Bernstein Center for Computational Neuroscience, Berlin

Transfer Learning for MRI

Oliver Atanaszov, Dr. Kerstin Ritter

April 12, 2019

Abstract

Early diagnosis of neurodegenerative diseases is crucial in preventing the progress and treating disases such as Alzheimer's and Parkinsons's. Identifying disease-related biomarkers using neuroimaging requires a great expertise and effort from radiologists. Deep learning methods have been immensely successful in learning useful data representations in an end-to-end fashion, surpassing human performance in many fields where labelled data is available in abundance. Thus, they are a promising tool in medical image analysis too, with the potential to help revealing novel pathological biomarkers or provide assistance to radiologists. However, the common limitations of such algorithms is evident: neuroimaging datasets are typically scarce, making representation learning and downstream tasks very cumbersome. The methods often yield unreliable results, that are far from satisfactory in a clinical setting. This report introduces a framework, transfer learning, in attempt to circumvent this deficiency. Transfer learning aims to improve learning in a task through the transfer of knowledge from a related, but different task that has already been learned. We discuss will its potential application to MRI data, in particular, when the number of samples is limited.

Key words: brain MRI, Alzheimer's disease, deep learning, autoencoder, transfer learning

1 Introduction

Neurodegenerative diseases become more and more prevalent, especially in the elderly population. Alzheimer's disease (AD), a progressive brain disorder is the most common cause of dementia. It is estimated to affect one of every 85 persons by 2050 [10] [1]. Parkinson's disease (PD) affects around one percent of the population over the age of 60 [12]. There is no cure for these diseases; treatments are typically aimed at improving symptoms. Evidently, such conditions not just cause significantly worsened quality of life for the patient, but places a heavy burden on the patient's kin. Furthermore, AD is one of the most costly diseases in developed countries [3]. Consequently, early-stage diagnosis of neurodegenerative diseases is crucial; by identifying biomarkers, (i.e. subtle, pathological deviations of the brain's structure), the patient's potential condition can be predicted well before the clinical diagnosis. This gives chance to physicians to intervene with treatment in time [4]. Automated analysis of MRI scans with machine learning can facilitate timely diagnosis. A wide variety of traditional, statistical learning methods have been applied in the field of neuroimaging, in some cases outperforming clinicians at predicting AD [8].

In the last decade, however, advances in compute power and the 2012 AlexNet paper - arguably, the most influential paper in Computer Vision - has set off neural networkds (NNs) and deep learning (DL) as the dominating technique to analyze large-scale data in multiple domains (image, speech and text) [9]. Training deep NNs, typically on large datasets enables end-to-end learning of tasks. Their capability to extract hierarchical features from the input data circumvents the previously tedious job of feature engineering, often requiring great domain expertise. Stacking several of these feature extracting layers, higher-level features are composed from the previous layer's output, that can improve the performance of e.g. classification. A robust computer-aided diagnostics system should be able to adapt to various datasets, collected by multiple patient groups, minimizing the biases towards specific groups and discrepancies in data distributions.

However, the common limitations of modern DL are emphasized in medical imaging [11]: 1) to sufficiently train a network from scratch, one needs a large number of human-annotated images, which is resourceful and data is often not publicly available; 2) training on a large number of samples with huge input size is computationally very expensive and slow; 3) deep networks usually demand careful hyperparameter tuning, to avoid under- or overfitting resulting in poor generalization performance (which phenomena can also be a results of limitation 1).

This paper addresses these challenges by deploying using transfer learning. This technique aims at transferring knowledge gained solving some task in domain A (source) to a different, but related target task in domain B (target). Knowledge transfer can be simply achieved by reusing the parameters Θ of a feature extractor fitted in source, and fine-tuning them on the target. According to Yosinki et. al, going deeper in the network, subsequent layers learn more and more specific features, thus one can expect early layers

to have higher transferability [13].

