UNIVERSITY OF PORTSMOUTH

Big Data And Machine Learning Applications

COURSEWORK 2:
Mini Project - Deep Learning for Medical Image Classification
UP2225522

Contents

Introduction & Background

Medical imaging, especially chest X-rays, plays a pivotal role in the early detection and diagnosis of pulmonary diseases such as pneumonia (Gulati & Balasubramanya, 2021). The timely and accurate classification of chest X-ray images into normal and pneumonia categories is crucial for effective patient management and treatment planning. Deep learning models such as CNN are widely employed in real-time medical image analysis to enhance patient outcomes (Li et al., 2023).

It has been shown that successful applications of CNNs in computer-aided disease diagnosis or prognosis have been demonstrated (Shen et al., 2017); however, this project aims to explore the power of deep learning techniques, specifically CNNs for the classification of chest X-ray images. Moreover, transfer learning, a popular approach in deep learning, will be employed to leverage pre-trained CNN architectures, which have been previously trained on large datasets for general image recognition tasks.

In addition to CNNs, this project seeks to integrate support vector machine (SVM) and random forest classifiers with the CNN model. By combining these classifiers, the aim is to take advantage of the strengths of each model and improve the overall classification performance. The effectiveness of the combined models will be evaluated using appropriate metrics to ensure robustness and reliability in distinguishing between normal and pneumonia cases.

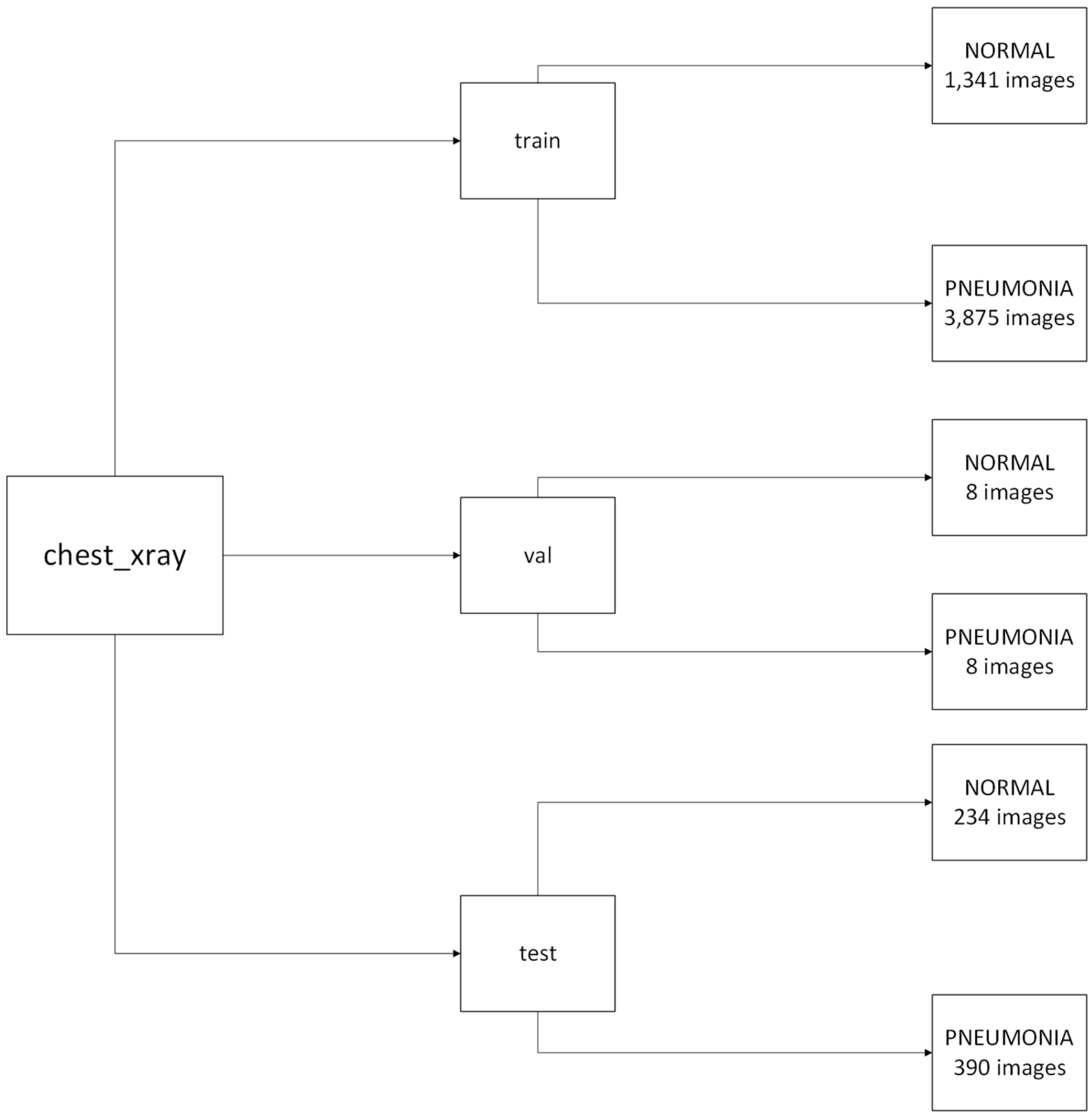
This study aims to contribute to the field of medical image classification by developing an accurate and efficient model for chest X-ray classification. By comparing the performance of the combined models with standalone CNN approaches, this research seeks to assess the potential improvements in classification accuracy and efficiency.

