Deep Learning

Michael Harrison

Submitted for the Degree of Master of Science in

Machine Learning

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

August 6, 2018

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Michael Harrison

Date of Submission:

Signature:

Acknowledgement

I would like to thank...

Abstract

ABSTRACT TBD

Contents

List of Figures

1 Introduction

The aim of this project is to explore the use of Deep Learning in a practical situation. Deep Learning techniques have been used in a range of problems, such as image recognition [ref GoogleLeNet], natural language processing [ref Google Neural Machine Translation], playing games [ref AlphaGo] and self-driving cars [ref WayMo], among others. These techniques have often achieved cutting-edge performance on these tasks - for instance, surpassing human performance in the ImageNet task [ref ILSVRC], or beating the world champion Lee Sedol at the game Go [ref AlphaGo win]. As such, Deep Learning is an exciting branch of Machine Learning with the potential to make significant impacts in many areas of life. Indeed Deep Learning is one of (the?) largest areas of active research in Machine Learning at the time of writing [ref numbers of citations].

This project focuses on image recognition, as this is one of the areas where Deep Learning has seen its biggest successes. In particular, we will be applying Deep Learning to the MURA (**mu**sculoskeletal **ra**diographs) dataset, published by the Machine Learning Group at Stanford University [1]. This is a collection of 40,005 X-ray images of a part of the upper extremity - comprising the arm, shoulder, wrist, hand etc. Each image is from one of 14,656 studies of an individual patient, performed at a particular point in time on one of these parts of the upper extremity. Each study was labelled as normal or abnormal at the time of clinical interpretation by a radiologist from Stanford Hospital - and these labels have been published alongside the images themselves. Therefore the aim of this project is to use Deep Learning to perform binary image classification on this set of X-rays - classifying them as either normal or abnormal.

While any model developed as part of this project can only represent a toy solution to this problem, the principle of combining Deep Learning and medicine seems to be a good one. Inded this too is an area of active research, both within radiology [ref radiology], medical imaging [ref other DL/med image work] and medicine more widely [ref DL/med work]. The works cited of course represents only a small fraction of what is being done. Thinking optimistically, Deep Learning can help improve patient outcomes, produce results more quickly, alleviate pressure off medical pracitioners, reduce medical mistakes and improve medical decision-making. As such, this represents good motivation for me to dip my toe into this area as part of my project.

2 Background

This section describes the background of Deep Learning, and the methodology behind the techniques used in this project.

2.1 Deep Learning

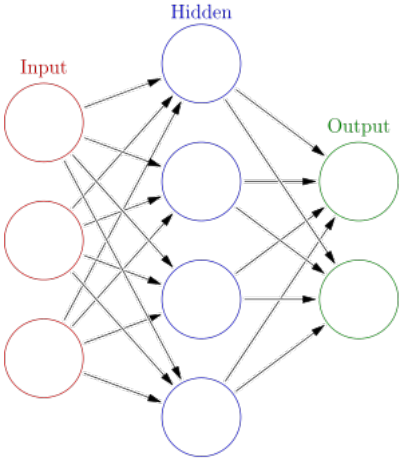
The main idea behind Deep Learning is for a system to learn representations of the data given to it. While these representations can start out simple, they are organised into a hierarchical sequence of layers so that simpler representations from earlier layers can be composed and built up into higher-level, more complex representations. This is repeated over many layers so that the final representations have sufficient information and detail for the task at hand. This process of building up complex representations from simpler parts means that feature engineering - the choice of which aspects of the data to include in the model, how they should be transformed or combined etc - is performed automatically by the system, as part of training. This is one of the great strengths of Deep Learning since feature engineering as a manual process can be difficult and time-consuming, especially when the data is very high-dimensional (as with images).

2.2 Artificial Neural Networks

2.2.1 Basic Structure

More specifically, these Deep Learning systems tend to be artificial neural networks (ANN's), of various forms depending on the nature of the task. A simple version of such a network [2] is shown in **Figure 1** below. These consist of a set of "neurons" (the circles in the image below) organised into layers, with each neuron from one layer connected to all neurons in the next layer. Each neuron has an associated activation, typically just a real number, which is derived from the activations of the neurons feeding into it from the previous layer. The input layer is a special case, where the neuron activations represent values from the item of data being fed into the network. For instance, for greyscale images, the input neurons correspond to each pixel in the given input image, and the input activations are simply each pixel's value. The activations of the final output layer may represent different things, depending on the task - for instance, the final learned representation of the data item when doing dimension reduction. Or, for classification, the probability distribution over the set of classes that captures the network's prediction of which class the given data item is in.