We are interested in the most general *inductive* transfer learning setting: given a source task τ_S and *any* target task τ_T (s.t. $\tau_S \neq \tau_T$) we aim to improve performance of τ_T. Typically, more training samples are available in the source domain and the two data distributions are somewhat similar.

Our proposed framework uses an autoencoder (AE) architecture to capture useful, *domain-invariant* features in the source domain. AE is a neural network with the objective to reconstruct its input $\mathbf{x}$. It has two parametrized components: an **encoder** function $\mathbf{h} = f(\mathbf{x})$, resulting in the hidden layer $\mathbf{h}$ (sometimes called "bottleneck", if its dimension is lower than the input's), and the **decoder** function, that produces reconstruction $\mathbf{x'} = g(\mathbf{h})$. The **code** describes a latent space representation of the input [2]. Since copying the input to the output is not interesting, we can impose certain constraints on the network and hope that minimizing the reconstruction loss $L(\mathbf{x}, \mathbf{x'})$ results in $\mathbf{h}$ capturing useful features [5]. In neuroimaging, a natural choice is to use convolutional layers in $f(.)$ and $g(.)$, resulting in convolutional AEs (CAEs), taking advantage of *weight sharing* and the inherent *translational invariance* of the convolution operations. Ultimately, a simple linear classifier can be stacked on top of the trained encoder, e.g. to output disease probabilities. This stacked encoder-classifier network can then be fine-tuned on the target task.

Our goal is to obtain domain-invariant features. Therefore, our analysis will explore how various modifications to the loss function influences the properties of the latent space.

2 Materials and Methods

2.1 Data

SOURCE In the source domain, our data used for pre-training the CAE was obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database.[1]. For this study, we included 309 subjects (167 AD and 142 healthy controls, respectively). This resulted in a total of 969 T1-weighted MPRAGE magnetic resonance imaging (MRI) scans (AD: 475, CN: 494). Volumes were resized to shape 108x128x108.

TARGET Furthermore, we considered two target datasets to assess the knowledge transfer capability of our proposed framework. Namely, a) the Parkinson's Progression Markers Initiative dataset [2] (PPMI) and b) the ICA iDSS (iDSS) dataset [3]. PPMI consists of unique scans of subjects either diagnosed with PD or being healthy controls (HC),

[1] http://adni.loni.usc.edu/
[2] https://www.ppmi-info.org/
[3] catharina.lange@charite.de

resulting a total of 564 T1-weighted (386 PD, 178 HC) and 329 T2-weighted (225 PD, 104 HC) MRI scans. Samples were resized to shape 105x127x105 and 105x126x105 (T1 and T2, respectively) in order to be of similar shape to the source domain data (ADNI). From the iDSS dataset, we only considered subjects with either cardiovascular dementia (CVD) or AD, resulting in a total of 40 samples (23 and 17, respectively). Samples were reduced to shape 105x127x105 for the above reason.

For all datasets, two additional preprocessing steps were applied on top of the standard MRI processing: 1. skull-stripping by multiplying the volumes with a binary mask; 2. intensity rescaling by applying min-max scaling into the range between 0 and 1. To measure the generalization error, 10% of the data was selected as a hold-out test set for both pre-training and fine-tuning, ensuring to include different subjects in the training and test sets.

2.2 Models

2.2.1 Vanilla Convolutional Autoencoder (CAE)

Our 3D-CAE architecture's encoder consists of three blocks of [Conv3D-ReLU-MaxPool3D] and an additional [Conv-ReLU] mapping to the hidden layer. All convolutional layers use 3x3x3 kernels and have (8, 16, 32, 16) feature maps, respectively. MaxPool layers take the maximum value of 2x2x2 cubes over their input volume.
The decoder mirrors the encoder, but uses ConvTranspose3D and MaxUnpool3D layers and sigmoid non-linearity after the final ConvTranspose3D layer.