Dataset

The dataset is from Kaggle, Chest X-Ray Images (Pneumonia) (MOONEY, 2018). The Datacard section explains that the dataset has three main folders: train, test, and val, containing images of chest X-rays. These X-rays come from kids aged one to five, taken at Guangzhou Women and Children's Medical Centre. Initially, they removed any X-rays that were blurry or unclear. Then, two doctors looked at the remaining X-rays to label them as normal or showing pneumonia. Another expert checked a portion of the X-rays to make sure the labels were correct. Figure 1 below shows how the file path works for this dataset.

Figure 1

Diagram indicating the file path system of this dataset



Dataset Limitations

• Small validation set with 16 images, limiting performance evaluation.
• Significant class imbalance in the training set, with more pneumonia-related images than normal.
• Images from specific hospitals may not represent diverse cases in other settings, limiting performance.

Methodology

The methodology used in the report is based on the lectures and labs done in class about CNN and recent articles (Donges, 2022; Foster, 2023). Python will be the programming language used and will be implemented in the Jupyter Notebook.

1. Pre-processing:

Before training our model, data preparation is done. This involves validating/cleaning up discrepancies and ensuring all images are in a consistent format and size. Normalisation of pixel values to ensure they're all within a similar range. Additionally, the use of data augmentation techniques to increase the diversity of our training data, such as rotating or flipping images.

2. Model Development:

Convolutional Neural Network (CNN) architecture tailored for classifying medical images will be created. CNNs are great for this task because they can automatically learn important features from the images.

3. Training and Evaluation:

After the pre-processed dataset and CNN architecture are ready, the model will be trained using the training data. During training, the model will adjust its parameters to minimise the difference between its predictions and the actual labels. Once trained, we'll evaluate the model's performance using accuracy, which measures the percentage of correctly classified images.

4. Implement Transfer Learning:

The trained model from the previous step is saved and then pre-processed again removing its last few layers, and replacing them with new layers to fit the specific classification task. Then, additional models like Support Vector Machines (SVM) and Random Forests are trained on top of the features extracted by the pre-trained CNN. Finally, the results are evaluated and compared with the pre-trained CNN using accuracy.

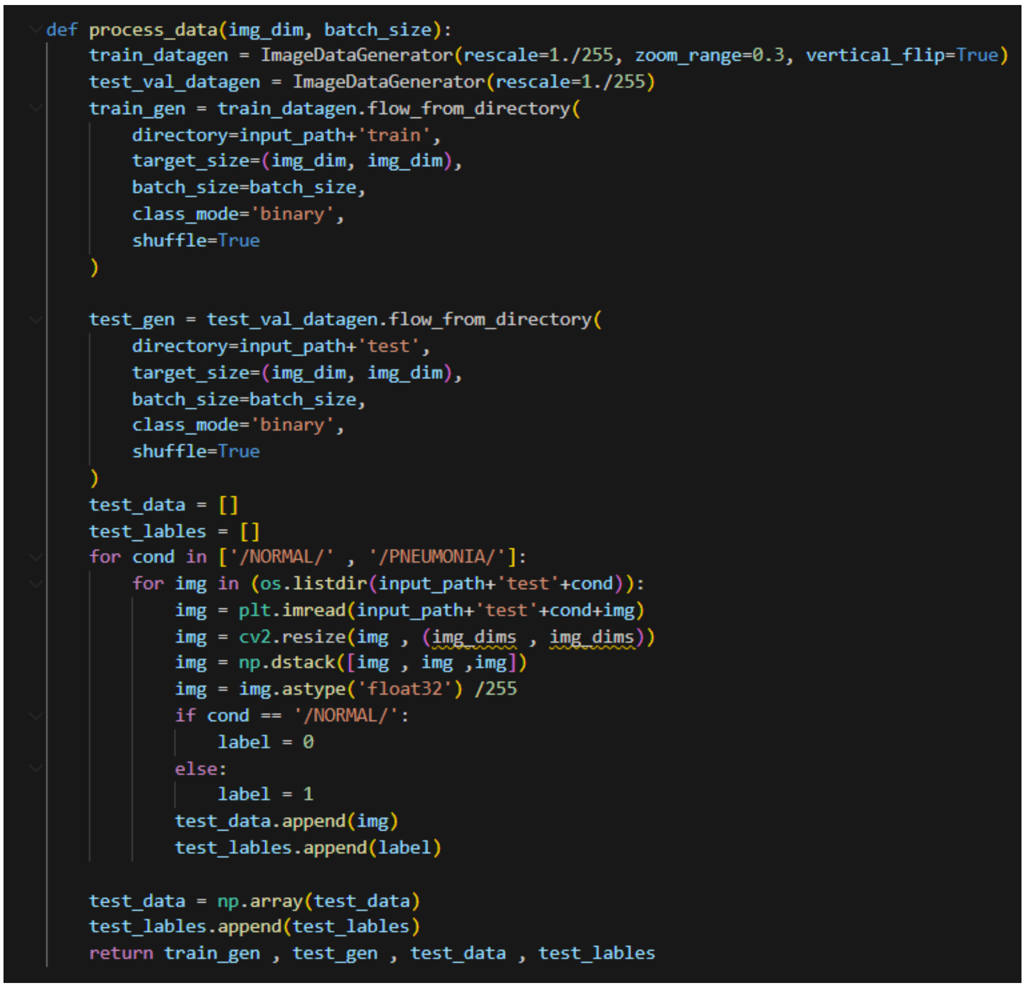
Implementation

Data Pre-Processing

A function process_data(img_dim, batch_size) is created:

Figure 2

Screenshot of the function process_data



```python
def process_data(img_dim, batch_size):
    train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.3, vertical_flip=True)
    test_val_datagen = ImageDataGenerator(rescale=1./255)
    train_gen = train_datagen.flow_from_directory(
        directory=input_path+'train',
        target_size=(img_dim, img_dim),
        batch_size=batch_size,
        class_mode='binary',
        shuffle=True
    )

    test_gen = test_val_datagen.flow_from_directory(
        directory=input_path+'test',
        target_size=(img_dim, img_dim),
        batch_size=batch_size,
        class_mode='binary',
        shuffle=True
    )
    test_data = []
    test_lables = []
    for cond in ['/NORMAL/' , '/PNEUMONIA/']:
        for img in (os.listdir(input_path+'test'+cond)):
            img = plt.imread(input_path+'test'+cond+img)
            img = cv2.resize(img , (img_dims , img_dims))
            img = np.dstack([img , img ,img])
            img = img.astype('float32') /255
            if cond == '/NORMAL/':
                label = 0
            else:
                label = 1
            test_data.append(img)
            test_lables.append(label)

    test_data = np.array(test_data)
    test_lables.append(test_lables)
    return train_gen , test_gen , test_data , test_lables
```

This function processes image data for a machine-learning model. It first defines how to process the training and testing data. For training, it applies various transformations like rescaling, zooming, and flipping to augment the images, then creates a generator to yield batches of processed images along with their labels. For testing, it only rescales the image without any augmentation. Additionally, it manually processes the test images outside the generator loop, resizing them, and normalising pixel values before appending them to an array along with their corresponding labels. Finally, it returns the generators for training and testing data, along with the processed test images and labels. Overall, this function sets up data generators for training and testing for evaluation. It ensures the image data is properly processed and ready for training and testing.

CNN Model Development

Figure 3

Flow diagram of the CNN Model Architecture

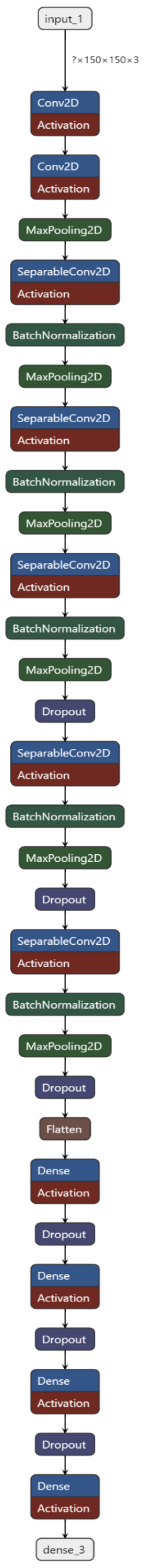
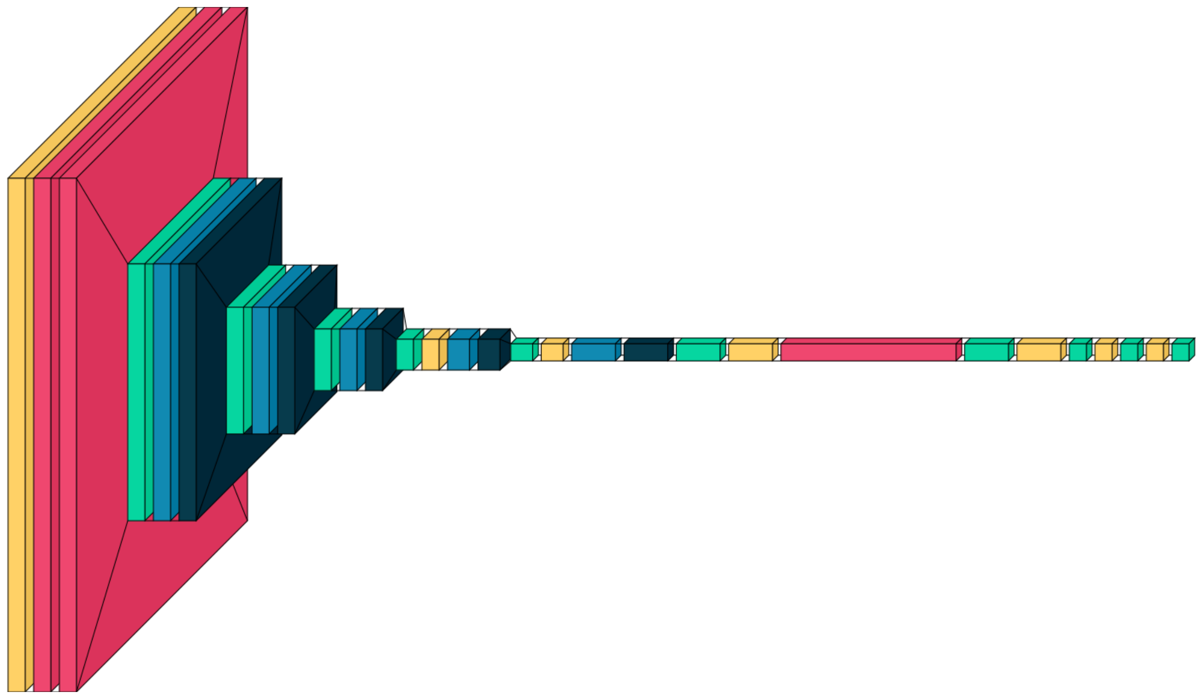


Figure 4

Visualisation of the CNN architecture



This convolutional neural network (CNN) model is designed for classifying chest X-ray images into two categories: normal and pneumonia. It consists of six blocks of convolutional layers, each followed by max-pooling and batch normalisation. The number of filters increases progressively in deeper layers, allowing the model to learn hierarchical features from the input images. Dropout layers are included to prevent overfitting by randomly deactivating neurons during training. The flattened output is passed through fully connected layers with decreasing units and dropout rates, followed by a final sigmoid layer for binary classification. Overall, the model is designed to extract relevant features from chest X-ray images and make accurate predictions while mitigating overfitting through regularisation techniques.

