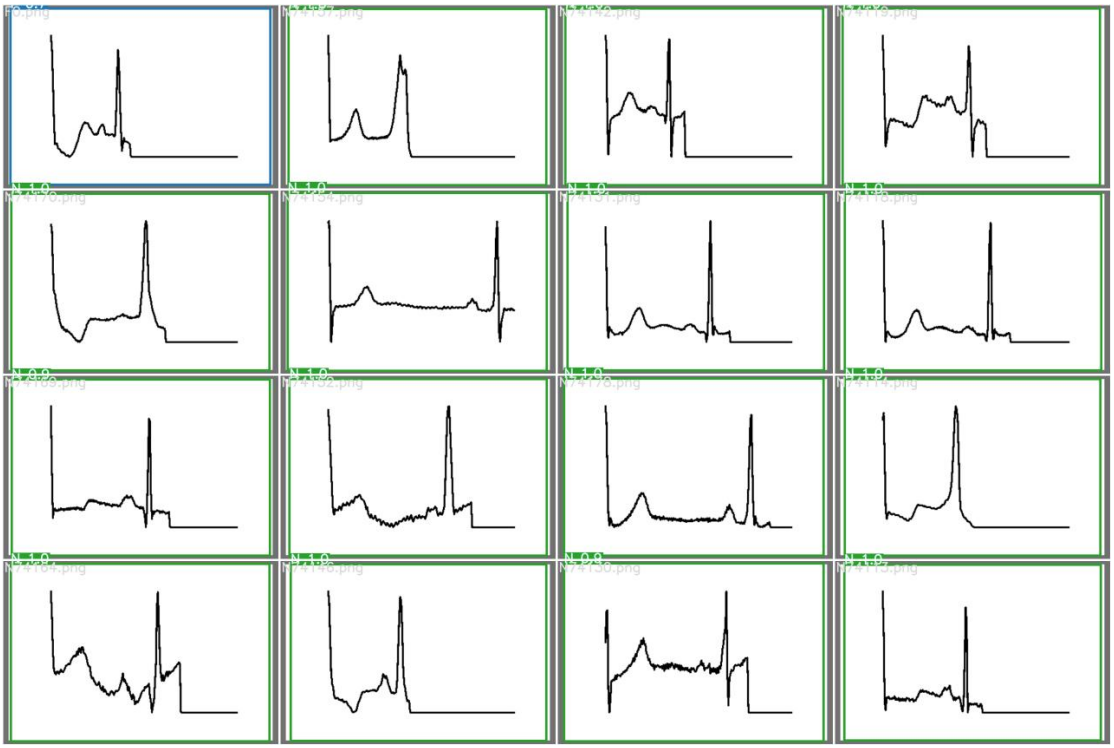

Figure 5: example of test output

Confusion matrix:

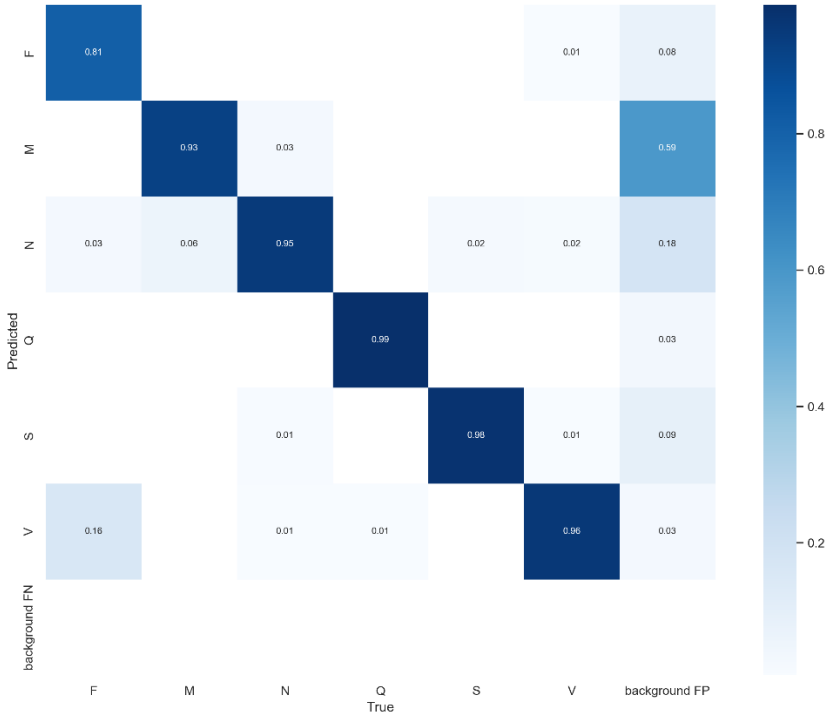

Figure 6: confusion matrix for yolov7

Another pretrained model VGG19, using Adam optimizer and categorical_crossentropy loss function, training 99199 images in batch sizes of 32 images at a time, and using a validation set of 24799 also in batch sizes of 32 images at a time for 10 epochs. The result is a validation accuracy of 0.9998 which performs extremely well.

