Deep Learning- Final Course Project

Ariel ID. 316440593, Sarel ID. 208197053

Submitted as final project report for the DL course, BIU, 2024

1 Introduction

As part of our studying in the DL course this semester, we introduce some neutral networks that works with the Kaggle Chest X-ray Pneumonia Dataset, which contains 5,863 chest X-ray images of sick and healthy people. The pneumonia class is combined of two sub-classes, bacterial and viral. In this project we classify the data into the two main categories and into three categories, using both sub-classes as separate categories. We also implement an explainability technique that shows which part of the picture gives us the highest indication of what class it belongs to. The methods we use include transfer learning, anomaly detection, the KNN algorithm, heat maps and more.

1.1 Related Works

There are many models and algorithms in the field of medical diagnostics and machine learning that confronts the problem of diagnosis via a deep neutral network algorithms. An honorable mention is the following article, called "Deep convolutional neural network based medical image classification for disease diagnosis". The article was cited quite a lot and also uses our pneumonia data. The aricle itself can be found here.

Furthermore, Kaggle is full of models tackling various problems that arise from our data. Several of them can be found here.:

Let it go without mentioning, that many of them inspired our models deeply, as we didn't practice any code writing during the course.

2 Solution

2.1 General approach

To solve the task we use 3 approaches, convolutional neural networks (CNNs) with transfer learning, the KNN algorithm, and anomaly detection. afterwards we use the heat map technique to better understand the model's decisions. In order to implement the models we used the Tensorflow and PyTorch python libraries. We divide our goal into 4 tasks. In each we use a different approaches, models and libraries.

2.1.1 task 1

This task is combined of two sub tasks. Classification of healthy/sick and classification of healthy/bacterial pneumonia/viral pneumonia. For both sub tasks we use a CNN that transfer learns from the Resnet Model. The reason we picked a CNN model to learn from is that they excel in preserving the information of a picture. we also used the fine tuning and data augmentation techniques in order to strengthen and improve our models.

2.1.2 task 2

Here in order to tackle the problem we used the KNN algorithm on the picture's embedding vector. We gave access to the embedding vector by creating a self.embedding variable during the forward function of the model. After gaining access to the vector we used python libraries to implement and visualize the KNN algorithm.

2.1.3 task 3

The code addresses the anomaly detection problem by training an autoencoder neural network on normal chest X-ray images. The autoencoder learns to reconstruct normal images with minimal reconstruction error. After training, the encoder part of the autoencoder is extracted, and a kernel density estimator (KDE) is fitted on the encoded representations (latent space) of the normal images. For a new test image, the code computes its reconstruction error from the autoencoder and its density score from the KDE model. If either the reconstruction error is high or the density score is low compared to predefined thresholds, the image is classified as an anomaly (potentially pneumonia). This approach leverages the autoencoder's ability to reconstruct normal data well while struggling with anomalies, combined with the density estimation of the normal data distribution in the latent space to detect outliers or anomalies.

2.1.4 task 4

For this task we wanted to visualize the way the CNN makes it's decisions by creating a heat map that shows which part of the image is the most relevant when making the decision of which class the image belongs to. We created the heat map by using hooks that showed which of the gradients was most significant at the last layer of the model. by doing that we show which part of the picture is the most relevant right before the model decides whether the image is of a sick or a healthy person at the fully connected stage.

2.2 Design

For all models we increased the validation set on the expense of the training set, as we saw that good validation is more important then a bit larger training set.

2.2.1 task 1

At first we tried various architectures of many CNNs, but due to long training time and low accuracy result we focused on using transfer learning from the well known resnet model. In sub task 1.1 we used transfer learning from the resnet 101, with some data augmentation and fine tuning of the last FC layer. For the loss function we picked the Cross Entropy Loss, as it is fit for our model. The optimizer we picked is the AdamW, which is a more stable version of Adam. We also used a learning rate scheduler function we found online, as we saw it is improving our results. For sub task 1.2 we created various models, but remain with two winners, as they both achieved high scores. the first is based on resnet 18, with a convolutional layer before it in order to fit the data to the model. The other is based on resnet 101 and is very similar to the model from the previous sub task (but with 3 output classes of course). For both models we picked the same loss and optimizer as in sub task 1.1, as we saw they were very effective.

2.2.2 task 2

As mentioned before, we used python libraries both for classification and visualization, so there is not much to talk about design.

2.2.3 task 3

The anomaly detection design consists of an autoencoder neural network architecture and a kernel density estimation (KDE) model. The autoencoder is trained to reconstruct normal chest X-ray images, with the encoder part learning to compress the input images into a lower-dimensional latent space representation. After training, the encoder is extracted, and a KDE model is fitted on the encoded representations of the normal training data. For a new test image, its reconstruction error from the autoencoder and its density score from the KDE model are computed. If either the reconstruction error exceeds a threshold or the density score falls below a threshold, the image is classified as an anomaly. The autoencoder's ability to reconstruct normal data well while struggling with anomalies, combined with the density estimation of the normal data distribution in the latent space, enables the detection of anomalous or outlier images.