2.2.2 Strided Residual Convolutional Autoencoder (SRCAE)

Since training deeper networks becomes increasingly difficult, [6] have proposed *residual networks*, showing that these are easier to optimize and can potentially improve performance by enabling increased depth. As opposed to *plain networks*, these models contain so called skip-connections. In our case, the encoder consists of three EncoderBlocks, each of which performs the following computation:

$$h = Dropout3D(Conv3D(x), p)$$
$$out = StridedConv3D(x \oplus h),$$

where x denotes the input to the block, p is the dropout probability and $\oplus$ denotes the concatenation operator. Using strided convolutions for the output of each encoder helps to reduce the size of the resultoing feature maps while having the beneficial effect of reducing the checkerboard-effect [4]. SRCAE is a smaller model, with around 60% fewer parameters and is influenced by recent deep learning advances, however, the vanilla CAE performed equally well. Therefore, in the following, all results are shown using the simpler, more straightforward architecture.

[4] https://distill.pub/2016/deconv-checkerboard/

3 Results

The results consist of analyzing different aspects of the pretraining phase (i.e. learning the latent space where input is mapped by the autoencoder) since discriminative fine-tuning was not converging with the discussed architectures, transfer learning framework and datasets. Our objective for pretraining is to minimize the MSE (Mean Squared Error) loss measuring the similarity of input and output. In the fine-tuning stage, we used the cross-entropy loss. For all training, we minimize the objective function with stochastic gradient descent (in batches of 16 samples) and the backpropagation algorithm, using the Adam optimizer [7]. with learning rate 10^{-3} on a single NVIDIA GTX 1080 GPU. Convolutional layers follow the Kaiming-Normal initialization with 0 bias (He. et al. 2015).

3.1 Latent space

The characteristics of the learned latent space will be explored by adding penalty terms to the optimized loss function (explicit regularization) or by adding an auxiliary task to the objective function (multi-task learning). For each setting, we train the autoencoder until convergence and visualize aspects of the latent space on a hold-out test set.

3.1.1 Sparsity penalty (L1 regularization)

Our loss can be modified to result in sparsely-coded latent space. Simply, we add the L1-norm of the weights to the default loss, $\mathcal{L} = loss_{MSE} + \lambda \sum_i |w_i|$, with hyperparameter λ. This term has a constant derivative, therefore in each weight-update it can change our weights with a given amount. This can drive unimportant weights to zero; ultimately, certain weights are turned off. This regularization practically performs feature selection. This is demonstrated on 3.1: L1 regularization resulted in a sparse, unimodal activation histogram compared to the unpenalized solution. Sparsity can let us to use larger, more expressing latent space.

3.1.2 Multi-task learning

Extracting discriminative features (i.e. features that projects input from the same class to a similar region of the latent space) can be facilitated by introducing the concept of multi-task learning; this simply means, that we modify our original reconstruction loss (namely, the MSE) with a task-specific component. Here, a fully connected classifier head is added on top of the encoder. This head is trained to discriminate samples from different classes in a supervised fashion; together with the original decoder, this modification results in multi-task learning. This results in the objective $\mathcal{L} = \alpha loss_{MSE} + (1 - \alpha) loss_{CE}$, the weighted sum of MSE and the cross-entropy loss. Looking at 3.2, the 2-dimensional latent space projections of our data shows a better separability if we include the cross-entropy auxiliary loss.