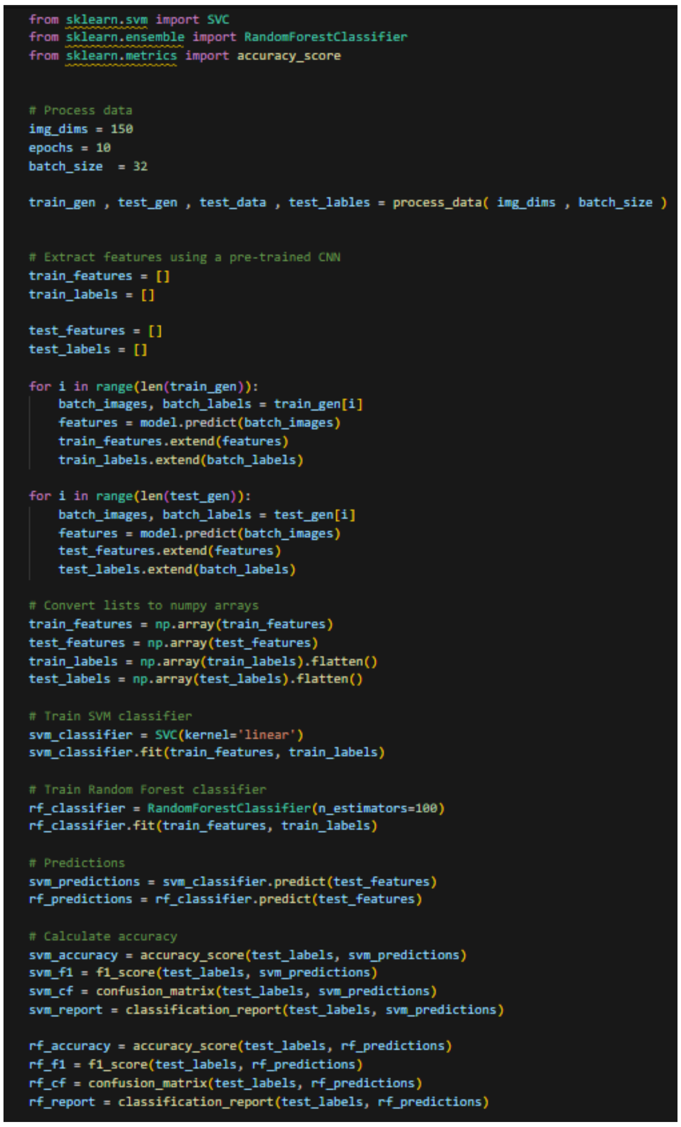
Transfer-Learning

Donges (2022) defines transfer learning as a deep learning technique which allows training with very little data by applying a previously trained model to a new task. Rather than starting from scratch during training, it applies the information gained from a task with a large amount of labelled data to a new, similar task with a smaller amount of data.

For this report, the pre-trained model CNN from the previous section is used for feature extraction and SVM and Decision forest are trained on top.

Figure 5

The screenshot of the implementation of Transfer-Learning



```python
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


# Process data
img_dims = 150
epochs = 10
batch_size  = 32

train_gen , test_gen , test_data , test_lables = process_data( img_dims , batch_size )


# Extract features using a pre-trained CNN
train_features = []
train_labels = []

test_features = []
test_labels = []

for i in range(len(train_gen)):
    batch_images, batch_labels = train_gen[i]
    features = model.predict(batch_images)
    train_features.extend(features)
    train_labels.extend(batch_labels)

for i in range(len(test_gen)):
    batch_images, batch_labels = test_gen[i]
    features = model.predict(batch_images)
    test_features.extend(features)
    test_labels.extend(batch_labels)

# Convert lists to numpy arrays
train_features = np.array(train_features)
test_features = np.array(test_features)
train_labels = np.array(train_labels).flatten()
test_labels = np.array(test_labels).flatten()

# Train SVM classifier
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(train_features, train_labels)

# Train Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100)
rf_classifier.fit(train_features, train_labels)

# Predictions
svm_predictions = svm_classifier.predict(test_features)
rf_predictions = rf_classifier.predict(test_features)

# Calculate accuracy
svm_accuracy = accuracy_score(test_labels, svm_predictions)
svm_f1 = f1_score(test_labels, svm_predictions)
svm_cf = confusion_matrix(test_labels, svm_predictions)
svm_report = classification_report(test_labels, svm_predictions)

rf_accuracy = accuracy_score(test_labels, rf_predictions)
rf_f1 = f1_score(test_labels, rf_predictions)
rf_cf = confusion_matrix(test_labels, rf_predictions)
rf_report = classification_report(test_labels, rf_predictions)
```