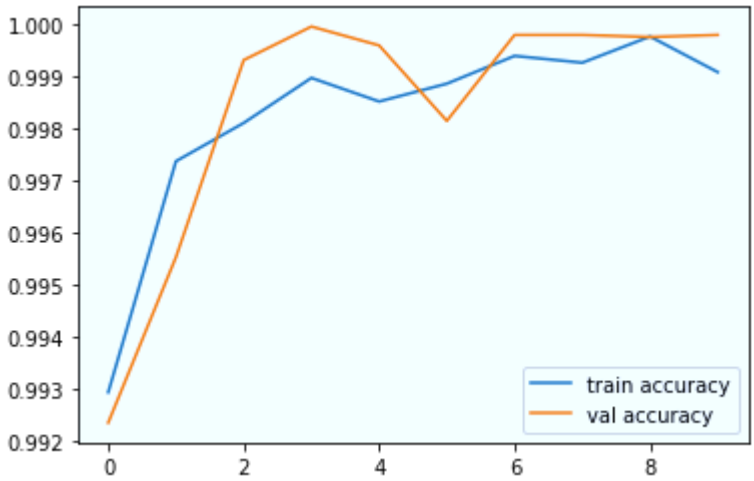

Figure 7: accuracy for vgg19 model

As such, we believe that indeed convolutional neural networks do perform well in training a model to classify ECG images for triage.

Our CNN Model: ECG_Classifier [Code in ECG_images_CNN folder, ECG_CNN.ipynb]

Model uses 5 blocks of conv2d, batchnorm2d and ReLU layers before flattening and passing through linear and ReLU layers until 6 output classes for predictions.

Initially attempted to start of with simple layers of 2 conv2d layers and 2 linear layers, but experienced problem of limited GPU. Next tried to reduce batch size from 256 to 32. However still faced issues of extremely long training duration.

Upon adding more blocks of layers to the model, until the 5 blocks, each batch seems to be training faster. And improved training rate by introducing BatchNorm2d layers and dropout layers to further speed up training.

For training with train validation split:

- ImageFolder, transform images by reducing their size and changing to greyscale
- Train, validation random split. Train size : 90199, Validation Size : 9000
- Optimizer: Adam
- Loss function: Cross entropy
- Learning Rate: 0.001
- 10 epochs, 11275 batches of 8 images each

The results of training after the 10 epochs:

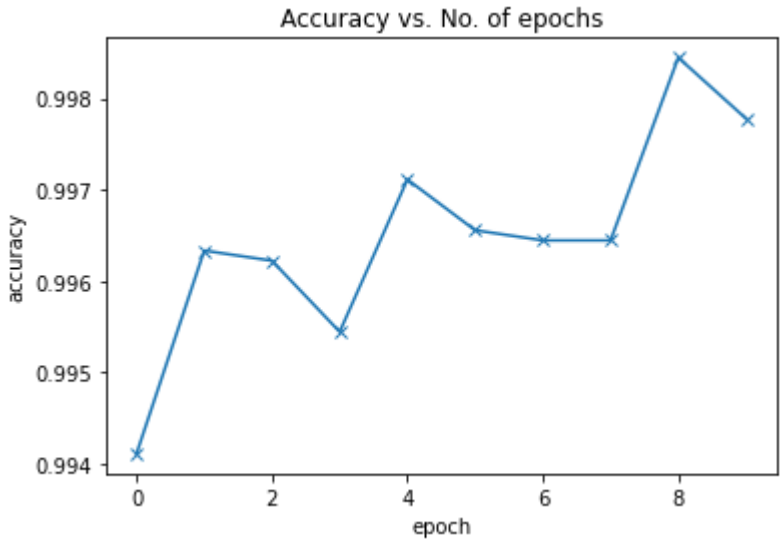

Figure 8: Validation Accuracy vs Epoch

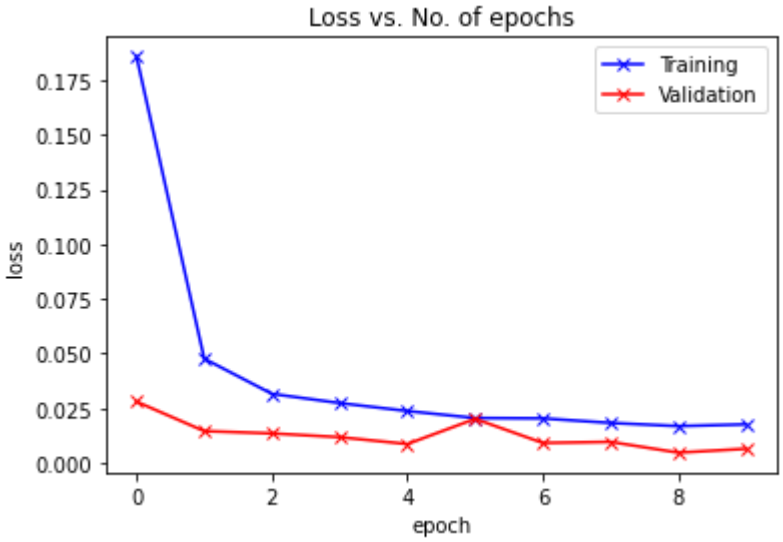

Figure 9: Train and Validation Loss against Epoch

Using trained model to predict on Test dataset of 24k images:

Test Accuracy: 99.7419250776241%

As such, we believe that ECG images can used in conjunction with the first dataset to further improve automation of triaging patients, through the use of training CNN models.

Future improvements:

As of now there is no dataset available that combines both patients' details and symptoms as well as images of their scans such as ECG, which can be used to train a model to complete a full triage process. However, we believe that if such datasets are available and well labelled, existing deep neural networks would be able to perform well in training these models to classify patients for triage.