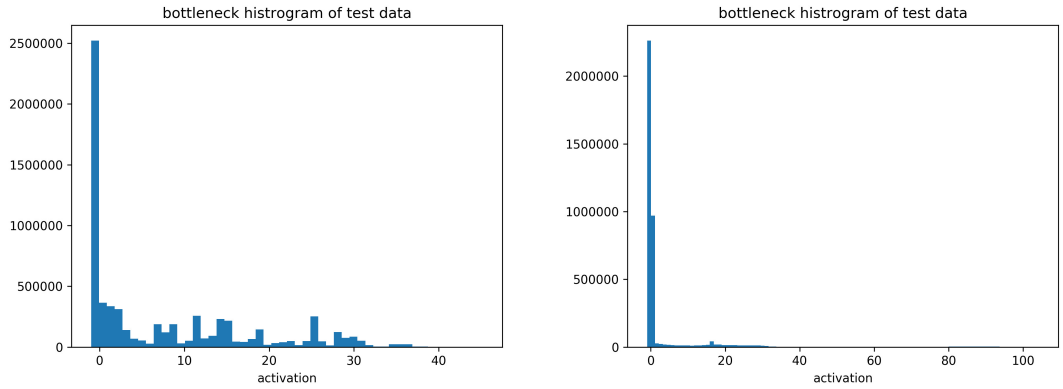


Figure 3.1: Activation histogram in the latent space. The histograms were obtained by averaging the activations of the ultimate layer of the encoder over the entire test set. Left: using MSE loss, right: using MSE loss and L1-penalty.

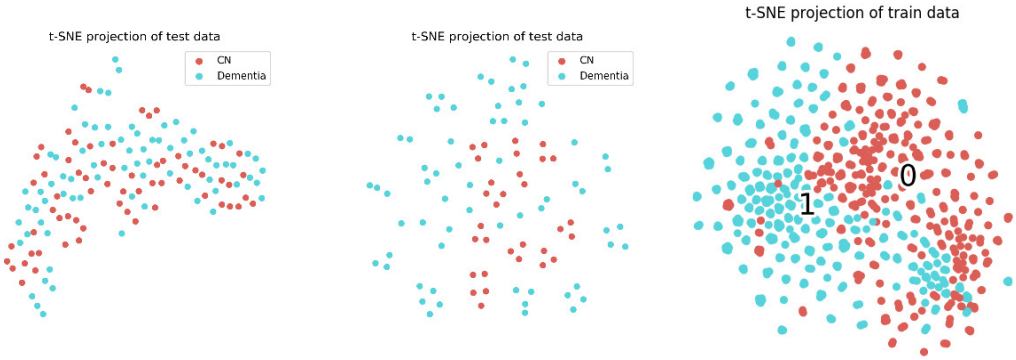
3.1.3 Fine-tuning

A fully-connected (linear) classifier head is placed on top of the source-domain pre-trained encoder body. Initially, we only train the fully-connected head; after certain number of epochs, the encoder (thus far, acting like a fixed feature extractor) is unfreezed, to further adapt the network to the target domain. Performance is reported on the test set of unknown scans. Transfer learning was performed in two different configurations: i) from ADNI to PPMI T1 or iDSS; ii) from PPMI T1 to PPMI T2. Configuration i) is immensely challenging, since the scans in target were acquired using different scanners and processing pipeline. What can be hoped in this setting is that some high-level, generic features are extracted in the source domain, which can make fine-tuning the target task easier (also called *warm starting*, i.e. initializing the network with not fully random weights). ii) on the other hand is a more realistic transfer learning scenario, since the subjects' brains are expected to exhibit biomarkers in similar regions and the data acquisition stage is more comparable between the source and the target. Unfortunately, we were unable to achieve convergence in the scope of this study, therefore no useful empirical results are available for fine-tuning experiments. The potential reasons are discussed and possible improvements or ideas are proposed in the Discussion section.

3.2 Discussion

The effect of modifying our loss has been demonstrated and this strategy can be useful in some cases. Introducing L1-regularization acts as a built-in feature selector and finds sparse solutions, allowing us to use a larger latent space with more expressive power, while still not only memorizes data. Multi-task learning enables us to incorporate more and more domain knowledge into the pretraining phase, if, for instance our samples are labeled by sex, age and disease. However, our original aim was to investigate general-

Figure 3.2: ADNI data projected into the latent space. These projections were obtained using the tSNE algorithm in 2-dimensions. Each dot denotes a sample in the test set for the left and middle figures and a sample in the training set for the right figure. Left: standard MSE loss, middle, right: MSE loss with auxiliary cross-entropy loss.



domain pretraining, this approach could be very useful, if we aim to transfer between not-too-distant domains, for instance, between one Alzheimer'z dataset to another.