2.2.4 task 4

For better understanding of the decisions made by the model trained in sub task 1.1, we implemented the Grad-CAM (Gradient-weighted Class Activation Mapping) technique. Grad-CAM computes the importance of each feature map by taking a weighted sum of the gradients of the target class activation with respect to the feature maps. This process allows Grad-CAM to provide more localized and precise visualizations of the important regions in the input image for making the classification decision. The output of Grad-CAM we implemented is a heat map overlaid on the input image, where regions with higher intensity correspond to regions that are more important for the prediction of the target

class. By visualizing these heat maps, we can gain insights into how the CNN model is making its decisions and which parts of the input image are influencing those decisions the most.

3 Experimental results

3.1 task 1

In this task as mentioned, we aimed to classify the images into categories: normal and pneumonia (that later was divided into two separate categories). Our first model achieved a test accuracy of about 87 percent, when most of the misclassifications were false positives. We show it all in our jupiter notebook (we also created a confusion matrix to better understand the results). Our second models achieved a score of 80-82 persent, when the bacteria predictions were the most accurate by far (usually less then 10 misclassifications of these images). During our model constructing, we tried various model architectures before we decided to use transfer learning, but due to high training time and low test score we abandoned these ,models. after we decided we'll use transfer learning, we explored many models but the best results we got were with resnet. training the models took about 1-1.5 minutes per epoch, thanks to a rather strong GPU we have on one of our computers. we trained for 10-20 epochs per model for optimal results (we used our enlarged validation set to pick the right model before overfitting). The loss function and optimizer we used were chosen empirically by receiving the best results by a long shot. All the results backing all of these claims were sadly not recorded, as we attended the report only after finishing our project. Alas, the rather high results back our claim, as they were very hard to achieve. For any other inspection I guess you'll have to take our word for it (or try to play with the code and witness yourself). our deepest apologies for that.

3.2 task 2

This task was rather easy compared to the rest, as the already written function we used worked smoothly. For sub task 1.1 we got an accuracy rate of 86 percent, and for sub task 1.2 we got a 76-78 accuracy rate. We believe that the lower scores (which were lower by a bit for every model we used) are because the KNN algorithm is not as strong as the fully connected layers models usually have. Also for this section we provided a confusion matrix that better explains the results. We also creates a t-SNE visualization of the embeddings that clearly shows the clusters created by the different classes on the plain.

3.3 task 3

The experimental setup involves training an autoencoder model on normal chest X-ray images from the training set, and evaluating its performance on separate validation and anomaly (pneumonia) test sets. The autoencoder is trained to

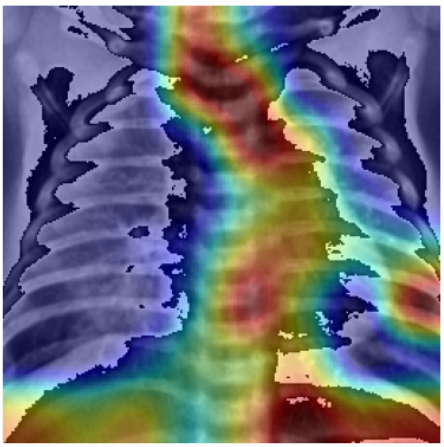

Figure 1: Task 4 results

minimize the reconstruction error, measured by the mean squared error loss function. During the evaluation phase, two key metrics are computed: reconstruction error and density score. The reconstruction error is calculated as the mean squared error between the input image and its reconstructed counterpart from the autoencoder. The density score is obtained by feeding the encoded representation (latent space) of the input image into a kernel density estimation (KDE) model fitted on the encoded normal training data. Lower density scores indicate a higher likelihood of the input being an anomaly. To classify an image as normal or anomalous, predefined thresholds are set for both the reconstruction error and density score. If either metric exceeds its respective threshold, the image is classified as an anomaly (pneumonia). The code also provides a function to calculate the average and standard deviation of the reconstruction error and density scores for both normal and anomaly batches, which can aid in setting appropriate thresholds.

3.4 task 4

As we really liked the Grad-CAM explainability technique, we didn't explore any other techniques in this project (which is fine as only one technique exploring was required). Our results clearly shows by the heat map which part of the picture is the most relevant to the classification decision and by that help us to better understand the decision making process our model does. An example is shown here at figure 1, for more please inspect the notebook.

4 Discussion

This final project has been a challenging yet enlightening experience, teaching us by practice more about the deep learning and neural networks world. We produced some models, that helped us tackle a classification problem that impicates modern medicine and medical diagnosis. We learned how to pick a model, train it and achieve high test accuracy, which made us proud of ourselves. As part of our study for the project, we encountered many obstacles that we faced using all the material we learned during the course (and chat GPT of course :)). We also learned the hard balance between accuracy and computational efficiency, a hard lesson indeed which costed us a lot of time. Overall we feel the entire experience was very educating, and we hope that the knowledge we gained will help us along the way of becoming computer scientists.

5 Code

All the code backing all of the above and more can be found by clicking this sentence. Code for task 2 appears in the end of section 1.1 and 1.2.