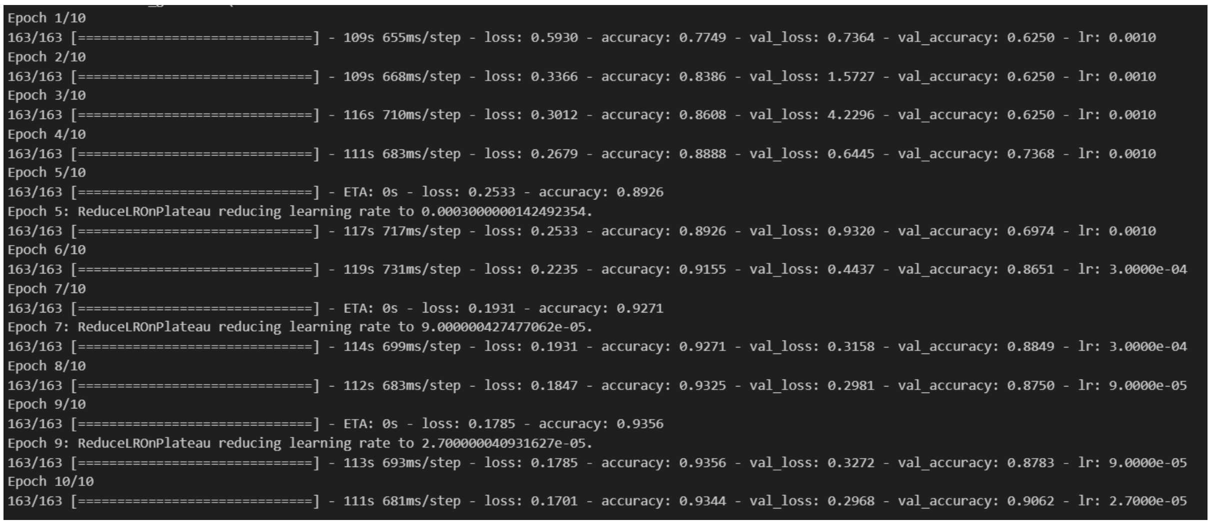
Transfer-Learning is implemented by firstly, processing the data using the previously defined function, setting the image dimensions, epochs, and batch size. Then, it uses a pre-trained Convolutional Neural Network (CNN) model to extract features from the images in both the training and testing datasets. These features are then used to train Support Vector Machine (SVM) and Random Forest classifiers. The SVM classifier uses a linear kernel, while the Random Forest classifier uses 100 decision trees. Next, the classifiers make predictions on the test features extracted earlier. After that, the code calculates the accuracy, F1 score, confusion matrix, and classification report for both the SVM and Random Forest classifiers. These metrics help assess how well the models perform at classifying the images into their respective categories (e.g., normal or pneumonia).

Results

CNN

Figure 6

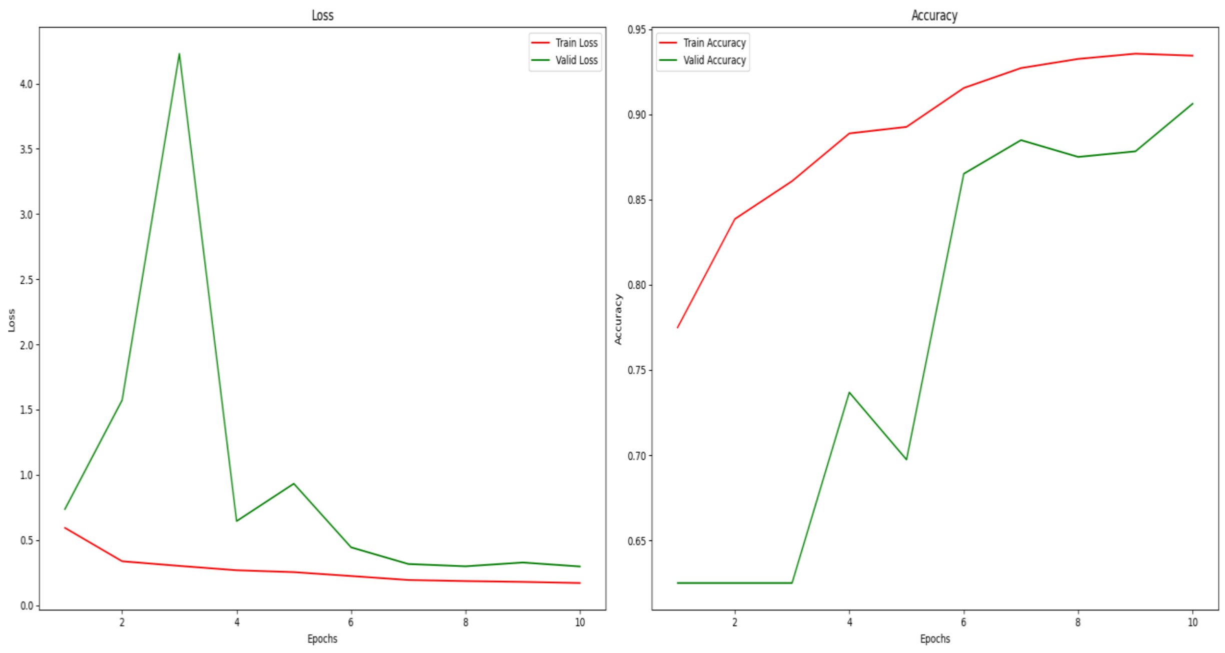
Screenshot of the training process for CNN



```
Epoch 1/10
163/163 [==============================] - 109s 655ms/step - loss: 0.5930 - accuracy: 0.7749 - val_loss: 0.7364 - val_accuracy: 0.6250 - lr: 0.0010
Epoch 2/10
163/163 [==============================] - 109s 668ms/step - loss: 0.3366 - accuracy: 0.8386 - val_loss: 1.5727 - val_accuracy: 0.6250 - lr: 0.0010
Epoch 3/10
163/163 [==============================] - 116s 710ms/step - loss: 0.3012 - accuracy: 0.8608 - val_loss: 4.2296 - val_accuracy: 0.6250 - lr: 0.0010
Epoch 4/10
163/163 [==============================] - 111s 683ms/step - loss: 0.2679 - accuracy: 0.8888 - val_loss: 0.6445 - val_accuracy: 0.7368 - lr: 0.0010
Epoch 5/10
163/163 [==============================] - ETA: 0s - loss: 0.2533 - accuracy: 0.8926
Epoch 5: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
163/163 [==============================] - 117s 717ms/step - loss: 0.2533 - accuracy: 0.8926 - val_loss: 0.9320 - val_accuracy: 0.6974 - lr: 0.0010
Epoch 6/10
163/163 [==============================] - 119s 731ms/step - loss: 0.2235 - accuracy: 0.9155 - val_loss: 0.4437 - val_accuracy: 0.8651 - lr: 3.0000e-04
Epoch 7/10
163/163 [==============================] - ETA: 0s - loss: 0.1931 - accuracy: 0.9271
Epoch 7: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
163/163 [==============================] - 114s 699ms/step - loss: 0.1931 - accuracy: 0.9271 - val_loss: 0.3158 - val_accuracy: 0.8849 - lr: 3.0000e-04
Epoch 8/10
163/163 [==============================] - 112s 683ms/step - loss: 0.1847 - accuracy: 0.9325 - val_loss: 0.2981 - val_accuracy: 0.8750 - lr: 9.0000e-05
Epoch 9/10
163/163 [==============================] - ETA: 0s - loss: 0.1785 - accuracy: 0.9356
Epoch 9: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
163/163 [==============================] - 113s 693ms/step - loss: 0.1785 - accuracy: 0.9356 - val_loss: 0.3272 - val_accuracy: 0.8783 - lr: 9.0000e-05
Epoch 10/10
163/163 [==============================] - 111s 681ms/step - loss: 0.1701 - accuracy: 0.9344 - val_loss: 0.2968 - val_accuracy: 0.9062 - lr: 2.7000e-05
```

