Technische Universität München

Transfer Learning

AI in Medicine I: Practical Exercise 3

Authors:

Name	ID
Muhammed Elbushnaq	03786474
Mohammed Elmasry	03786399
Evangelos Fragkiadakis	03786480
Nada Elsherbeny	03788337

Note:

Evangelos Fragkiadakis's ID is **0378648****0**. However, it was written as **0378468****0** in the previous assignments. We apologize for this mistake and hope that it didn't cause any issues with you.

Introduction

In this assignment, we get to use the chest X-ray dataset from MedMNIST to perform various tasks. The dataset includes 14 classes for each data point, representing different medical conditions. We apply dimensionality reduction techniques and transfer learning to examine the differences in model performance.

Task 1

During Task 1, we initially modified the multi-label ChestMNIST dataset to binary (healthy-respiratory disease). Then, we train a simple CNN on the classification problem using a small data sample of 300 images for 20 epochs. As we can observe, the model is overfitting since the training accuracy is near perfect (around 99.7%), but test accuracy is only 58% which indicates that the model struggles to generalize to previously unseen data.

(a) Low data Performance

2- Is accuracy appropriate here?

Given that the positive rate is around 45%, which is not too far away from 50%, that shows that it's not perfectly balanced Data (Moderate imbalance). The accuracy can be a useful metric here (in the binary classification task that only shows Diseased vs Healthy). But given this slight imbalance considering metrics like F1-score would be very good, and also using precision and recall with accuracy can give a better understanding of the model performance on both classes, especially when one class is more important than the other, in this case, I'd suggest the classifying and not missing diseased class is more critical in this task, so using Recall (Sensitivity) would be very useful here too.

3- What would be a better way to select a model than a fixed number of steps that would allow us to set a very high number of epochs and not overfit too much? How would we select the best model using such a ”smart train” function if we are still worried about overshooting our optimum?

In our effort to think of ways that might solve this problem without risking overfitting, we identified four different methods that can be implemented either separately or in combination. The first one is early-stopping, where the model does not allow the network to keep training if a metric does not improve for a number of consecutive epochs. This number is called ‘patience’ and is specified by the user. The second method that could be used is ‘dropout’, where we set a ratio that determines the percentage of randomly selected weights that are not trained in every epoch. A third method is the application of checkpoints. We set epoch intervals that the model saves the weights while training, and then the model sets the weights of the best-performing checkpoint. Last but not least, we can use a method that has proven valuable in scenarios with limited data like ours: cross-validation. In this method, we set a number of folds in our train dataset (k-fold cross-validation), train our network separately for each fold, and then provide an overall performance metric for our model. Using the above methods can improve model performance and alleviate the need to manually repeatedly tailor the number of epochs to reach the optimum solution, thus saving time.

(b) Latent Space

Figures 1 shows the visualizations of the latent space using PCA and t-SNE.

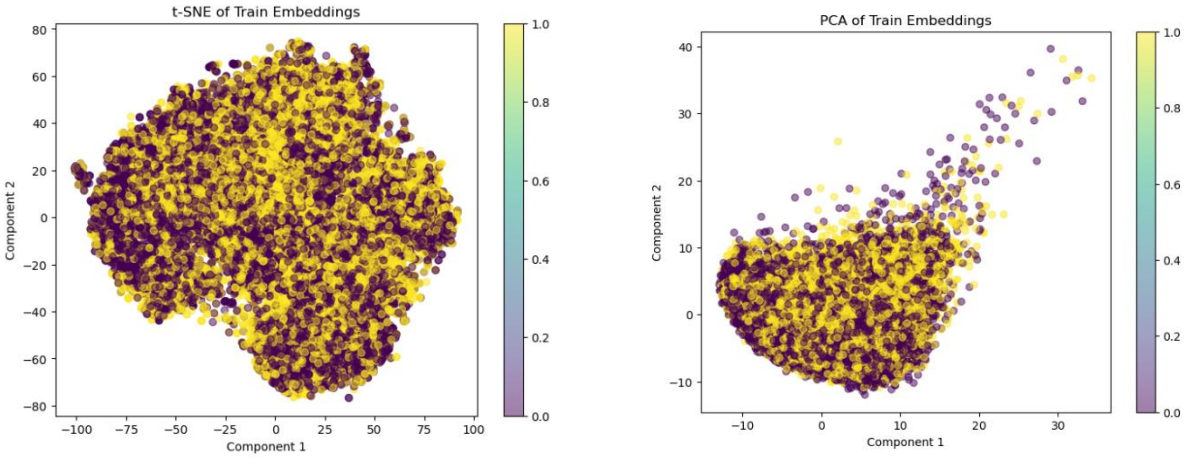

Figure 1: PCA (right) and t-SNE (left) of the embeddings of the chestMNIST CNN.