Figure 1: Example of a simple artificial neural network



The connections between the neurons from one layer to the next represent the network's weights, each weight again usually just a real number. In addition to these weights between layers, each neuron (aside from the input layer) has an associated bias, once again usually just a real number. Taken across the whole network, these weights and biases form the set of parameters that define the model. They dictate how the activations from neurons in the previous layer should be used to calculate the activations of neurons in the next layer. For instance, let the activations of the neurons in layer i be given by the vector a_i, and the weights connecting neurons in layer i to those in layer $i+1$ be given by the matrix $\mathbf{W_{i+1}}$. So if the numbers of neurons in layers i and $i+1$ respectively are n_i and n_{i+1}, then a_i will have dimension $(n_i, 1)$, and $\mathbf{W_{i+1}}$ dimension (n_i, n_{i+1})[1]. Then the activations for the neurons in layer $i+1$ are given by:

$$a_{i+1} = f_{i+1}(\mathbf{W_{i+1}}^\top a_i + b_{i+1})$$

where:

- b_{i+1} is the vector of biases for the neurons in layer $i+1$ (so of dimension $(n_{i+1}, 1)$).

[1]Each of the n_i neurons in layer i connects to each of the n_{i+1} neurons in layer $i+1$, making $n_i \times n_{i+1}$ connections in total.

- f_{i+1} is the activation function for layer $i+1$, that applies element-wise to the vector on which it operates, and hence returns a vector. The choice of activation functions is part of the design of the network.

This process is repeated, with activations from the input layer feeding into activations of the first hidden layer; activations from this hidden layer feeding into the next hidden layer; etc until the final activations in the output layer are produced. Note that while **Figure 1** above showed only one hidden layer between input and output, in practice there can be arbitrarily many hidden layers - each taking as input the activations from their previous layer, and sending their own activations as output to the next layer. The choice of the numbers of hidden layers, and the numbers of neurons in each of these layers is again part of the design of the network.

2.2.2 Activation Functions

The activation functions are usually chosen to be non-linear - since otherwise, composing several layers of neuron activations reduces to taking a linear function of the original input neurons and the network itself simply becomes a linear function. To see this, let $f(x) = x$ be the linear activation function used throughout the network. Then if layer 0 is the input layer, the activations of layer 1 are:

$$a_1 = f(\mathbf{W_1}^\top a_0 + b_1) = \mathbf{W_1}^\top a_0 + b_1$$

And hence the activations for layer 2 are:

$$\begin{aligned} a_2 &= f(\mathbf{W_2}^\top a_1 + b_2) && (1)\\ &= \mathbf{W_2}^\top a_1 + b_2 && (2)\\ &= \mathbf{W_2}^\top (\mathbf{W_1}^\top a_0 + b_1) + b_2 && (3)\\ &= (\mathbf{W_1}\mathbf{W_2})^\top a_0 + \mathbf{W_2}^\top b_1 + b_2 && (4)\\ &= \mathbf{W'}^\top a_0 + b' && (5) \end{aligned}$$

where:

- $\mathbf{W'} = \mathbf{W_1}\mathbf{W_2}$ is an $n_0 \times n_2$ matrix
- $b' = \mathbf{W_2}^\top b_1 + b_2$ is an $n_2 \times 1$ vector
- Equation (4) uses the matrix transpose rule $(AB)^\top = B^\top A^\top$

Hence both a_1 and a_2 are linear functions of a_0. This argument can be repeated for any number of hidden layers, and so the output of the whole network (regardless of its size) is simply a linear function of the input. This is undesirable since such a network cannot learn more complicated, non-linear features of the data. Training the network is effectively equivalent to just performing linear regression.

Therefore non-linear activation functions are usually preferred. In this project we have used the following activation functions, in line with standard practice for image classification [ref e.g. VGG16].

ReLU
The Rectified Linear Unit (ReLU) is a scalar activation function that takes a linear function of its argument above a certain threshold m, and is constant otherwise:

$$f(x) = max\{x, m\}$$

Note that m is a hyperparameter, specified in advance of training, rather than learning during it. Typically it is just set to 0 - as otherwise this implies some sort of prior knowledge about the scale of the "pre-activation" values $\mathbf{W}_{\mathbf{i+1}}^{\top} a_i + b_{i+1}$ for all the various layers and neurons in the network (since m could theoretically be set on a neuron-by-neuron basis). This would be difficult to justify, especially when adjusting the level of these pre-activation values is already handled by the bias term b_{i+1}, a parameter that is learned during training.

This activation function introduces a non-linearity at the point $x = m$, since its slope jumps from 0 to 1 as x increases beyond m. While the function is linear above m, this non-linearity at m has proven sufficient in practice to obtain good results with ANN's.