Figure 7

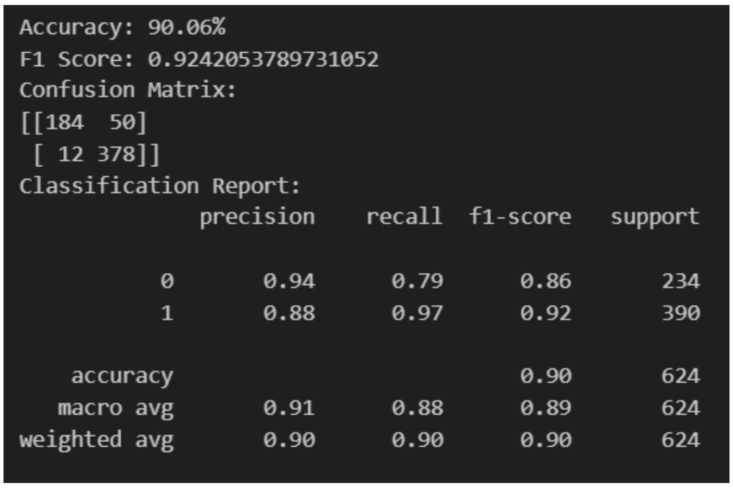
Line graph showing the behaviour of the model for each epoch of training



This Convolutional Neural Network (CNN) has been trained over 10 epochs to classify images, likely for detecting pneumonia from chest X-rays. Initially, the training accuracy was around 77%, gradually improving to approximately 93% by the final epoch. Similarly, the validation accuracy increased from 62.5% to 90.6% over the training period. The model's loss, which measures how well it's performing, decreased consistently during training, indicating that it's getting better at making accurate predictions. Throughout training, the learning rate was adjusted based on the model's performance to fine-tune its training process. In addition, there was a large drop in loss after epochs 3 and 5 indicating that the model has started to converge towards a better solution for the given task, resulting in a more optimised performance.

Figure 8

Screenshot of the CNN classification report



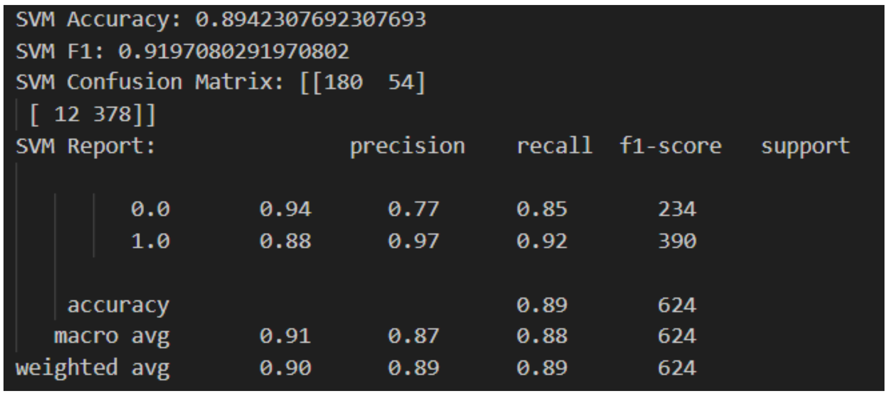
```
Accuracy: 90.06%
F1 Score: 0.9242053789731052
Confusion Matrix:
[[184  50]
 [ 12 378]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.79      0.86       234
           1       0.88      0.97      0.92       390

    accuracy                           0.90       624
   macro avg       0.91      0.88      0.89       624
weighted avg       0.90      0.90      0.90       624
```