(c) Autoencoder Performance

Finishing the first task, we develop an autoencoder utilizing the architecture of the previous network to form the encoder and the latent space. Then, we train it on the full dataset to generate the given input images for 10 epochs. Observing Figure 2 and Table 2, the model recreates the images very roughly, which indicates that it has identified the fundamental structures within an X-ray image but is not able to reconstruct all the details. While the autoencoder is not useful on its own, we can transfer the weights of the models to a new instance of our previous smaller model to examine whether this results in improved performance.

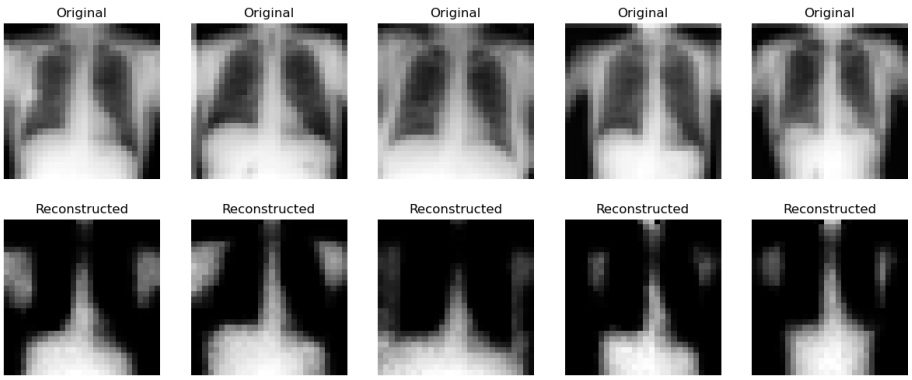

Figure 2 Autoencoder's input and output of the first 5 images of the dataset.

Test Metrics	Score
MSE	0.0025
PSNR	13.8898
SSIM	0.1947

Table 1: Test metrics of theautoencoder.

After training the new instance of the CNN with the transferred weights, we observed little to no difference in the test accuracy (Table 2). Still, we saw a significant drop in the training accuracy, indicating that the model might be better at generalizing, given some favorable circumstances.

Set	Accuracy
Train	0.784
Test	0.594

Table 2: Train and Test Accuracy of the CNN with transfer learning on the small dataset.

Task 2

Moving on to the second task, we use PneumoniaMNIST, which has relatively few labels compared to ChestMNIST. For this reason, we first train an instance of the first model on ChestMNIST to use the weights for transfer learning. After transferring the weights to a new instance, we train it on the PneumoniaMNIST, once freezing all layers except the final fully connected layer and once with finetuning. Both transfer learning techniques produce slightly better results but with no statistical significance. What is worth mentioning, though, is the fact that in the model version, where we freeze all layers except the final fully connected layer, the runtime is much faster (~ 40% faster). On the other hand, when we train all layers after transferring the weights, we get better accuracy in both training and test sets. Table 3 summarizes the training and test accuracies for each case.

Accuracy - No transfer	
Train Set	1.00
Test Set	0.86
Accuracy - Layer Freezing	
Train Set	0.9584
Test Set	0.8109
Accuracy - Trainable	
Train Set	0.9998
Test Set	0.8574

Table 3: *Train and Test Accuracy of the CNN on the smalldataset.*

2a) Freezing Networks

Transfer learning is mainly used in Literature as most of the time it's better to have a pre-trained network to have a good initialization for the weights, especially when the network is trained on data with a similar distribution. Also, the pre-trained networks are trained in a large amount of data, whereas in new research, there are limitations on the amount of data. So it's hard to learn the features of this small amount of data. Furthermore, as discussed before, it's way faster to transfer weights instead of learning the whole network from scratch.

Here, we're testing the efficiency of such a thing. Where we transfer the weights that was trained on the chestMNIST, with 112,120 example and 14 classes. To the task where we only have 5,856 examples with 2 classes only. Where both data are chest x-ray images.

We can see the latent space in Figure 3, which can now be separated, unlike before, as the model was trained in larger amount of data and learned how to represent this data in the latent space, and can now be used easily in the binary classification task of the Pneumonia.

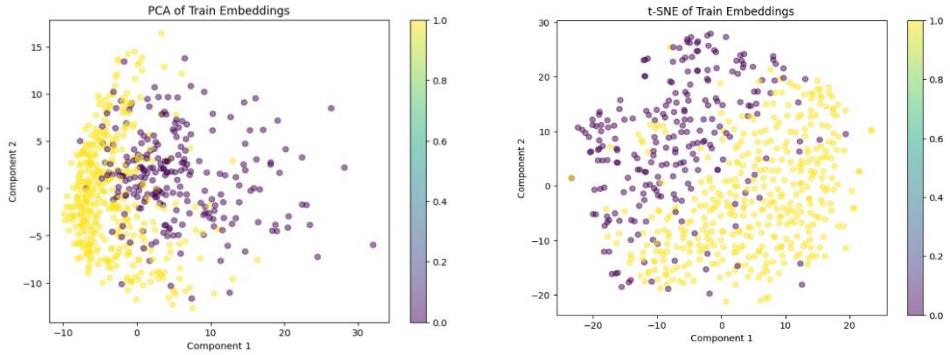

Figure 3: PCA (left) and t-SNE (right) of the embeddings of the PneumoniaMNIST CNN with frozen layers.