Although the idea of general-domain pre-training followed by a fine-tuning stage in target domain is now widely popular in multiple fields of deep learning (ImageNet models for object recognition, language models in Natural Language Understanding), we didn't succeed to transfer knowledge from a general domain to a more task-specific one. Most likely, training a deterministic autoencoder did not induce a useful hypothesis space for the investigated target tasks. By adding the auxiliary cross-entropy objective, our latent space representation managed to somewhat capture discriminative features in latent space; this could possibly lead to a better-than-random initialization for supervised fine-tuning on related target tasks.

To improve results, one could train different layers to different extents; since deeper layer extract most task-specific features, higher learning rates could result in faster domain-adaptation when fine-tuning.

Furthermore, a larger and more general-domain dataset, such as the UK Biobank could be a better candidate for the task-agnostic pretraining phase, allowing to capture more varying, generic features of the brain.

Generative models, such as variational autoencoders (VAEs) are more suitable at modeling data distributions and thus induce a hypothesis space from which transfer learning is more feasible.

3.2.1 Concusion

Transfer learning for MRI is a very difficult task. The input space is huge; variability is present at the level of data acquisition and processing, as well as at the biomarkers of diseases across subjects. Representation learning can come handy, if labelled samples are scarce; however they are only useful as a pretraining step, if the extracted features are useful. General-domain pretraining followed by discriminative fine-tuning is a promising idea that could alleviate the problem of having small sample sizes for neuroimaging datasets; however more expressive, perhaps generative models and larger, more diverse source datasets are probably needed to achieve robust domain adaptation.

4 Code

To facilitate further experimentation with transfer learning, all code is available in the `trans-mri` [5] Python module for the PyTorch framework. The module can be directly installed from Github with a single terminal command. This module allows end-to-end transfer learning by implementing convenient dataset handlers and modular model definitions for neuroimaging data. Further datasets and models can be easily added to it in the future.

`MRIDataset` Container object of paths of scans and the corresponding labels (optional).

`DataBunch` The main data object, which contains training and test `MRIDatasets` that are split and built from an input CSV-file, together with the corresponding iterators. Balanced-splitting, grouped-splitting (ensuring that patients used for training are not included in the test set), masking, zooming, transformations and caching. Furthermore, `DataBunch` implements a bunch of useful methods:

- build_dataloaders: re-batchify the training and test iterators
- normalize: normalize datasets with the mean and standard deviation obtain from the training set
- show_sample: show a random masked, processed training sample
- print_stats: print descriptive statistics of the training and test data (e.g. no. of images/patients in each category and set)
- save/load: to serialize (into a pickle dump) the processed DataBunch and load it back later; useful for reproducibility.

Three datasets {ADNI, PPMI, IDSS} can be simply loaded with the respective `get_{dataset_name}` helper functions.

[5] https://github.com/ben0it8/trans-MRI

LEARNER This object connects a DataBunch together with a PyTorch model and a device, enabling model fitting, logging, inference and visualizations. Supports three model_types {classifier, autoencoder, mixed}, upon instantiation, the respective loss functions, handlerss, trainers and evaluators are initialized. Important method the user might interact with:

- init(DataBunch, model, path, loss_fn, opt_fn, model_type, device): initializes a Learner
- from_pretrained(DataBunch, model, path, pretrained_model, model_type): initialized a Learner from a pretrained PyTorch model.
- fit(epochs, lr, wd, l1_reg): Fit model on training data, optionally running validating after each epoch on the test set. Supports various regularization (wd: weight-decay, l1_reg: l1 penalty), early stopping, reconstruction visualization and logging. Saves the entire model and the encoder to path.
- predict: Returns the predictions and the confusion matrix of the test set.
- encode_data(to_encode): Encodes the *to_encode* set, either on gpu or cpu.
- plot_history: Plots training (with optionally validation) curves over epochs.
- plot_TSNE: Computes and plots the two-dimensional t-SNE projection of the encoded test (or training) samples.
- plot_PCA: Computes and plots the two-dimensional t-SNE projection of the encoded test (or training) samples.
- plot_bottleneck_histogram: Computes and plots the distribution of activations at the bottleneck (i.e. the code).
- freeze_encoder: Freezes the encoder part of model.
- unfreeze_encoder: Unfreezes the encoder part of model.
- del: Deletes model and DataBunch, saving up memory on device.
- reset_model: Deletes model checkpoint (saved after fitting is finished) and reinitializes model weights.
- save(name): Saves model to path/model_dir/name.
- load(name): Loads model from path/model_dir/name.

Three factory functions can be used to initialize a Learner without specifying anything more than a DataBunch and a model:

- create_classifier: Initializes a classifier Learner.
- create_autoencoder: Initializes an autoencoder Learner.
- create_classifier_from_encoder: Initializes an classifier from a pretrained encoder.

MODEL Contains predefined architecture classes, each designed with the consideration of transfer learning, following a unified implementation. This means that autoencoders

have encoder and decoder container attributes (i.e. collections of layers); this allows saving the encoder weights after training and to transfer them as a feature extractor for classification later. The implemented models are:

- `ConvAutoEncoder`: Consists of encoder and decoder modules (optionally, a classifier on top of the encoder in mixed mode)
- `StridedConvAutoEncoder` Consists of encoder and decoder modules (optionally, a classifier on top of the encoder in mixed mode)
- `EncoderClassifier` Consists of encoder (optionally pretrained) and classifier modules

References

[1] A. Association et al. 2014 alzheimer's disease facts and figures. *Alzheimer's & Dementia*, 10(2):e47–e92, 2014.

[2] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36, 2012.

[3] S. Bonin-Guillaume, D. Zekry, E. Giacobini, G. Gold, and J. Michel. The economical impact of dementia. *Presse medicale (Paris, France: 1983)*, 34(1):35–41, 2005.

[4] Y. Ding, J. H. Sohn, M. G. Kawczynski, H. Trivedi, R. Harnish, N. W. Jenkins, D. Lituiev, T. P. Copeland, M. S. Aboian, C. Mari Aparici, et al. A deep learning model to predict a diagnosis of alzheimer disease by using 18f-fdg pet of the brain. *Radiology*, page 180958, 2018.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] D. Kingma and J. Ba. Adam: a method for stochastic optimization (2014). *arXiv preprint arXiv:1412.6980*, 15, 2015.

[8] S. Klöppel, C. M. Stonnington, J. Barnes, F. Chen, C. Chu, C. D. Good, I. Mader, L. A. Mitchell, A. C. Patel, C. C. Roberts, et al. Accuracy of dementia diagnosisâĂŤa direct comparison between radiologists and a computerized method. *Brain*, 131(11):2969–2974, 2008.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] G. McKhann, D. Drachman, M. Folstein, R. Katzman, D. Price, and E. M. Stadlan. Clinical diagnosis of alzheimer's disease report of the nincds-adrda work group* un-

der the auspices of department of health and human services task force on alzheimer's disease. *Neurology*, 34(7):939–939, 1984.

[11] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[12] T. Vos, C. Allen, M. Arora, R. M. Barber, Z. A. Bhutta, A. Brown, A. Carter, D. C. Casey, F. J. Charlson, A. Z. Chen, et al. Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990–2015: a systematic analysis for the global burden of disease study 2015. *The Lancet*, 388(10053):1545–1602, 2016.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.