The CNN model trained on the provided dataset achieved an accuracy of approximately 90.06% on the test data. Additionally, its F1 score, which balances precision and recall, was around 0.924, indicating a good overall performance. Looking at the confusion matrix, it correctly classified 184 out of 234 normal images and 378 out of 390 pneumonia images, with only 50 normal images and 12 pneumonia images misclassified. This CNN model demonstrates strong performance in distinguishing between normal and pneumonia chest X-ray images.

SVM

Figure 9

Screenshot of the SVM classification report



```
SVM Accuracy: 0.8942307692307693
SVM F1: 0.9197080291970802
SVM Confusion Matrix: [[180  54]
 [ 12 378]]
SVM Report:               precision    recall  f1-score   support

         0.0       0.94      0.77      0.85       234
         1.0       0.88      0.97      0.92       390

    accuracy                           0.89       624
   macro avg       0.91      0.87      0.88       624
weighted avg       0.90      0.89      0.89       624
```

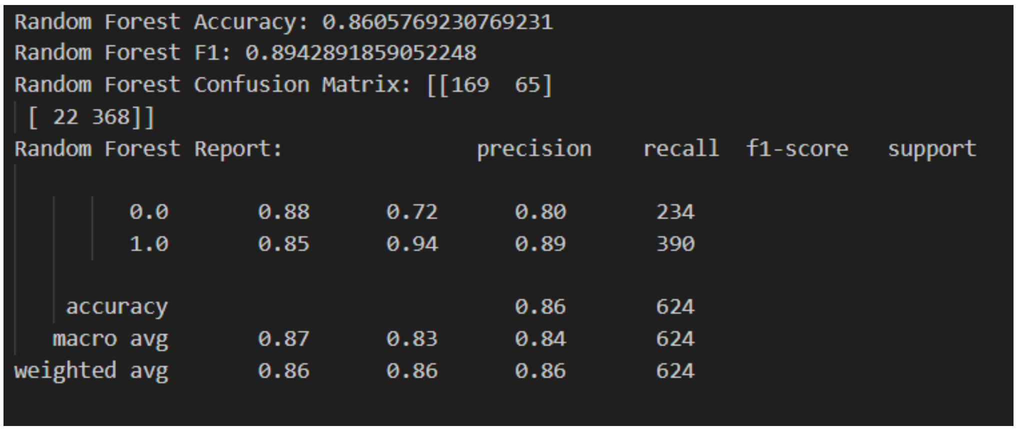
The SVM classifier displayed solid performance on the test data with an accuracy of approximately 89.42%. Its F1 score, indicating a balance between precision and recall, stood at around 0.920, reflecting commendable overall effectiveness. In the confusion matrix, it accurately classified 180 out of 234 normal images and 378 out of 390 pneumonia images, with

54 normal images and 12 pneumonia images incorrectly labelled. The SVM classifier demonstrated good performance, although slightly less than the performance of the CNN model.

Decision Forest

Figure 10

Screenshot of the DF classification report



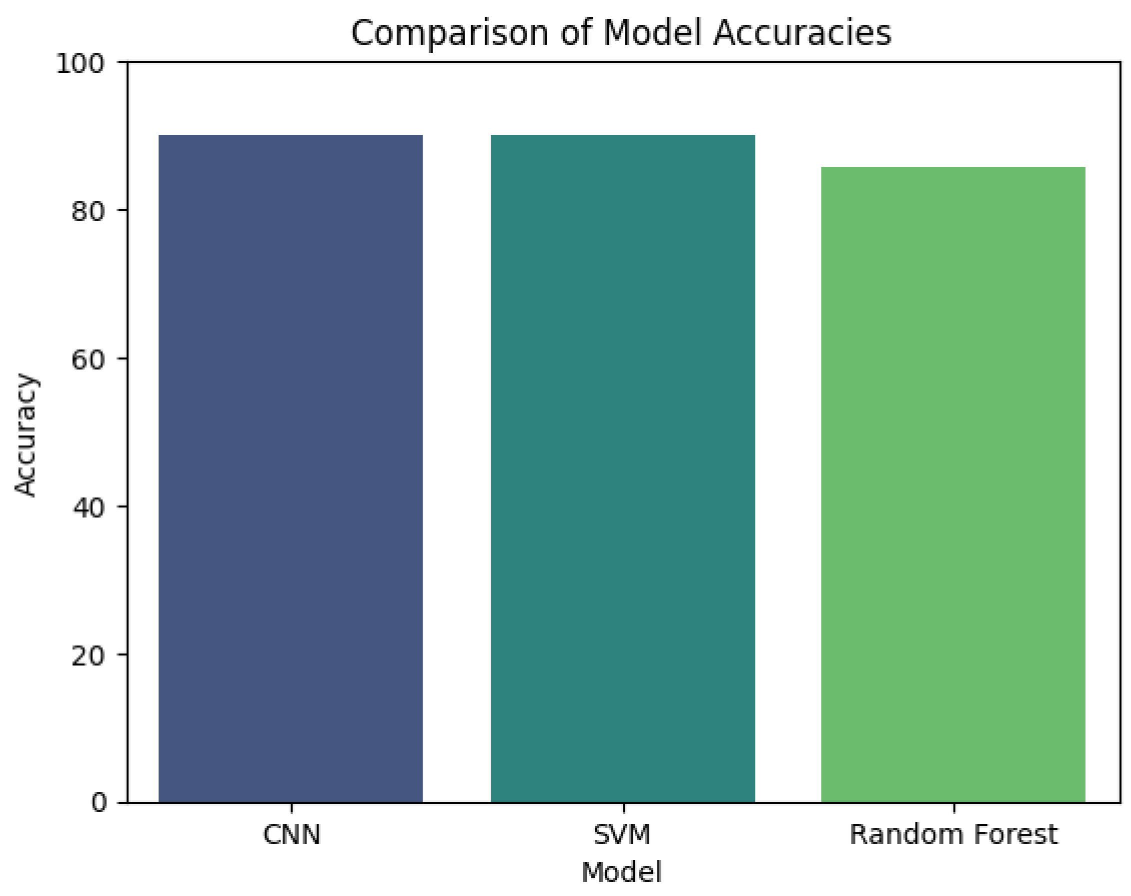
```
Random Forest Accuracy: 0.8605769230769231
Random Forest F1: 0.8942891859052248
Random Forest Confusion Matrix: [[169  65]
 [ 22 368]]
Random Forest Report:              precision    recall  f1-score   support

         0.0       0.88      0.72      0.80       234
         1.0       0.85      0.94      0.89       390

    accuracy                           0.86       624
   macro avg       0.87      0.83      0.84       624
weighted avg       0.86      0.86      0.86       624
```