2b) What worked better? Freezing or leaving the weights trainable? Why?

Training all layers shower better and more sparable latent space (figure 4), as now we can consider the transfer learning as a good initialization of the data, while fine tuning the whole layers to be specifically for our task, Pneumonia classification. Not only the final Fully Connected Layer.

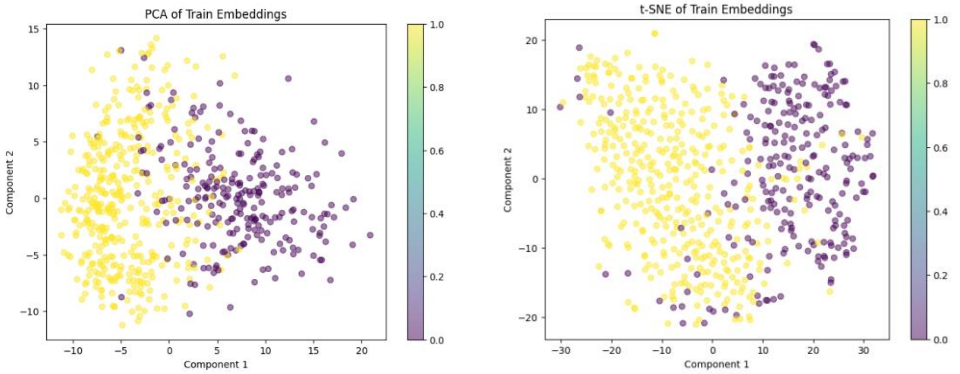

Figure 4 PCA (left) and t-SNE (right) of the embeddings of the PneumoniaMNIST CNN with trainable layers.

In terms of Accuracy, as shown in Table 3, training all the layers was better in both the Training and Test sets, but in terms of run time, training only the final layer was much faster. As now the model is learning Task-Specific Features, rather than the general feature from the transfer learning.

2c) Latent Space Musings

First, with the ChestMNIST latent space, it was inseparable. However, when we transferred it to the Pneumonia binary task, it was separable in both cases, as the model had already learned the general features from a more challenging task with much more data, with data from a similar distribution, and now is transferred to a data that is inherently binary and does not contain multiple dimensions like ChestMNIST. Having all CNN layers froze showed slightly worse results from training all of them, this can be interpreted as while training all the layers, the model is now learning more task-specific features, with a very good weights initialization. Furthermore, we also noticed better performance in the t-SNE than in the PCA. In general, PCA is a linear method, assuming that all data points are linearly dependent. This way, it linearly projects the data points on the maximum variance vectors. On the other hand, t-SNE is a non-linear method, meaning it can handle data with non-linear dependencies. Furthermore, t-SNE uses a manifold topology which distorts the feature space to keep similar data closer in their 2D representation. This is particularly helpful when we want to identify clusters in our data. PCA is affected more by the trainability of the layers (Fig. 3 and 4).

Conclusions

In conclusion, transfer learning can have dual benefits for a model. It can improve its performance and also accelerate the learning process. Furthermore, we established awareness of the suitability of the metrics that we use based on the data balance. Finally, we acquired valuable information about how PCA and t-SNE can provide different results for different scenarios. PCA performs better in linear data and t-SNE in non-linear data. Furthermore, fully trainable models can perform better than the CNN-froze networks, but take more time.

References

[A. Ramezan *et al.*, 2019] A. Ramezan, Christopher, A. Warner, Timothy, & E. Maxwell, Aaron. 2019. Evaluation of Sampling and Cross-Validation Tuning Strategies for Regional-Scale Ma- chine Learning Classification. *Remote Sensing*, **11**(2).

[Allamy & Khan, 2014] Allamy, Haider, & Khan, Rafiqul Zaman. 2014 (01). Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study).

[Anowar *et al.*, 2021] Anowar, Farzana, Sadaoui, Samira, & Selim, Bassant. 2021. Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE). *Computer Science Review*, **40**, 100378.

[Eisenman *et al.*, 2022] Eisenman, Assaf, Matam, Kiran Kumar, Ingram, Steven, Mudigere, Dhee-vatsa, Krishnamoorthi, Raghuraman, Nair, Krishnakumar, Smelyanskiy, Misha, & Annavaram, Murali. 2022. Check-N-Run: a Checkpointing System for Training Deep Learning Recommen- dation Models. *Pages 929–943 of: 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association.

[Sitaula & Ghimire, 2017] Sitaula, Chiranjibi, & Ghimire, Nabin. 2017. "An analysis of Early Stopping and Dropout regularization in deep learning". *International Journal of Conceptions on Computing and Information Technology*, **5**(1), 17–20.

[Zhuang *et al.*, 2021] Zhuang, Fuzhen, Qi, Zhiyuan, Duan, Keyu, Xi, Dongbo, Zhu, Yongchun, Zhu, Hengshu, Xiong, Hui, & He, Qing. 2021. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, **109**(1), 43–76.