Softmax
The Softmax can be regarded as a vector-to-vector function, that takes as input a vector v, of length (say) n, and produces an output vector w with the same length as v. The j^{th} component of w is given by:

$$w_j = f(v)_j = \frac{e^{v_j}}{\sum_{k=1}^{n} e^{v_k}}$$

where v_j is the j^{th} component of v. Since $e^x \geq 0 \; \forall \; x \in \mathbb{R}$ we have $w_j \geq 0$. And, by definition:

$$\sum_{j=1}^{n} w_j = \sum_{j=1}^{n} \frac{e^{v_j}}{\sum_{k=1}^{n} e^{v_k}} = \frac{\sum_{j=1}^{n} e^{v_j}}{\sum_{k=1}^{n} e^{v_k}} = 1$$

Hence $w_j \in [0, 1] \; \forall \; j = 1, \ldots, n$ and so w represents a probability distribution over the set of values $1, \ldots, n$. Note that this is true regardless of the scale or values of the input v.

This is extremely useful for classification, since we can set the final layer of our ANN to have the same number of neurons as numbers of classes, and let it use the Softmax activation function. Then each neuron would correspond to one of the w_j above, and the activations across the whole output layer would form a probability distribution over the set of classes. The network then provides a means to go from, say, an image to predicted probabilities of which class the image belongs to.

More concretely, for our purposes we set the final layer to have 2 neurons - corresponding to normal or abnormal respectively.

2.3 Convolutional Neural Networks

Write about conv nets, their performance, history etc

2.4 Training

Write about training nets, loss functions, optimisers etc

2.5 Prediction

Write about generating predictions from the networks

2.6 Performance Measures

Write about performance measures - ROC curve, Cohen's Kappa etc Discuss train vs valid performance, overfitting etc.

2.7 State-of-the-Art

What is the current image processing SOTA - e.g. ILSVRC Anything done specifically for radiology?

2.8 Pretrained Models

Discuss the use of pre-trained models - can keep weights the same, replact the head. Describe the structure of the pre-trained models we'll use for comparison: [MAYBE PUT INTO APPENDIX?]

2.8.1 VGG16

Describe VGG16

3 Data

Write about the MURA data, show some exmaple images (normal & abnormal), give data breakdowns

3.1 MURA

Discuss MURA - what it is, how it was produced

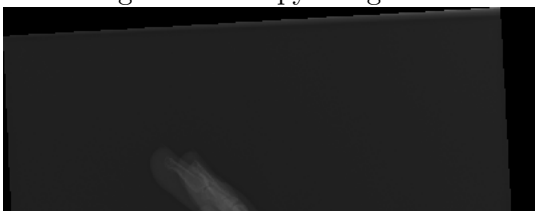
3.2 Image Examples

As can be seen in **Figure 2** below, the x-ray is of a hand. A side-profile of the hand from the same study can be seen in **Figure 3**. **Figure 4** is a copy of **Figure 3**.

Figure 2: Example X-Ray Image

Figure 3: Another example X-Ray Image

Figure 4: A copy of figure 2



3.3 Data Breakdowns

Data breakdowns

4 Model Development

4.1 Working Environment

Describe google cloud set-up, GPU, python packages used etc

4.2 Data Preprocessing

Describe the preprocessing work done on the data - resize images, normalise values, data augmentation etc

4.3 Model

Describe the final ab-initio model I end up with - structure, loss function, training params/hyperparams etc

4.4 Predictions

Discuss getting predictions and aggregating down to study-level predctions

5 Results

5.1 Baseline Performance

Give the performance of my chosen ab-initio model

5.2 Image Size

How do my results vary with size of image? Does increasing image size increase scope for

5.3 Regularisation

Perform some regularisation experiments, show results

5.4 Pre-Trained Models

How does my model compare vs a selection of pretrained models

6 Conclusions

Include introspective chapter Work here not comaparable to clinical setting, e.g. smaller, lower resoution images; radiologist may have a relationship with patient - know medical history, other symptoms etc What is "abnormal"? - type & severity of abnormality not known, MURA paper not clear.

7 Professional and Ethical Issues

Potential Impact on Radiology - Deep Learning tools used to help triage/prioritise radiologists' work, not replace them; Can we trust diagnosis to a computer program? Would you be happy to do so? Conversely - medical errors happen a lot (est. cost $X p.a.; any specifics for radiology?) but DL tools may at least help cut that down.

8 Extensions

Enquire further about what "abnormal" means Alternative data e.g. CheXNet, others(?) Alter NN to accept multiple images simultaneously and so predict based on several views at once (e.g. by weight-sharing, or appening all study images into a single 3D tensor)

References

[1] P. Rajpurkar, J. Irvin, A. Bagul, D. Ding, T. Duan, H. Mehta, B. Yang, K. Zhu, D. Laird, R. L. Ball, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng. MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs. *ArXiv e-prints*, December 2017.

[2] Wikipedia contributors. Artificial neural network — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Artificial_neural_network`, 2018. [Online; accessed 5-August-2018].

A Pretrained Model Architechtures

Below we describe the architechtures of the various pre-trained neural network models mentioned in the text.

A.1 VGG16

This is the VGG16 model.