The Random Forest classifier shows a respectable performance on the test data with an accuracy of approximately 86.06%. Its F1 score, which balances precision and recall, was around 0.894, indicating good overall effectiveness. Reviewing the confusion matrix, it correctly classified 169 out of 234 normal images and 368 out of 390 pneumonia images but misclassified 65 normal images and 22 pneumonia images. In summary, the Random Forest classifier demonstrated solid capability although it has slightly lower accuracy compared to both the SVM and CNN models.

Discussion

Figure 11

Bar plot comparing the accuracy of each model



The CNN model demonstrated the highest accuracy, achieving approximately 90.06% on the test data. This superior performance can be attributed to the CNN's ability to automatically learn complex features from the data, particularly advantageous for image classification tasks. However, its training process requires significant computational resources and time as shown in Figure 6, which may not be feasible in resource-constrained environments.

On the other hand, the SVM and Random Forest models achieved lower accuracies of around 89.42% and 86.06%, respectively. While these models are computationally less demanding compared to CNNs, they may not capture intricate patterns in the data as effectively. Moreover, their performance might have been hindered by the small size of the validation image set, leading to generalisation.

A key limitation across all models is the small size of the validation image set and class imbalance due to the high volume of the pneumonia images. Both factors make it challenging to properly evaluate the models' performance and fine-tune hyperparameters, increasing the risk of overfitting the training data. Additionally, the poor implementation of transfer learning may have impacted the performance of the SVM and Random Forest models. If not executed effectively, transfer learning may fail to leverage the knowledge from pre-trained models, resulting in diminished performance gains.

While the CNN model outperformed the SVM and Random Forest models in accuracy, it comes with higher computational costs. The limitations of all models, including the small validation image set, image class imbalance and potential shortcomings in transfer learning implementation.

Conclusion

In this report, we have gone through various aspects of deep learning for image classification; from pre-processing to successfully developing the CNN architecture and implementing transfer learning.

In conclusion, the comparison of CNN, SVM, and Random Forest revealed varying levels of performance in classifying chest X-ray images for pneumonia detection. While the CNN model exhibited the highest accuracy, its computational demands may not be feasible in resource-constrained environments. The SVM and Random Forest models are computationally lighter but are slightly behind in accuracy, possibly due to limitations in capturing complex image features, class imbalance and the small size of the validation image set. Additionally, the suboptimal implementation of transfer learning may have impacted the performance of these models.

To improve future studies, parallel processing techniques can reduce computational time for CNNs, enabling real-world deployment and observe more behaviour of CNNs in higher epochs. Acquiring larger validation image sets and a more diverse class of images is crucial for better model evaluation and generalisation. Lastly, hyperparameter tuning of both SVM and Decision Forest and/or using a better pre-trained model for medical image classification can potentially lead to better performance.

Word count excluding reference: 2136

Reference

Donges, N. (2022, August 25). *What is transfer learning? Exploring the popular deep learning approach*. Built In. https://builtin.com/data-science/transfer-learning

Foster, L. (2023, April 17). Building a Medical Image Classifier with Deep Learning and Python. Medium. https://medium.com/@lfoster49203/building-a-medical-image-classifier-with-deep-learning-and-python-329f63c8ee89

Gulati, A., & Balasubramanya, R. (2021). *Lung Imaging*. StatPearls.

https://www.ncbi.nlm.nih.gov/books/NBK558976/

Li, M., Jiang, Y., Zhang, Y., & Zhu, H. (2023). Medical image analysis using deep learning algorithms. *Frontiers in Public Health*, *11*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5479722/

MOONEY, P. (2018). *Chest X-Ray Images (Pneumonia)*. Kaggle. https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Shen, D., Wu, G., & Suk, H.-I. (2017). Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering*, *19*(1), 221–248. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5479722/